# COSC 122
# Computer Fluency

## Course Introduction

**Dr. Ramon Lawrence**
**University of British Columbia Okanagan**
**ramon.lawrence@ubc.ca**

## The Essence of the Course

If you walk out of this course with nothing else you should:

**Become a sophisticated user by understanding the basic skills and concepts of Information Technology.**

This course is more than using apps and Office!

We will answer questions like:

- How does the computer and the Internet work?
- What is a program? How do I tell the computer what to do?
- What are the social challenges of an information society?
- How do I become a life-long productive IT user?

This course shows how technology works, the fundamentals of IT, and how to ***think (and create) differently***.



## Technology is For Everyone

It does not matter what discipline you are studying or what job you get in the future, technology is a critical part of your life.

- Business: sales and marketing data analysis and planning
- Science: modern science requires computational experiments
- Arts: digital and artistic creativity, global and social impacts
- Life: Can you live without your phone or the Internet? Can you imagine the technologies in the next 20 years?

Beyond the technology, this course will encourage you to think differently by learning how to communicate precisely, think critically, and problem solve algorithmically.



## My Course Goals

My goals in teaching this course:

- Summarize and document the information in a simple, concise, and effective way for learning.
- Strive for ***all*** students to understand the material and pass the course.
- Be available for questions during class time, office hours, and at other times as needed.
- Provide an introduction to computers, applications, the Internet, and simple programming.
- Help students become fluent computer users with an understanding of a wide variety of applications and the capability of life-long productivity with technology.
- Encourage students to continue with other computer science courses.



## Course Objectives

1) To understand common computer terminology

2) To learn the basics of networking and Internet applications

3) To be exposed to the fundamental concepts of information representation, abstraction, and algorithmic thinking

4) To try simple programming by creating web sites in HTML and JavaScript

5) To use word processors, spreadsheets, and databases to manipulate, document, and analyze information

6) To appreciate the role and effect of IT in society



## Academic Dishonesty

Cheating in all its forms is strictly prohibited and will be taken very seriously by the instructor.

A guideline to what constitutes cheating:

- Assignments
  - ⇨ Working in groups to solve questions and/or comparing answers to questions once they have been solved (except for group assignments).
  - ⇨ Discussing HOW to solve a particular question instead of WHAT the question involves.
- Exams
  - ⇨ All exams are closed book, so no course materials should be present.

Academic dishonesty may result in a "F" for the assignment or course and ***all*** instances are recorded in the Dean's office.

## How to Pass This Course

The most important things to do to pass this course:

- Attend class
  - ⇨ Read notes *before* class as preparation and try the questions.
- Attend the labs and do all lab assignments
  - ⇨ Labs are for marks and are practice to learn the material for the exams.

To get an "A" in this course do all the above plus:

- Practice programming and working with applications.
- Do more questions than in the labs. Practice makes perfect.



## Systems and Tools

Connect is used for a discussion board, for posting marks, and for anonymous feedback.

- Please use the discussion board and feedback survey.

All software is available in the laboratory at SCI 126/FIP 133.



## My Expectations

My goal is for you to ***SHOW UP TO CLASS AND LABS*** and spend the effort to learn the material.

Although this class may be "easy" for some, you will not pass this class without effort and ***attendance***.

- Previous: Avg. mark attending class = 75%, not attending=40%

The course will be very straightforward – If you do the work, you will do well. Some labs teach material on Windows and Microsoft Office, but the web development labs (HTML and JavaScript) will require you to think and work.

**Your mark is 90% perspiration and 10% inspiration.**



## The Lab Assignments

In each lab we will work on computers on a lab assignment.

Lab assignments are worth ***20%*** of your overall grade.

Lab assignments may take ***more than the two hours*** lab time.

You have at ***least one week*** after your lab to complete it.

- No late assignments will be accepted.
- An assignment may be handed in any time before the due date.

Lab assignments are done individually or in groups of two depending on the assignment.

***The lab assignments are critical to learning the material and are designed to prepare you for the exams!***



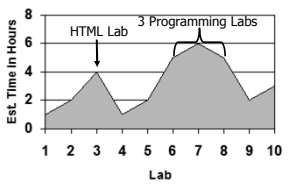
## Lab Workload

The lab is two hours long, but you may require more or less time to complete the lab.

- Some labs will be done very quickly while others will require many hours outside the lab time to complete.
- Be prepared for this difference and use the shorter labs at the start of the class to meet your TA and establish good habits.

Lab difficulty by week:

Average: **3 hours**






## The In-Class Quizzes

To encourage attendance and effort, ***10%*** of your overall grade is allocated to answering in-class questions.

These questions are answered electronically using a clicker.

- The clicker can be purchased at the bookstore and sold back to the bookstore like a used textbook.
- The clicker is personalized to you with your student number.
- At different times during all the lectures, questions reviewing material will be asked. Reponses are given using the clickers.

There will be at least 100 questions throughout the semester. Each question is worth 1 mark, and you need at least 80 right answers to get the full 10%.

- That is, if you answer 60 questions right, you get 60/80 or 75%. Thus, do not worry if you must miss a class or two or forget your clicker one day!

## *Why are you here?*
## *Reasons Why People Take This Course*

A) I want an easy credit.

B) I want an easy Science credit (Arts Majors).

C) I want to learn more about Microsoft Office.

D) I want to learn more about how technology works.

E) I am interested in computing, web development, programming, or future courses.



## *What to Learn*
## *What Topic are You Most Interested In?*

A) Microsoft Office (Word, Excel, Access)

B) How the Internet/Computers Work

C) Building Web Pages using HTML/JavaScript Programming

D) The Effect of Technology on Society

E) None of the above



## *What do you expect?*
## *What Grade are You Expecting to Get?*

A) A

B) B

C) C

D) D

E) F



## *Why this Course is Important*

This course is designed to introduce the fundamental skills and concepts of Information Technology. You will learn to become a sophisticated user that is knowledgeable about how the technology you use works.

Important results:

- ***Office Software Proficiency*** – Every person needs to know how to use basic office software (editors, spreadsheets, and databases). We will cover these fundamental skills.
- ***The Internet and You*** – We will learn the basics of Internet terminology, how it works, and how it effects you.
- ***Web Development*** – We will build simple web sites using HTML and JavaScript.
- ***Deeper Understanding*** – We will see how technology works and appreciate the awesome capabilities, challenges, and opportunities in Information Technology.

# COSC 122
# Computer Fluency

## Computer Terminology

**Dr. Ramon Lawrence**
**University of British Columbia Okanagan**
**ramon.lawrence@ubc.ca**



## Why are you here?
## Reasons Why People Take This Course

A) I want an easy credit.

B) I want an easy Science credit (Arts Majors).

C) I want to learn more about Microsoft Office.

D) I want to learn more about how technology works.

E) I am interested in computing, web development, programming, or future courses.





## What to Learn
## What Topic are You Most Interested In?

A) Microsoft Office (Word, Excel, Access)

B) How the Internet/Computers Work

C) Building Web Pages using HTML/JavaScript Programming

D) The Effect of Technology on Society

E) None of the above





## What do you expect?
## What Grade are You Expecting to Get?

A) A

B) B

C) C

D) D

E) F





## Key Points

1) People do not have any natural technological abilities, so systems are designed to match users previous knowledge about the domain or other systems.

2) Fundamental concepts of information technology:
- abstraction
- generalization
- algorithmic thinking

3) Programming is the process of constructing programs in order to instruct a computer on how to solve problems. It is the act of writing out the steps of an algorithm.





## Why is Terminology Important?
## Why is there so much of it?

Using terminology precisely and correctly demonstrates ***understanding of a domain*** and simplifies communication.

Information technology has many terms because:
- Information technology (IT) is a ***broad*** field.
- IT concepts are often ***virtual*** and described using metaphors.
- IT businesses use ***marketing*** terminology to differentiate and sell their products.
- Abbreviations and ***acronyms*** are extensively used.

## Computers

A ***computer*** is a device that can be programmed to solve problems.

***Question:*** Is a cell phone a computer?

**A)** yes

**B)** no





## Computers Are Everywhere

Computers are not just desktops and laptops but also tablets, smart phones, and embedded chips in consumer electronics, cars, televisions, and appliances.

- ⇨ There have been over 30 billion ARM embedded processors shipped.
- ⇨ There are over 350 million computers sold annually.

***Question:*** If you consider this general definition of "computer", how many "computers" do you own?

**A)** 0

**B)** 1 to 5

**C)** 6 to 10

**D)** 11 to 20

**E)** 21 or more





## Software and Hardware

***Hardware*** refers to the physical part of the computer.

- ◆"Hardware is something that you can hit with a hammer."
- ◆This includes components like:
  - ⇨ Input/Output (I/O) devices – mouse, keyboard, monitor, printer, scanner, sound system
  - ⇨ Storage devices – CD/DVD readers/writers, hard drives, USB drives
  - ⇨ Motherboard, processor, memory, graphics card, sound card, bus

***Software*** is the programs the computer follows to perform functions.

- ◆Software is virtual. Although programs may be stored on media, the essence of software is information.





## Computer Components
## The Monitor

The ***monitor*** is a video screen that displays information stored in the computer's memory. Monitor types:

- ◆**LCD** – liquid crystal displays – slim, flat monitors
- ◆**LED** – light-emitting diode – LCD with power efficient semiconductor backlight source
- ◆**touch/multi-touch** – capacitive touchscreen (human touch distorts electrostatic field) or resistive (force connects layers)

The screen is divided into a grid of ***pixels***. (picture elements)

- ◆Common screen sizes: 1024 x 768 and 1280 x 800
- ◆The more pixels the finer (more detailed) the resolution and the crisper images appear.
- ◆***Pixel density*** is number of pixels in an area. iPhone has 326 pixels/inch compared to about 120 for laptops.





## Screen Resolution

***Question:*** The current screen resolution is 1024 x 768 pixels, and we change the screen resolution to 1280 x 800 pixels. What happens to the text (characters) on the screen:

**A)** get smaller

**B)** get larger

**C)** stay the same size





## iPhone Resolution

***Question:*** A common screen resolution for a 17" monitor is 1280 x 1024. The iPhone5 screen is 4". The iPad4 screen is 9.7". Select a true statement:

**A)** The 17" monitor resolution of 1280 x 1024 is significantly higher than the iPhone5 and iPad4.

**B)** The iPad4 screen resolution is about twice the iPhone5.

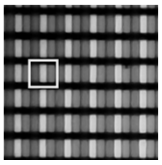
**C)** The resolution of all three displays is very close to each other (within 10%).

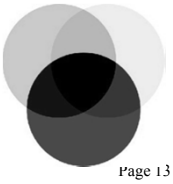
## Computer Components
## The Bitmapped Monitor

A monitor is ***bitmapped*** as each pixel on the screen shows the values of one or more bits in the computer's memory.

- Black and white – only one bit needed (black = 1, white = 0)
- Colors – may have multiple bits representing relative intensities of three primary colors: red, green, blue (RGB)
  - ⇨ Note: Mixing light primary colors is different than pigment primary colors: red, yellow, blue.

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 |









## Computer Components
## Processor

The ***Central Processing Unit (CPU)*** or ***processor*** is the "brains" of the computer as it directs data flow inside the machine and knows how to perform basic operations.

CPU executes program instructions, performs math operations, fetches/stores data in memory, controls data flow of devices.

- e.g. Intel Pentium/Core/Xeon, AMD Athlon/Phenom, Apple A6 (ARM processor) (iPhone)





## Computer Components
## Memory

***Memory*** - is the general term for devices which allow the computer to store data either temporarily or permanently.

- **Temporary memory**: only stores data while the computer is on
  - ⇨ random-access memory (RAM) stores data and programs while the computer is on and is a fast, common type of memory
- **Permanent memory**: data is stored even after computer is off
  - ⇨ read-only memory (ROM) is permanent memory that cannot be changed
  - ⇨ Most permanent memory is considered **secondary storage** because the memory is stored in a separate device (hard drive, DVD, flash).
  - ⇨ Since memory in secondary storage is in a separate device, the device is capable of holding more data, but is often slower than main memory.

***Cache*** - is a term used to describe memory which stores a subset of the memory in a larger memory for performance.

- processor cache (Level 1 & 2), disk cache, network cache





## iPod Memory

***Question:*** Is the memory that stores the songs in an iPod/MP3 player temporary or permanent?

**A)** temporary

**B)** permanent



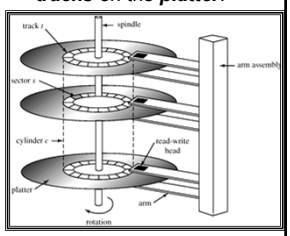


## Computer Components
## Hard Disk

A ***hard drive*** is a permanent secondary storage device.

- It magnetizes areas on the disk. The charge remains even with no power to drive.

The read/write ***head*** is on an ***arm*** that moves to different ***tracks*** on the ***platter***.








## Computer Components
## Flash Memory

***Flash memory*** is used in many portable devices (USB, cell phones, music/video players) and also solid-state drives.

- Flash memory is permanent memory.

Flash memory replaces random access memory in portable devices. It can also be used for secondary storage (USB devices) or to replace hard drives.

Flash drives have many benefits over hard drives including:

- Increased performance (especially random reads)
- Better power utilization
- Higher reliability (no moving parts)

## Aside: What is Disk Defragmentation?


A computer tries to store your files in one contiguous block on the hard drive but may not be able to.

Each piece of a file is called a fragment and a table is used to keep track of where all the fragments of a file are.

The ***error checker*** will find physical errors and logical errors in files.

The ***disk defragmenter*** will try to combine fragments at various locations on the disk into one larger fragment in order to improve performance.





## Research Question
## Solid State Disk Defragmentation

***Question:*** TRUE or FALSE: Disk defragmentation should be performed on solid state disks (SSDs) just like hard drives (HDs).

**A)** true

**B)** false





## Sequential vs. Random Access

RAM, hard drives, and flash memory allow random access. ***Random access*** means that you can access any location in any order.

Tape drives and VCR tapes allow sequential access. ***Sequential access*** means that you can only get to a particular location by visiting previous locations in sequential order.

- That is, you cannot skip ahead, but must go through the tape in order until you reach the desired location.





## Computer Components
## Memory Size

***Memory size*** - is a measure of memory storage capacity

- Memory size is measured in **bytes**.
  - Each byte contains 8 **bits** - a bit is either a 0 or a 1.
  - A byte can store one character of text.
- Memory sizes are measured in:
  - kilobytes (KBs) - 1,000 bytes (one thousand)
  - megabytes (MBs) - 1,000,000 bytes (one million)
  - gigabytes (GBs) - 1,000,000,000 bytes (one billion)
  - terabytes (TBs) - 1,000,000,000,000 bytes (1,000 billion)

Various memory devices and their storage capacities:

- RAM (Main memory) : 2 GB to 256 GB
- Hard Drive : 100 GB to 2 TB
- CD-ROM/DVD: 640 MB / 10 GB



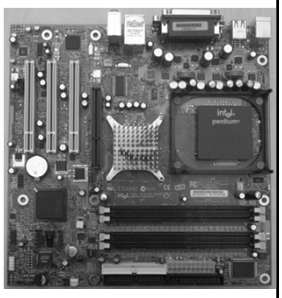


## Computer Components
## Motherboard

The ***motherboard*** is a circuit board that connects components including the CPU, memory, and devices.

The ***bus*** is a set of wires that interconnects the components.

- e.g. When the CPU requires data from memory, the data is sent over the bus from the memory to the CPU.
- The bus is the freeway in the system and can be a ***bottleneck*** if it cannot transmit data as fast as the CPU and other devices require.







## "The Cloud"

"The Cloud" is not part of your computer but rather a network of distributed computers on the Internet that provides storage, applications, and services for your computer.

These systems and services simplify tasks that otherwise would be done by programs on your computer.

Examples:

- **Dropbox** is a cloud service that allows you to store your files on machines distributed on the Internet. Automatically synchronizes any files in folder with all your machines.
- **iCloud** is an Apple service that stores and synchronizes your data, music, apps, and other content across Apple devices.

## Research Question Cloud Computing

***Question:*** What company had the largest cloud computing company based on revenue in 2012? Consider only revenue from cloud computing services.

**A)** Microsoft
**B)** Apple
**C)** Amazon
**D)** Google
**E)** IBM





## Algorithm

An ***algorithm*** is a precise and systematic method for solving a problem.

**Exercise:** With a partner, describe how you would find a person's name in a list of names sorted by last name. Assume your partner does not know very much!

Remember algorithms must be precise!





## Algorithm

***Question:*** Put the following steps in order to write an algorithm to construct a camp fire.

**1)** light match
**2)** place wood in fire pit
**3)** put match on wood
**4)** gather wood

**a)** 2,4,3,1
**b)** 4,2,1,3
**c)** 1,2,3,4
**d)** 4,3,2,1





## Programming

What is programming?

- ***Programming*** is the process of constructing programs in order to instruct a computer on how to solve problems. It is the act of writing out the steps of an *algorithm.*
- A ***program*** is a sequence of simple computer instructions in some *language* which tell the computer the necessary steps to solve a problem or complete a task.
- A ***language*** is the structure and syntax used to communicate to the computer the tasks it is required to perform.

We all "program" by giving instructions to others!





## Abstraction

***Abstraction*** focuses on the key concept while ignoring details.

Examples:

- We ignore details around us to focus on "the task at hand."
- As users we do not see the details on how a system works when we use it.
- When building a system or solving a problem, we focus on a particular component or piece at a time.
- Children's stories often have a moral that is independent of the story characters.





## Generalization

***Generalization*** is applying a common idea or concept in many different situations.

- Note: Generalizations may not apply in every single situation. There may be "exceptions to the rule."

Examples:

- Cars generally have their pedals/controls in the same locations.
- Caps usually twist left (counter-clockwise) to loosen and right (clockwise) to tighten.

## Analytical Thinking

***Analytical thinking*** uses specific, quantitative facts.

- Non-analytical statement:
  - ⇨ The world record in the mile run has improved.
- Analytical statement:
  - ⇨ The world record in the mile has improved from 3.59.4 in 1954 to 3.43.13 in 1999, a 7% improvement.





## Computer vs. Human Improvement

How much faster have computers become?

| Computer | Year | Speed (ops./second) | Improvement |
|---|---|---|---|
| UNIVAC 1 | 1951 | 2000 | |
| IBM 650 | 1954-1962 | 2500 | 25% |
| IBM S/360 | 1964-1978 | 1,000,000 | 850 times |
| Apple II | 1977 | 1,000,000 | 850 times |
| Commodore64 | 1982 | 1,000,000 | 850 times |
| PC 486 (50 MHz) | 1994 | 40 million | 20,000 times |
| iPhone4 ARM Cortex A9 | 2009 | 5,000 million | 2.5 million times |
| i7Core PC (3.4 Ghz) | 2011 | 160,000 million | 80 million times |
| K Computer | 2011 | 8 quadrillion | 4 trillion times |





## Technological Ability is from Experience not Genetics

People do not have natural technological abilities.

Our experience using systems helps us know what to expect. Designers who create devices know about this experience and design products to match what we already know.

Understanding how a system works allows us to be more effective users.

- e.g. By knowing that lids usually twist counter clockwise to loosen, we know which way to twist if they are stuck.

Question: When you get a new gadget do you read the manual first or starting using it right away? Does it depend on what type of gadget it is?





## Designing Software for Users

Products are designed to make it simpler for users to use them.

Software designers use two key ideas:

- 1) Users have *knowledge of the domain* of the software including prior experience with non-computer products.
  - ⇨ E.g. The ***desktop*** environment on a computer is a *metaphor* as working at a computer is similar to working at a desk. Now everything is **touch**!
  - ⇨ Question: What do these buttons do? 
- 2) Users have *knowledge of other software* and user interfaces that can be transferred to a new application if developed consistent with this prior experience.
  - ⇨ e.g. command buttons, sliders, etc.





## User Interface Design Goals

1) Strive for familiarity and consistency
- Exploit users knowledge of domain and other software

2) Chose good mappings and metaphors
- Proper use of color, spatial, and organization cues

3) Provide useful feedback
- Let the user understand what is going on
- e.g. Indicate that the computer is still working on a task (change cursor) or action occurred (button animation).

4) Manage complexity
- Show the right amount of information required for the task and make operations simple to perform and remember.

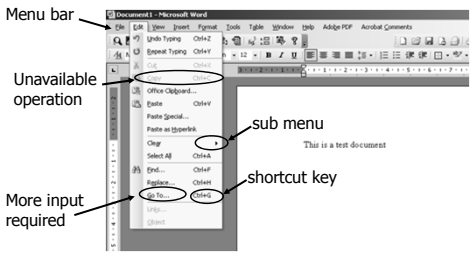




## Standard Interface Components - Menu

A ***menu*** is a list of operations the software can perform. The operations are grouped by function and shown in a ***menu bar***.

- Menus on the top bar are called *pull-down* or *drop-down* menus.



## Experimenting with Software

The key to being a expert user is to:

- be willing to apply past knowledge to learn new software
- be willing to experiment and test software features

The easiest way to learn software is to experiment with its features and interface. Nothing will break… usually…

Watching others is another good way to learn.





## Virtual World

The virtual world and experiences provided by computers is limited only by creativity and imagination.

Although our interactions with computers is based on familiar, real-world concepts and abilities, computers provide new opportunities and experiences not controlled by physical reality.

Examples:

- Communications: Facebook, Twitter, messaging, email
- Virtual realities: 3D experiences, online games
- Creativity: Almost anyone can create art or music or videos and share with a world-wide audience.





## Technology: Taking IT Personally

When learning a new software program ask yourself:

- What do I have to learn about this software to do my task?
- What does the designer of this software expect me to know?
- What additional information does the software need to do its task?

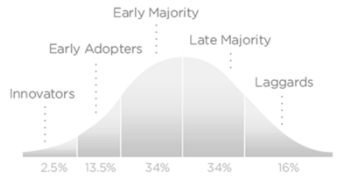
To evaluate if you need to change your IT use, ask yourself:

- Is there IT that I am not now using that could help me?
- Am I more or less productive using this technological solution?
- Can I customize the technology to improve my productivity?
- Have I assessed my uses of information technology recently?





## Innovation Adoption Lifecycle




INNOVATION ADOPTION LIFECYCLE

Innovators – seek new solutions and take risks to gain advantages
Early adopters – leaders who will go before the crowd
Early majority – slower adoption ; adopt when peers do ; "group think"
Late majority – innovation skeptics ; follow crowd after
Laggards – do not want to change ; traditional





## Innovation Adoption

***Question:*** Which of the categories for innovation adoption do you fall in?

**A)** Innovators
**B)** Early adopter
**C)** Early majority
**D)** Late majority
**E)** Laggards





## Is There Any Money in IT?

The opportunities to profit from IT knowledge are enormous. There are numerous IT jobs and opportunities for businesses.

| Job | Salary | Description |
|---|---|---|
| IT support | $35-75,000 | Technical support for users |
| Computer trainer | $35-50,000 | Train users on software/hardware |
| Database Admin | $55-100,000+ | Develop/maintain databases |
| Data entry staff | $20,000+ | Input information into systems |
| Systems manager | $80,000+ | Manager position, CIO |
| Network admin. | $50-95,000 | Manage organization network |
| Programmer | $60-100,000+ | Develop and test software |
| Software engineer | $50-100,000+ | Design software systems with users |
| Technical writer | $40-80,000 | Write user documentation for systems |
| Webmaster | $50-75,000 | Develop web sites and marketing |
| IT Business | $$$$$ | Easiest way to be a millionaire… |

## Conclusion

A computer consists of numerous components, but as users we can normally ***abstract*** away the hardware internal functions.

Since a computer is very fast but not very smart, a computer must be given instructions or programs in the form of ***software***.

Software is developed by programming an ***algorithm*** in a language that the computer understands. Programming involves specifying precisely the sequence of operations and representation of information used.

We become more effective users of technology if we use the correct terminology, understand how systems work, and are confident on using prior knowledge to learn new systems.





## Objectives

- Explain why it is important to understand and use IT terminology.
- List some reasons why there are so many IT terms.
- Define: computer, hardware, software
- Define: monitor, LCD, pixel, bitmapped
- Define: processor, memory (temporary/permanent), cache
- Compare: random vs. sequential access
- Define: motherboard, bus
- Define: algorithm, program, language, programming
- Define: abstraction, generalization, analytical thinking
- List and explain four ideas designers use to make their software easier for us to use.
- Explain the characteristics of an expert user.
- List and explain the five steps in the innovation lifecycle.





## Review
## Memory – Temporary or Permanent

***Question:*** Is main memory (RAM) in your computer temporary or permanent?

**A)** temporary

**B)** permanent





## Review
## Memory – Temporary or Permanent

***Question:*** Is your hard drive considered temporary or permanent memory?

**A)** temporary

**B)** permanent





## Review
## Sequential vs. Random Access

***Question:*** What device performs sequential access?

**A)** main memory (RAM)

**B)** DVD

**C)** VCR

**D)** iPod

**E)** hard drive





## Review
## Memory Size

***Question:*** Which is bigger?

**A)** 10 GB

**B)** 100 MB

**C)** 1,000,000,000 bytes

**D)** 1 TB

## *Review*
## *Programming*

***Question:*** Match the programming related terms with related terms in cooking.

**Programming, Language, Algorithm, Program**

**1)** Cooking **2)** Recipe written in French **3)** English
**4)** Recipe **5)** Writing a cook book

**A)** 1,3,2,4
**B)** 5,3,4,2
**C)** 5,3,2,4
**D)** 1,3,4,2





## *Review*
## *Hard Drive Terminology*

***Question:*** Put the following hard drive terminology in order of smallest to largest size:

**platter, sector, cylinder, track**

**A)** platter, sector, cylinder, track

**B)** sector, cylinder, track, platter

**C)** sector, track, cylinder, platter

**D)** sector, track, platter, cylinder

# COSC 122
# Computer Fluency

## Networking and the Internet

**Dr. Ramon Lawrence**
**University of British Columbia Okanagan**
**ramon.lawrence@ubc.ca**



## Key Points

1) Networks allow computers to communicate information.

2) Communication requires a shared medium, a common language, and a protocol.

3) TCP/IP is the standard protocol for computers on the Internet.

4) The Internet and computers have made a significant impact on our lives, both positive and negative.





## What is Communication?

***Communication*** is the act of sending information from one party to another.

A ***sender*** transmits the information to one or more ***receivers***.

For communication to be effective we need:

- a ***shared medium*** accessible to both senders and receivers
- a ***language*** or encoding for representing the information sent
- a ***protocol*** or set of rules explaining how the medium is used by both the sender and the receiver

Example: What are the medium, language, and protocol used in a classroom lecture like this one?





## Types of Communication Synchronous vs. Asynchronous

Communication can be categorized in several ways.

***Synchronous communication*** is when the sender and receiver are active at the same time.

- e.g. telephone call, instant messaging

***Asynchronous communication*** is when the sending and receiving occur at different times.

- e.g. email





## Types of Communication Broadcast vs. Point-to-Point

In ***broadcast communication*** (or multicast) there is a single sender and many receivers.

- e.g. cable and satellite television

In ***point-to-point communication*** there is a single sender and a single receiver.

- e.g. telephone calls





## Aside: Spam

***Spam*** is unsolicited e-mail.

Marketers can flood a person's mailbox with unwanted e-mail messages. By design, the receiver is forced to accept these messages which are most likely deleted or filtered out by their e-mail software (spam filter).

- ***White list*** – accepted sender ; ***black list*** – rejected sender

Spam may have derived from a Monty Python skit in which the word "spam" was chanted by Vikings in a restaurant and their chants drowned out other conversation.

Question: How does the Vikings' chants relate to the three communication issues of shared medium, language, and protocol? Which of these were they exploiting?

## Medium, Language, and Protocol

***Question:*** Fill in the blanks: The Vikings' repetition of "spam" drowned out other conversations because they were not following the _____ for the _____.

**a)** language / medium

**b)** medium / protocol

**c)** language / protocol

**d)** protocol / medium





## Practice Questions

Determine if the following are *synchronous* or *asynchronous* and *broadcast* or *point-to-point*:

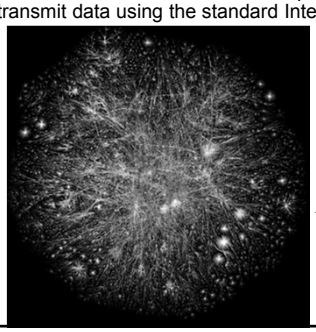
- radio
- classroom lecture
- instant messaging
- e-mail
- telephone call
- postcard
- whispering to another person
- wireless Internet (challenging)
- Others?





## What is the Internet?

The ***Internet*** is a collection of computers and networks that transmit data using the standard Internet Protocol (IP).



Visualization of Internet (2005)
Colors represent domains
Lengths represent delays





## Internet Layers

The Internet is divided into layers to handle its complexity.

***Layering*** is a technique for dealing with complex systems where each layer provides services for the layers above and uses services of the layers below.

Example: Postal mail system. Three layers:

- person layer (send letter)
- mail carrier layer (distribute mail in area)
- distribution layer (planes, trucks)
- You as a person sending a letter do not worry about how the letter gets to its destination as long as you address it correctly.





## Five Internet Layers

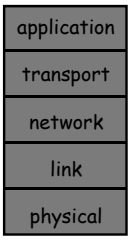
***application***: supports messages between programs
- e.g. HTTP between browser and server

***transport***: process-to-process data transfer
- e.g. TCP – guaranteed message delivery

***network***: send *packets* from source to destination
- e.g. IP – send message to any machine

***link***: data transfer between neighbors
- e.g. Ethernet – communicate within building

***physical***: encoding of bits on medium
- e.g. send signals over CAT5 wire








## The Internet's Communication Properties

The Internet is an ***asynchronous***, ***point-to-point*** communication system.

However, the speed of electronic communications allows for the development of applications on the Internet that appear synchronous and for information to be broadcast to many users.

Examples:

- Point-to-point, asynchronous – email
- Point-to-point, synchronous – instant messaging
- Broadcast, asynchronous – web pages, blogs
- Broadcast, synchronous – chat rooms

## How do you get on the Internet?

For your computer to communicate with others on the Internet, you need the three basic communication components:

- shared medium, language, and protocol

The medium is either a wire or wireless link.

- Your ISP may be a phone, cable, or satellite company.
  - ⇨ The medium is the phone/cable wire entering your house or air waves if satellite/cellular transmission.
  - ⇨ Your computer communicates with an ISP computer which relays information to and from the Internet on your behalf.

The communication format (language) and protocol used is called TCP/IP (Transmission Control Protocol/Internet Protocol).





## I am on the Internet… Now what? IP Addresses

A computer on the Internet is given a unique identifier called an ***Internet Protocol*** (***IP***) ***address***.

- An IP address is similar to your telephone number.

An IP version 4 (IPv4) address consists of 4 numbers in the range of 0 to 255. The numbers are separated by dots.

- Example: 142.231.95.1

Since there are an increasing number of computers and devices being added to the Internet, there is an ongoing transition to IP version 6 (IPv6) addresses which have 16 numbers from 0-255 represented in hexadecimal.

- Example: 2002:CE57:25A2:0000:0000:0000:CE57:25A2





## IP Addresses and Domain Names

Although IP addresses are unique, they are even harder to remember than phone numbers. ***Domain names*** are text names for computers that are easier to remember.

- A ***domain*** is a related group of networked computers.

Domain names are organized ***hierarchically***. The most general part of the hierarchy is at the end of the name.

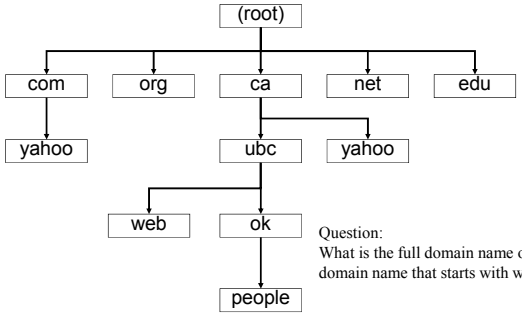
Example: `people.ok.ubc.ca`

- `ca` – Canadian domain
- `ubc` – University of British Columbia
- `ok` – Okanagan campus
- `people` – name of computer/server on campus





## Visualizing the Domain Hierarchy




Question:
What is the full domain name of the domain name that starts with web?





## Top-level Domains

Top-level domains appear in the last part of domain name:

- com commercial organizations
- edu educational institutions
- org organizations
- net networks
- mil military
- gov government agencies
- ca Canadian domain

Top-level domains are controlled by ICANN. Many new domains have been recently approved (2011) allowing for considerable freedom in naming. e.g. .soda or .pizza

- Controversial one: .xxx for pornography sites





## DNS Servers

The ***Domain Name System*** (***DNS***) translates the human-readable names into IP addresses. There are DNS servers on the Internet which provide this mapping function.

- A DNS server has a similar function as a phonebook.

Each Internet computer knows the IP address of its nearest DNS server. When you use a domain name in a request, your computer asks the DNS server to look up the IP address.

If the closest DNS server does not know the IP address, it asks a ***root name server***, which keeps the master list of name-to-address relationships.

- There are 13 root name servers (with multiple mirrored instances) distributed across the globe.

## DNS Root Servers

Source: http://www.root-servers.org





## Review
## Synchronous vs. Asynchronous

***Question:*** Select ***one*** that performs synchronous communication.

**A)** email

**B)** letter

**C)** telephone call

**D)** television





## Review
## Broadcast vs. Point-to-Point

***Question:*** Select ***one*** that performs broadcast communication.

**A)** radio

**B)** classroom lecture

**C)** telephone call

**D)** email





## Review
## IP Address

***Question:*** Which one of the following is a valid IP4 address?

**A)** 0.0.0.0

**B)** 255.255.255.255.255.255

**C)** 1.2.3.256

**D)** 111.222.3456





## Review
## Domain Names

***Question:*** Which part of the address **people.ok.ubc.ca** is the largest (most general) domain?

**A)** people

**B)** ok

**C)** ubc

**D)** ca





## The World Wide Web (WWW)

The ***World Wide Web*** (***WWW***) is an application built on top of the Internet that allows for the display and transmission of documents called web pages.

- Developed by Tim Berners-Lee at CERN in 1991.

A ***web page*** is a document that contains mark-up that allows it to be displayed graphically by a ***web browser***. The page may also contain ***hyperlinks*** to link to related web pages.

A ***web server*** is a computer on the Internet with the task of storing web pages and responding to clients' requests for them.

The World Wide Web (WWW) is the web servers and the files they store.

## Clients and Servers

A ***server*** is a computer that stores information such as a web page, e-mail, database, etc.

A ***client*** is a computer that requests information stored at a server.

When you click a hyperlink in your browser, your computer becomes the client and requests the appropriate web page from the server that stores that page (***web server***).

Once the web page is sent to you, the client-server interaction is complete. The server fulfills many brief requests from clients very rapidly.





## Requesting a Web Page

A web page is requested by the user by either:

- typing in a ***Universal Resource Locator*** (***URL***) into the web browser's address field OR
- clicking on a hyperlink in a document that contains a URL

A request for a URL has three parts:

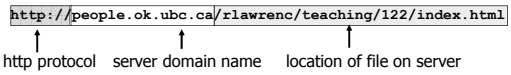
- Protocol: http:// - Hypertext Transfer Protocol
  - ⇨Tells the computer how to handle the file
- Server computer's domain name or IP address
- Page's path and file name:
  - ⇨Tells the server which file (page) is requested and where to find it.



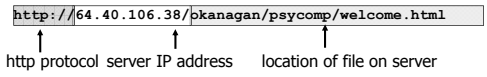


## URL Example

Here is an example URL:



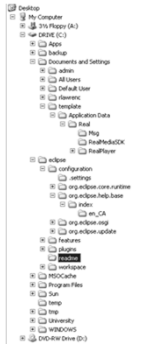

Another example with an IP address:








## File Structure



A ***file structure*** is a method for organizing files on a computer.

The common file structure is a ***hierarchy*** of directories or folders.

Each ***folder*** has a name and can contain any number of files or subdirectories.

Each ***file*** has a name.





## File Structure Usage

The file structure or file system is used when storing files locally on your computer.

It is also used as part of a URL as it provides the location on the server computer of the file that is requested by the client.

Example:

```
http://people.ok.ubc.ca/rlawrenc/teaching/122/index.html
```

- The directory path is: `rlawrenc/teaching/122/`
- The file with name: `index.html` is in the above directory.
- Note that directory names are separated by slashes "/".





## Hypertext

The HTTP protocol can transmit any file type, not just documents. However, it is most commonly used to transmit documents written in Hypertext Markup Language (HTML).

HTML describes the layout of a document including fonts, text style, image placement, and hypertext links.

Hypertext links provides a way to jump from point to point in documents (non-linear). Links may jump within a document, between documents on a server, and to documents on other servers.

We will see how to write HTML documents soon.

## TCP/IP (Transmission Control Protocol/ Internet Protocol)

TCP/IP (Transmission Control Protocol/Internet Protocol) is the structure (language) and protocol used for communication between computers on the Internet.

This is how TCP/IP works:

- Information is broken into a sequence of small fixed-size units called IP *packets*.
- Each packet has space for the unit of data, the source and destination IP addresses, and a sequence number.
- The packets are sent over the Internet one at a time using whatever route is available.
- Because each packet can take a different route, congestion and service interruptions do not delay transmissions.
- Receiver re-assembles packets using sequence numbers





## Moving Packets

How packets are transmitted between computers on the Internet is dependent on the medium of transmission.

The ***Internet backbone*** has the largest capacity (***bandwidth***) and is optical fiber.

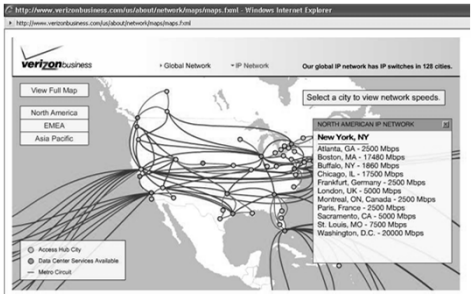
Smaller bandwidth connections may use copper fiber or cable to connect machines to hubs and routers.

End users may use phone lines, cable lines, or even fiber optic connections as their first ***hop*** (connection) to the Internet.





## Verizon Backbone

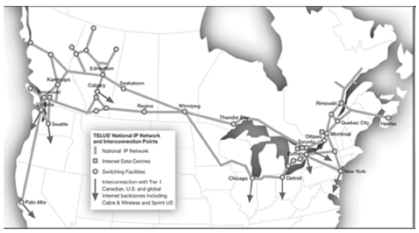



Source: http://www.verizonbusiness.com/us/about/network/





## Telus Network

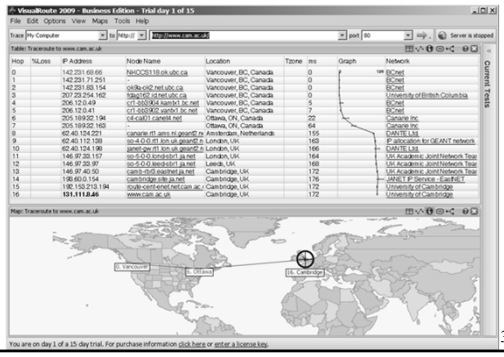



Source: http://about.telus.com/networktechnology/images/NationalNetwork.gif





## Trace Route Example








## Practice Question
## Sending a Message Using TCP/IP

Using two small pieces of paper. Write on each paper:

- The *destination row and seat #* of your receiver (IP address).
- Your *sender row and seat #*. (IP address)
- Row numbers start at 1 from the front of the class.
- Seat numbers start at 1 from the left and increase going right. Only count seats where people are sitting.
- The number 1 or 2 for the first or second piece of paper.
- The first part of a sentence on piece #1 and second part on piece #2.
- Example:

| **Src:** Row 4 Seat 5 **Dest:** Row 6 Seat 3 **Seq#:** 1 | **Src:** Row 4 Seat 5 **Dest:** Row 6 Seat 3 **Seq#:** 2 |
|---|---|
| This is how | TCP/IP works. |

## Practice Question
## Sending a Message Using TCP/IP (2)

Routing:

- The person at the left end of the row is your gateway to the Internet. They will be called the *gateway router*. All packets are sent to the gateway router even if the destination is in your row.
- When a gateway router receives a packet they will look at the destination address. If it is in their row, they will route it back down the row. If not, they will route it to an internal router.
- Several people are selected to be *internal routers*. They are responsible for some number of rows (e.g. 1 to 5). When they receive a message, and the destination is in their set of rows, they will give it to the gateway for that row. Otherwise, they will give it to the adjacent core router (up or down).
- When ready, send your message. (You may break it up).
- When you get a message, reassemble. On the back of the paper, send a reply back to the sender.





## TCP/IP

***Question:*** Put the following steps in order to describe transmitting information on the Internet using TCP/IP.

**1)** put sequence #, sender and destination IP addresses on packets

**2)** route packets through whatever route is available

**3)** re-assemble packets using sequence numbers

**4)** break data into fixed-sized packets

**a)** 1,2,3,4

**b)** 4,2,3,1

**c)** 4,1,2,3

**d)** 4,2,3,1





## Aside: What is DHCP?
## What are tiny URLs?

***Dynamic host configuration protocol*** or ***DHCP*** automatically assigns an IP address when a device is attached to the network.

- Your IP address will change on your device whenever you connect to a different access point.
- A static IP address for servers does not change.
- DHCP also assigns local DNS server addresses.

***Tiny URLs*** like http://tinyurl.com/cosc122 or http://bit.ly/pqTvJw used in Twitter and short messages are not full URLs. The first part of the URL http://tinyurl.com or http://bit.ly is a valid URL that goes to tinyurl.com or bit.ly. The last part "cosc122" or "pqTvJw" is a hash code that is used to look up the full URL that was originally entered and redirects to it.





## Aside: Where is the www?

Does a name have to contain `www` (e.g. `www.yahoo.com`)?

- If the server name contains `www`, then `www` would be the name of the computer in that domain.
- A computer can be a web server even if its name does not contain `www`.

Other Frequently Asked Questions (FAQs):

- 1) Do I have to type `www` or can I just type `yahoo.com`?
  - ⇨ You can normally just type `yahoo.com` without the `http` and `www`, and the browser will fill it in for you. Note that the site still is `www.yahoo.com`.
- 2) What is http?
  - ⇨ Http stands for hypertext transfer protocol. It is the name of the protocol for sending web pages over the Internet. It is the use of the http protocol that makes a computer a web server.
- 3) Are all computers on the Internet web servers? No.





## Aside: What is a protocol?

A ***protocol*** is a set of rules for transmitting information.

The underlying transmission protocol on the Internet is TCP/IP, but there are a variety of protocols for computer networks.

Many applications on the Internet have their own protocol built on top of TCP/IP to transfer information specific to them.

- We have already seen HTTP. Others include FTP and email.

In addition to the protocol, servers and clients need to know a ***port*** (or address of software on server) when multiple server applications may be running on the same machine.

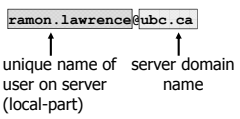
- The port for HTTP is usually 80.





## Aside: The anatomy of an e-mail address

An e-mail address has a structure similar to a URL:




FAQ:

- 1) Why the @ symbol?
  - ⇨ At "@" symbol was used as a separator because it is not in the user or server name and naturally implies that user is associated with the server.
  - ⇨ Developed by Ray Tomlinson in 1971.
- 2) Are e-mail addresses case-sensitive?
  - ⇨ Domain names are not case-sensitive. Local user names are case-sensitive, but most servers ignore case to avoid errors. Lower case characters are used for readability and by convention.

## Aside: What is a firewall?

A ***firewall*** is a network device that is installed on the edge of a network to prevent unauthorized network traffic from entering a local network.

- A firewall uses information in the packets (IP addresses, ports) to determine good traffic from bad traffic.
- Administrators can restrict access to certain sites or applications using a firewall.

A ***proxy server*** or ***gateway*** is a computer on the network that your computer must communicate through to get out onto the Internet.





## Other Types of Networks WAN and LAN

The Internet is a "network of networks" as it connects independent networks together using a common protocol.

A ***Wide Area Network*** (***WAN***) is designed to send information between widely separated locations.

A ***Local Area Network*** (***LAN***) connects computers close enough to be linked by a single cable or wire pair.

- Ethernet is the main technology for LAN.

Our campus has a WAN to connect all buildings together, and within each building is one or more LANs.





## LAN Overview

A local area network has a shared channel (wire, wire pair, or optical fiber) that connects a set of computers.

Each computer is connected to the channel, allowing it to send a signal that can be detected by all computers connected to the channel.

When you plug in a network cable to a wall socket, you are connecting your computer to the shared channel. Behind the walls are cables running to a central hub that connects all plug locations.





## Ethernet Protocol

The commonly used protocol for communicating between computers on a LAN is called ***Ethernet***.

Ethernet is a very simple, decentralized protocol. To send:

- A computer listens to the channel. If it's quiet, it's free.
- The computer starts sending on the channel.
- While sending, the computer listens to make sure it is the only one sending. If not, it stops for a random amount of time then starts again.

Try it: Select several people. Each tries to say "I am transmitting a message" at roughly the same time. Try until everyone has sent the message.





## Internet History

- **1960s:** Packet switching developed
- **1972:** ARPAnet had 15 nodes and a host-to-host protocol. First public demo. Ray Tomlinson at BBN wrote e-mail program.
- **1973:** Ethernet at Xerox PARC (Metcalfe)
- **1974:** Cerf and Kahn – TCP/IP (Turing Award)
- **1979:** ARPAnet has 200 nodes
- **1982:** SMTP e-mail protocol defined
- **1983:** DNS defined for name-to-IP-address translation
- **1990:** Internet has 100,000 nodes
- **1991:** World-Wide Web invented by Tim Berners-Lee of CERN.
- **1994:** Mosaic (later Netscape) developed by Marc Andreesen.
- **1995:** ARPANet decommissioned. Replaced backbone by commercial Internet service providers.





## Internet Current

- **1995-present:** Tremendous growth in the number of users and applications. Common applications:
  - Communications (text, voice, video) – email, Skype
  - News/advertising/entertainment
  - Massive Data Sets – Google Maps/Earth, scientific data, directories
  - Social sites – MySpace, Facebook, LinkedIn
  - E-commerce – Amazon, Walmart
  - Search – Google, Yahoo, Bing
  - Online gaming/social gaming
  - Mobile access and smart phones – iPhone, Android, Blackberry
- Improved network bandwidth and reduction in cost has made these applications possible. Standardization is also important.
- Bandwidth and hardware is now a commodity (*cloud computing*).
- Easier than ever to build an Internet application.

## Impact of the Internet on Society

- Nowhere is remote.
  - A person in Kelowna has the same access to Internet information as someone in Toronto.
- People are interconnected.
  - Can interact with people around the world.
- Social relationships are changing.
  - We are spending more time online and doing less in-person activities.
- English is becoming a universal language.
  - The influence of American culture since World War II has led to rapid adoption of English as the default language for global commerce, science and technology.
- Freedom of speech and assembly have expanded.
  - The Internet is ***technically*** unmediated allowing freedom of expression (both positive and negative). Anyone can publish at almost no cost.
  - Countries like China can restrict access to information on the Internet.





## What is The Value of Information on the Internet?

Since anyone can publish a web page with information (fact or fiction), this introduces several important issues:

- ***Information overload*** – too much information which makes it difficult to find relevant information
- ***Information quality*** – the lack of independent editing creates an issue of trustworthiness and completeness
- ***Information organization*** – how is information organized so that it can be easily found and used





## Measuring Information Quality

The quality of information can be measured in several ways:

- 1) Investigating the source – Trusted sources with an online presence should have quality information.
  - It is possible to look up the organization that publishes a web site using its domain name and the Whois facility.
  - Canada Whois: `http://whois.cira.ca/public`
- 2) Realistic site content – A site is more believable if it contains physical addresses, phone numbers and credentials, and if it appears current and professionally done.
- 3) Search engine ranking and external links – Although not fool proof, higher search engine rankings and links from other sites are an indication that others value the information on the site.





## Aside: How does a search engine work?

All popular search engines such as Google and Bing have two basic parts:

- ***Crawler***: Visits sites on the Internet, discovering Web pages and building an index to the Web's content.
  - A search engine has crawlers running continuous to refresh and update its index database of web pages.
  - When a crawler visits a page it identifies the terms on the page and then processes any outgoing links.
- ***Query processor***: Looks up user-submitted keywords in the index and reports back which Web pages the crawler has found containing those words.
  - The query processor does not search the Internet – it only returns answers previously found by the crawlers.
  - The ranking algorithm to identify important pages is critical to success of the search engine. Google uses the PageRank algorithm.





## Aside: How to Search Effectively

Search engines allow basic keyword search and more advanced search features. Some things you should know:

- Search for a phrase by putting double quotes around it.
  - e.g. “Computer Fluency” instead of Computer Fluency
- By default, there is a logical `AND` connecting terms. This means that all terms must appear in the document.
  - e.g. Computer Fluency means both Computer and Fluency must appear.
- You can also use `OR` to indicate either term is suitable:
  - e.g. (Book `OR` Magazine) - parenthesis are optional
- You can use `NOT` to indicate work should not appear:
  - e.g. `NOT` Fluency
- You can use plurals, but the search engine will normally discard them. (Called ***stemming***) E.g. trees becomes tree
- Question: What is a cached page in a search engine?





## Survey
## Essential Technology

***Question:*** What technology could you ***absolutely*** not live without?

**A)** television
**B)** cell phone
**C)** social network sites
**D)** email/text messaging/chat
**E)** none of the above

## *Discussion*
## *Effect of Internet and IT on Society*

In small groups, discuss what you think are the most important positive and negative effects of the Internet and computers on society.

- Are there issues that we have not covered?

Be prepared to give a short summary of your discussions.





## *Conclusion*

The Internet is an ***asynchronous***, ***point-to-point*** communication tool. However, due to its speed, ***broadcast*** and ***synchronous*** applications are also supported.

The three components of any communication are a shared medium, a common language, and an agreed upon protocol.

An ***IP address*** is a unique address that identifies a computer on the Internet. ***Domain names*** are used as they are easier to remember, and are mapped to IP addresses by a DNS server.

The ***World Wide Web*** allows for the storage, transmission, and display of information in documents called web pages.

The Internet and IT in general has made a significant impact on society and our daily lives.





## *Objectives*

- Compare and contrast: synchronous and asynchronous
- Compare and contrast: broadcast and point-to-point
- Identify what types of communication common devices use.
- List and define the 3 components of communication.
- Define: Internet
- Explain how you can get on the Internet.
- Explain the format and purpose of an IP address. IPv4 vs. IPv6.
- Describe the hierarchical structure of a domain name.
- Explain the purpose and role of a DNS server.
- Explain the key features of the TCP/IP protocol.
- Define: client, server
- Define: WWW, web page, web server, web browser
- List and explain the components of a URL.





## *Objectives (2)*

- Provide the unique feature of HTML documents compared to other documents.
- Define: file structure, file, directory
- Compare and contrast: WAN and LAN
- Provide an overview of the Ethernet protocol.
- List and discuss some of the impacts of the Internet and IT on society.
- List 3 challenges with the vast amounts of information available on the Internet.
- Discuss how you can evaluate the quality of information found online.
- List the two components of a search engine.

# COSC 122
# Computer Fluency

# HTML

**Dr. Ramon Lawrence**
**University of British Columbia Okanagan**
**ramon.lawrence@ubc.ca**



## Key Points

1) Hypertext Markup Language (HTML) is the standard language for building web pages.

2) HTML is our first example of a language for communicating instructions to the computer.





## Hypertext Markup Language (HTML)

***Hypertext Markup Language*** (***HTML***) is a language for describing how a web page appears in a web browser.

- HTML describes the layout of a document including fonts, text style, image placement, and hypertext links.

An HTML document looks like a regular text document except that it contains ***tags*** which are words or abbreviations enclosed in angle brackets: **<** and **>** .

- Each tag controls some appearance of the web page.
- In HTML 5, tags are not case-sensitive.
  - ⇨ We will use lower case as convention.
- Tags usually come in pairs such as:

```
<p>Hello world!</p>
```





## HTML "Hello World!“ Example

```
<!DOCTYPE html>

<html>
<head>
<title>Hello World using HTML</title>
</head>

<body>

<p>Hello world!</p>

</body>
</html>
```

Start of document

Beginning material such as title in head

Content of document goes in body

Text formatted using paragraph <P> tags.

End of document





## HTML "Hello World!“ Example (2)


HTML Source Code






## Basic Formatting Tags

To put text in a ***paragraph*** use the ** ** tags:

```
<p>This is text in a paragraph.</p>
```

To make text ***bold*** use the ** ** tags:

```
<b>This text is bold.</b>
```

To make text ***italic*** use the ** ** tags:

```
<i>This text is in italics.</i>
```

To identify important text use the ** ** tags:

```
<strong>This text is in strong format.</strong>
```

To emphasize text use the ** ** tags:

```
<em>This text is emphasized.</strong>
```

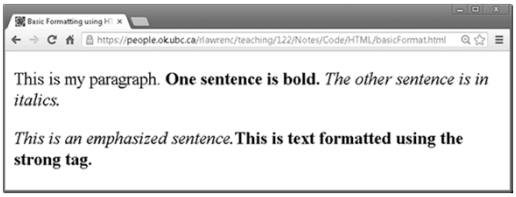
## HTML Formatting Example

```
<html>
<head>
<title>Basic Formatting using HTML</title>
</head>

<body>

<p>This is my paragraph.  <b>One sentence is bold.</b>
<i>The other sentence is in italics.</i></p>

<p><em>This is an emphasized sentence.</em><strong>This is text
formatted using the strong tag.</strong></p>

</body>
</html>
```





## HTML Formatting Example (2)








## More Formatting Tags

You can apply more than one formatting at a time:

```
<p><b><i>This text is in bold and italics.</i></b></p>
```

There are 6 levels of heading defined **<h1>**, **<h2>**, .. **<h6>**. Each heading creates a new line and displays in a large font.

```
<h1>Largest heading</h1>
<h2>Next largest heading</h2>
<h6>Lowest level heading</h6>
```

Use **<hr>** to put a horizontal line in the document.

Use **<br>** to put a line break.

- Note that these last two do not have a closing tag.





## Display vs. HTML Format

When displaying an HTML document, a web browser ignores white space. White space is considered spaces, tabs, and newlines. Multiple spaces and newlines are replaced with a single space when displayed (unless the **<pre></pre>** tags are used).

Since white space is ignored, it is advisable to make your HTML document easier to read and edit by inserting spaces and blank lines.

Remember the web browser uses the tags to determine how to display the document, not what it looks like in your editor!





## Special Symbols

Since the **<** and **>** are special (reserved) symbols in the HTML language, we need a way to use them in our documents.

The **&** (ampersand) is the escape symbol that tells HTML a special character is required. Terminate with a **;** (semi-colon).

Common characters:

```
&lt;          <
&gt;          >
&eactue;      é
&ntilde;      ñ
&amp;         &
         non-breaking space
```





## Text Alignment

You can left, right, or center a paragraph text by:

```
<p style="text-align:center">This text is centered.</p>
```

In general, ***attributes*** of tags are specified by providing their name and their value.

```
<p>Regular font.<br></p>
<p style="font-size:200%">Font twice as large!</p>
```

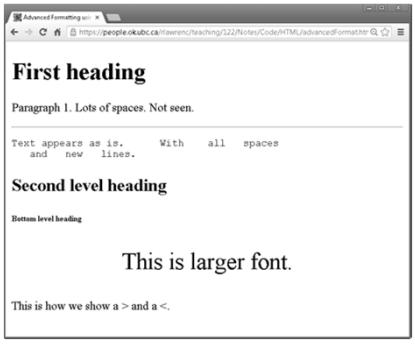
## HTML Advanced Formatting Example

```
<html>
<head><title>Advanced Formatting using HTML</title></head>
<body>
<h1>First heading</h1>
<p>Paragraph 1.     Lots of spaces.     Not seen.</p>
<hr>
<pre>
Text appears as is.      With    all   spaces
   and   new   lines.
</pre>
<h2>Second level heading</h2>

<h6>Bottom level heading</h6>
<p style="text-align:center;font-size:200%">This is larger
font.</p>
<p>This is how we show a &gt; and a &lt;.</p>
</body>
</html>
```





## HTML Advanced Formatting Example (2)

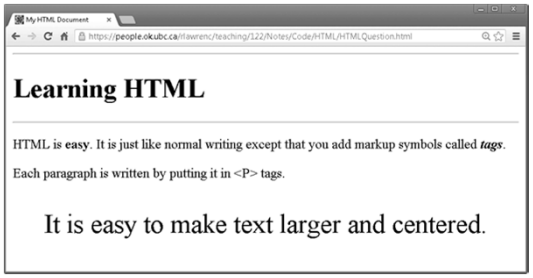







## HTML Practice Question

Write the HTML document that looks like this:








## General Syntax Rules: Comments

***Comments*** are used by the programmer to document and explain the code. Comments are ignored by the computer.

HTML comments use the syntax "**<!--**" for the start of the comment and "**-->**" for the end of the comment.

Example:

```
<!-- This is a HTML comment -->
<!-- This is a HTML comment
      that crosses
         multiple lines -->
```





## HTML Tags

***Question:*** Select ***one*** of the tags that do not have a matching closing tag.

**A)** `br`

**B)** `h1`

**C)** `hr`

**D)** `p`





## HTML Tags – No Formatting

***Question:*** Which one of these tags will display text "as-is" in your document with line breaks, spaces, etc.?

**A)** `p`

**B)** `hr`

**C)** `pre`

**D)** `b`

## HTML Tags – Reserved Symbols

***Question:*** Which one of these symbols is ***not*** reserved?

**A)** <

**B)** ;

**C)** >

**D)** &





## Marking Links With Anchor Tags

***Anchor tags*** are used to create hyperlinks in the document.

An anchor tag has two components:

- **Anchor text** – the text in the document that is highlighted
- **Hyperlink reference** – address of web page to link to

Example:

```
<a href="http://www.yahoo.com">Go to Yahoo!</a>
```






## Specifying A Hyperlink Location Absolute and Relative Paths

The location where the user goes to when clicking on the link may be given as a complete ***absolute*** URL:

```
<a href="http://www.yahoo.com">Go to Yahoo!</a>
```

or ***relative*** to the current location:

```
<a href="mydir/helloWorld.html">Go to Hello World in mydir</a>
```

Use an absolute URL when the page is on a different server.

Use a relative path when the page is on the same machine. The path depends on the current page location.

- Use ".." to navigate to the directory above your current location.

Analogy: If you give someone directions to the Science building, those directions will depend on where you start from!

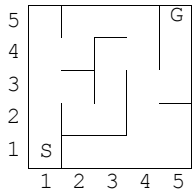




## Absolute and Relative Paths Question

Given this maze, specify the location of the goal (G) both in absolute terms and relative based on start location (S).

- Describe the path how you want.



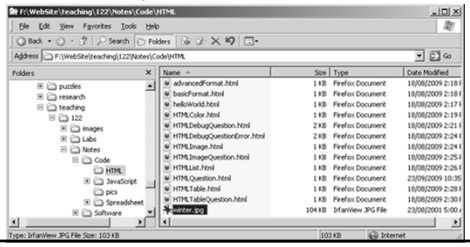





## Absolute and Relative Paths Question 2

Given these files, specify the location of the file `winter.jpg`:

- absolutely
- relative to directory `122` (directory 122 is the start point)
- relative to directory `pics`







## Including Pictures With Image Tags

Images can be shown in a document with the image **<img>** tag:

```
<img src="imgs/winter.jpg"/>
<img src="http://www.google.ca/intl/en_ca/images/logo.gif"/>
```

Notes:

- `src` stands for source and is either an absolute or relative path to a picture file.
- A picture may be in many different formats. Common formats:
  - ⇨GIF: Graphic Interchange Format
  - ⇨JPEG: Joint Photographic Experts Group
  - ⇨PNG: Portable Network Graphics
- The file extension (.gif, .jpg, .png) tells the browser which format the image is stored in.

## Positioning the Image in the Document

By default, images are inserted in the page at the point where the tag is specified in the HTML, and the text lines up with the bottom of the image.

The align attribute can line up image with the top of the line of text or the bottom.

Align left or right attribute puts the image on the side of the browser window and the text flows around it.

To put image on separate line, enclose within paragraph tags.





## Advanced: Images and Links Together

You can create a hyperlink on an image so when the user clicks on the image, they go to the desired location.

Example:

```
<a href="http://www.google.ca">
<img src="http://www.google.ca/intl/en_ca/images/logo.gif"/>
</a>
```

- This example shows an image retrieved from Google's web site and will go to the web site when the image is clicked.
- Note that we could have sent the user to any site, not just the Google site where the image came from.





## HTML Image and Link Example

```
<html>
<head><title>Images and Links in HTML</title></head>
<body>

<p>The image is placed <img src="../../../images/redball.gif"/>
in the text.</p>

<p>This is <a href="http://www.google.ca">
<img src="http://www.google.ca/intl/en_ca/images/logo.gif"/>
Google's Image</a>.</p>

<p><img align="left" width="100" height="100" src="winter.jpg"/>
We have wrapped some text around this wonderful winter scene and
resized it so that it is smaller than its original form.</p>

</body>
</html>
```





## HTML Image and Link Example (2)






## Handling Color

Color is used for both background and text. A color is specified either by name (red, yellow, orange) or by *hexadecimal* RGB color numbers.

- `background-color` defines the background color.
- `color` is text color.

Examples:

```
<body style="background-color:silver;color:yellow">
<p style="color:red">Red font</p>
<p style="color:#FF8E2A">Orange font</p>
```

To set link color (put this before the body tag):

```
<style>
a:link { color: orange }
a:visited { color: green }
a:active  { color: orange }
</style>
```





## Advanced: Specifying Color by Number

When a color is specified by a hexadecimal number it consists of three numbers from 0 to 255 representing the intensity of red, green, and blue respectively.

- However, instead of using decimal numbers (base 10), hexadecimal numbers are used (base 16) which have the digits 0 to 9 plus A (10), B (11), C (12), D (13), E (14), and F (15).

Examples:

| Color | RGB Intensity | Hexadecimal |
|---|---|---|
| Black | (0, 0, 0) | #000000 |
| White | (255, 255, 255) | #FFFFFF |
| Red | (255, 0 , 0) | #FF0000 |
| Green | (0, 255, 0) | #00FF00 |
| Blue | (0, 0, 255) | #0000FF |
| Orange | (255, 142, 42) | #FF8E2A |
| Purple | (147, 112, 219) | #9370DB |
| Yellow | (255, 255, 0) | #FFFF00 |

## HTML Image Practice Question


- The background is black. Image and link locations:
  - greenball.gif – relative path 4 directories up then in `images` directory
  - CBC logo – `http://www.cbc.ca/logo.gif`
  - helloWorld link – `helloWorld.html` is in current directory
  - CBC link – `http://www.cbc.ca`





## HTML Anchor Tag

***Question:*** Which one of these anchor tags is correct?

**a)** `<a ref="Click Here">http://www.yahoo.com</a>`

**b)** `<a href="index.html">My Home Page</a>`

**c)** `<a src="http://www.yahoo.ca">Yahoo</a>`

**d)** `<a ref="http://www.google.ca" src="My Page">`





## HTML Image Tag

***Question:*** Which one of these image tags is correct?

**a)** `<img "picture.gif"/>`

**b)** `<img>picture.gif</img>`

**c)** `<img ref="picture.gif"/>`

**d)** `<img src="picture.gif"/>`





## Lists in HTML

It is possible to create both bulleted and numbered lists.
For a bulleted list, use the tags **<ul></ul>** and **<li></li>**:

```
<ul>
   <li>Item 1</li>
   <li>Item 2</li>
   <li>Item 3</li>
</ul>
```

For a numbered list, use the tags **<ol></ol>** and **<li></li>**:

```
<ol>
   <li>Item 1</li>
   <li>Item 2</li>
   <li>Item 3</li>
</ol>
```





## Advanced Lists

You can nest lists inside each other to produce sublists:

```
<ul>
  <li>Item 1
    <ol>
       <li>Subitem 1.1</li>
       <li>Subitem 1.2</li>
    </ol>
  </li>
</ul>
```

Another type of list is the ***definitional list***:

- **<dl>** and **</dl>** tags begin and end the list
- **<dt>** and **</dt>** surround the terms to be defined
- **<dd>** and **</dd>** surround the definitions

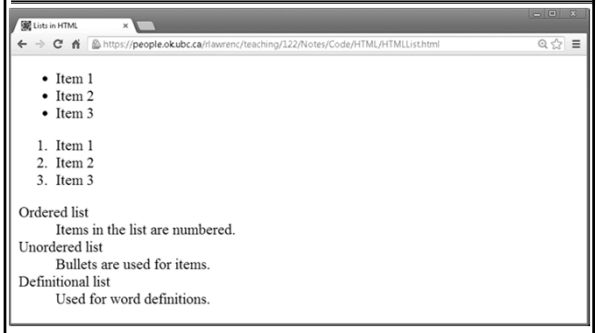




## HTML List Example

```
<html><head><title>Lists in HTML</title></head>
<body>
<ul>
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
</ul>
<ol>
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
</ol>
<dl>
  <dt>Ordered list</dt><dd>Items in the list are numbered.</dd>
  <dt>Unordered list</dt><dd>Bullets are used for items.</dd>
  <dt>Definitional list</dt><dd>Used for word definitions.</dd>
</dl>
</body>
</html>
```

## HTML List Example (2)








## Tables in HTML

Data can be displayed in tables using the **<table></table>** tags. Rows are enclosed in table row **<tr></tr>** tags, and each cell is denoted using table data **<td></td>** tags.

A table may have a caption centered at the top of the table by using the **<caption></caption>** tags.

A header row may be created using the **<th></th>** tags.





## HTML Table Example

```
<html><head><title>Tables in HTML</title></head><body>
<table border="1" style="background-color:orange">
  <caption>This is my table caption.</caption>
  <tr><th>Tag</th><th>Purpose</th></tr>
  <tr><td>TABLE</td><td>Start and end entire table</td></tr>
  <tr><td>TR</td><td>Start and end each row</td></tr>
  <tr><td>TH</td><td>Special column header formatting</td></tr>
  <tr><td>td</td><td>Start and end each data cell</td></tr>
  <tr><td colspan="2">COLSPAN allows a cell to span multiple
columns.</td></tr>
   <tr><td rowspan="2">ROWSPAN</td><td>ROWSPAN allows a column
to span multiple rows.</td></tr>
   <tr><td>This is a ROWSPAN example.</td></tr>
   <tr><td>Tables can be nested.</td><td>
      <table border="2" style="background-color:yellow">
          <tr><td>Nested table</td></tr>
          <tr><td>Row 2</td></tr></table>
   </td></tr>
</table></body></html>
```

Table with border

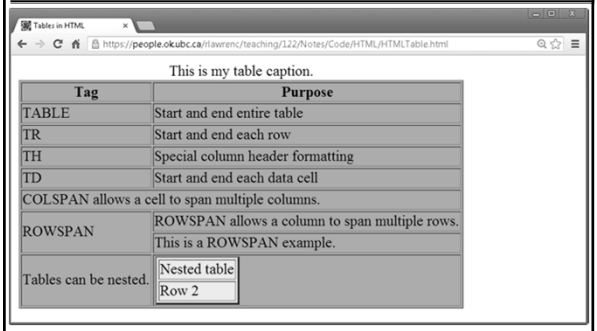
Header row

Nested table





## HTML Table Example (2)








## Where can I use HTML?

HTML is everywhere on the Internet: UBC Connect, Facebook, Yahoo, Google, etc.

All your most popular websites have HTML as a base (and maybe other languages on top of them.)

You can see the HTML source for any page in a browser by selecting `View->Page Source` in Firefox/Chrome or `View->Source` in Internet Explorer.

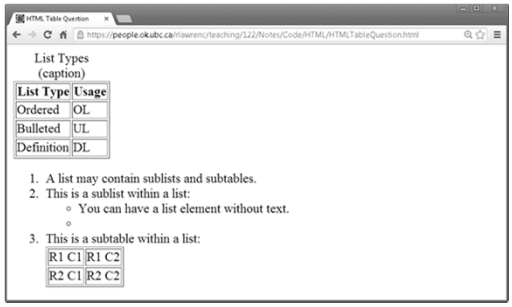
The idea of ***markup*** in HTML is common to many environments and languages.





## HTML Practice Question

Write the HTML document that looks like this:



## Advanced: Style Attribute

The `style` attribute can be added to a tag to control its appearance. Different settings are separated by semi-colons.

Example:

```
<body style = "background-color:black; color: white;">
```

Some common style settings:

- background-color e.g. background-color:yellow
- font-family e.g. font-family:"Times New Roman",Serif;
- font-style e.g. font-style:italic
- font-size e.g. font-size:100%
- color e.g. color:red;
- text-align e.g. text-align:center





## Advanced: Cascading Style Sheets

Cascading style sheets (CSS) is a language for controlling the appearance of web pages, especially color, layout, and fonts.

How it works:

- In a CSS source, you define the markup tag and its formatting.
- When that tag is used in your HTML page, the formatting is automatically applied. This makes changes easier!

Example:

```
<html><head><title>Using CSS</title></head>
<style type="text/css">
  body { font-family: "Times New Roman"; color: purple;}
  h1 { font-family: Helvetica; color: green;}
</style>
<body>
<h1>Formatted heading</h1>
<p>Regular text</p>
</body></html>
```





## Advanced: Three Types of Selectors

**By element** - Apply to all instances of a particular element:

```
h1 { font-family: Helvetica; color: red;}
```

Use:

```
<h1>This will be red</h1>
```

**By id** - Apply to all content with a specific id:

```
#section {text-align: left; background-color: blue;}
```

Use:

```
<div id="section"><h1>Heading</h1><p>Text..</p></div>
```

**By class** - Apply to specified instances of any tag:

```
h1 { color: green;}
h1.red { color: red;}
```

Use:

```
<h1>This will be in green</h1>  <h1 class="red">Red</h1>
```





## Conclusion

***Hypertext Markup Language*** (***HTML***) is a language for describing how a web page should appear in a web browser.

We have seen how we can use markup tags of HTML to:

- change fonts and formatting
- add images and hyperlinks
- create lists and tables

HTML is our first example of a language to instruct the computer on what to do.





## Objectives

- Define: HTML, tag

Remember the HTML syntax for:

- Formatting tags: `<p>`, `<i>`, `<b>`, `<h1>` to `<h6>`, `< pre>`
- Image and link tags: `<a>` and `<img>`
- Changing colors and text alignment
- Lists and tables

Be able to create HTML pages that have a given appearance.

Be able to draw what a HTML document will look like when displayed in a web browser.





## HTML List Tag

***Question:*** Which one of these HTML code fragments will produce a list like below?

1. Item 1
2. Item 2

**a)**
```
<ol>
  <li>Item 1</li>
  <li>Item 2</li>
</OL>
```

**b)**
```
<ul>
  <li>Item 1</li>
  <li>Item 2</li>
</ul>
```

**c)**
```
<ol>
  <il>Item 1</il>
  <il>Item 2</il>
</ol>
```

**d)**
```
<ul>
  <ol>Item 1</ol>
  <ol>Item 2</ol>
</ul>
```

## *HTML Table Tag*

***Question:*** Which one of these HTML code fragments will produce a table like this?

| Heading 1 | Heading 2 |
|---|---|
| Val1 | Val2 |

**a)**
```
<table border="1">
<tr><th>Heading 1</th><th>Heading 2</th></tr>
<tr><td>Val1</td><td>Val2</td></tr>
</table>
```

**b)**
```
<table border="2">
<tr><td>Heading 1</td><td>Heading 2</td></tr>
<tr><dt>Val1</dt><dt>Val2</dt></tr>
</table>
```

# COSC 122
# Computer Fluency

## Debugging

**Dr. Ramon Lawrence**
**University of British Columbia Okanagan**
**ramon.lawrence@ubc.ca**



## Key Points

1) Debugging is the act of finding and correcting errors in a system.

2) All users need to know the general debugging steps due to the complexity of computer systems.

3) A common reason for computer errors is our lack of precision in specifying instructions to the computer.





## Computers are Dumb… so We Must be Precise

Computers have no knowledge or intelligence unless they are programmed with it.

When talking with people, we assume knowledge and the ability to reason out errors or missing details when communicating.

Computers hate imprecision and cannot handle it by default.

- Programmers often write applications to detect simple, common imprecise statements and fix them (but not always).





## Entering Data into Forms

Data is typically entered into a computer using a form.

A programmer can restrict the types and number of symbols that can go into a form field.

- e.g. only allow numbers in a phone number field

Many errors occur when users either enter data that does not follow these restrictions, or they enter incorrect data that is accepted by the computer because it is not properly checked.

Question: Have you ever entered false data into a form?





## Debugging: What's the Problem?

***Debugging*** is the process of determining why a system does not work properly.

We perform debugging all the time in daily life, usually to fix problems with other systems and tools we interact with (cars, lights, appliances, electronics, our own bodies, etc.).

Debugging is a little different with computers and information technology because *usually* it is not a component failure that is the source of the problem. More commonly, it is our interaction and limited understanding of how the computer works.





## Whose Problem Is It?

When we debug an information system, we are always part of the problem!

- We give the commands and the input, so the only other possible cause is a broken system.

People do not *knowingly* make errors, but we frequently do if we do not understand how to use a system properly.

- We must be precise and know what the computer expects.

Debugging is challenging as a computer user because:

- the computer cannot debug itself
- we cannot debug it directly either because the error is internal to the computer

Debugging involves *working with* the computer to try and understand what is happening and why.

## Debugging: Solving a Mystery

Debugging is very similar to solving a mystery.

To discover and solve the problem we ask questions like:

- Do I need more clues?
- Are my clues reliable?
- What is a theory to explain the problem?
- How can I test if my theory is correct?

Like solving mysteries, the only way to get good at debugging is practice and gaining experience about common problems and solutions.





## The Four Key Steps in Debugging

1) Check that the error is reproducible.
- Computers are deterministic. Make sure you know exactly how to reproduce the error.

2) Do not jump to conclusions.
- The actual cause of the error may be many steps removed from the visible symptoms.

3) Check all the "obvious" sources of error.
- You would be surprised how often a cable is not plugged in…

4) Isolate the problem
- The goal is to make good assumptions and divisions of parts that you know are working and others that need investigation.
- Be careful! It is often parts (including yourself) that you assume are working that really are not.
  - ⇨Make sure assumptions are backed up by tests.





## Debugging HTML Web pages

How to debug HTML web pages according to the 4 steps:

- 1) Reproduce errors - This is easy. Every time you reload or refresh the page, you should see the same errors.
- 2) Do not jump to conclusions - Although there are bugs in web browsers, it is vastly more likely that the HTML document contains errors. Focus your attention there.
- 3) Obvious sources of errors - One "obvious" source of errors is non-matching open and closed tags. As you gain experience, more errors become obvious.
- 4) Isolate the problem - An HTML document is processed starting at the beginning, so try to fix errors at the start of the document first then work down.





## Common HTML Errors

Some common HTML errors:

- Open with no matching close tag
  - ⇨`<a href="test.html">...`
- Non-matching quotes
  - ⇨`<img src="myimage.gif/>` (no closing quote)
  - ⇨`<img src="myimage.gif'/>` (open with ", close with ')
  - ⇨`<img src="myimage.gif"/>` (HTML does not like smart quotes)
- Missing attribute or incorrect attribute name.
- Incorrect tag name (which may result in non-matching tags).
- Incorrect file name or hyperlink address.
- Forgot required tags like `<html>`, `<head>`, `<body>`.
- Forget to stop escape sequence with a semi-colon
  - ⇨e.g. `&lt` (missing semi-colon should be `&lt;`)





## Aside: The Cost of Debugging

When developing a computer system or application, the process of testing and debugging is extremely costly.

Most software requires 40% of the total time, cost, and effort to debug and fix problems in the system.
- Even so, many errors go unnoticed until the system is used.

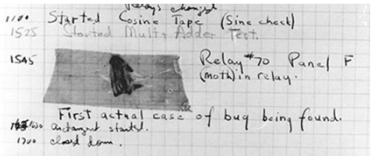
To make software development more efficient and less costly, ***software engineering*** principles and techniques are followed.
- Although building software is harder than building a bridge due to its complexity, software engineers continually strive to make software development better.





## Aside: The First Bug

The first "bug" in a computer system was actually a moth found in the Harvard Mark II computer system in 1947 by Rear Admiral Grace Hopper.




Source: U.S. Naval Historical Center Online Library Photograph NH 96566-KN

## Debugging Follows the Scientific Method

Determining what a program does and finding any errors follows the scientific method.

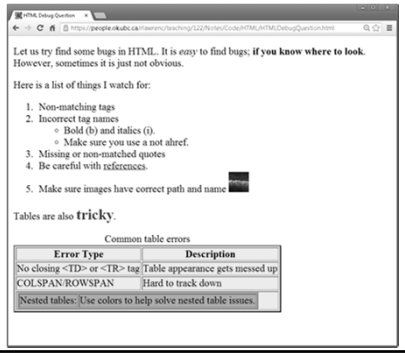
1) **Model** – create a hypothesis on what the program does
2) **Predict** - for inputs not yet tried / simulated
3) **Experiment** – run the program to check your prediction
4) **Refine** – modify your hypothesis based on experimental results and repeat.

Understanding software is in many ways similar to understanding how complex real-world processes work.





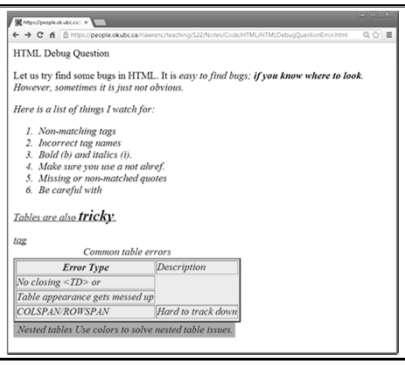
## HTML Debugging Question Desired Output








## HTML Debugging Question Actual Output








## HTML Debugging Question HTML Document

```
<html>
<head><tile>HTML Debug Question</title></head>
<p>Let us try find some bugs in HTML.  It is <i>easy to find
bugs; <b>if you know where to look</b>.  However, sometimes it is
just not obvious.</p>

Here is a list of things I watch for:
<ol>
<li>Non-matching tags</li>
<li>Incorrect tag names
       <li>Bold (B) and italics (I).</li>
       <li>Make sure you use A not AHREF.</li>
    </ul>
<li>Missing or non-matched quotes</li>
<li>Be careful with <a href="HelloWorld.html'>references</a>.
</li>
<li>Make sure images have correct path and name
       <img src="winter.jpeg'' width="32" height=32></li>
</ol>
```





## HTML Debugging Question HTML Document (2)

```
<p>Tables are also <b style="font-size:150%;
color:redd">tricky</b>.</p>

<table border=2 style="background-color:yellow">
 <caption>Common table errors</caption>
 <tr><th>Error Type</th><td>Description</td></tr>
 <tr><td>No closing &ltTD&gt; or <TR> tag</td><td>Table
appearance gets messed up</td></tr>
 <tr><td>COLSPAN/ROWSPAN</td><td>Hard to track down</td></tr>
 <tr><table style="background-color:orange">
     <tr><td colspan=2><td>Nested tables</td>
     <td>Use colors to help solve nested table issues.</td></tr>
     </table></td></tr></table>
</table>

</body>
</html>
```





## Conclusion

***Debugging*** is a systematic approach to discover and fix errors in a system.

- Debugging a computer system requires working with the computer to diagnose the problem with the realization that we are often the cause of the problem.

The four key steps of debugging are:

- 1) Check that the error is reproducible.
- 2) Make sure you know what the problem is.
- 3) Check all the "obvious" sources of error.
- 4) Isolate the problem

As users, we can resolve many errors with a little practice, experience, and patience without requiring help from IT service technicians.

## *Objectives*

- Give some examples of imprecise communication.
- Explain why precision is important for a computer.
- Define: debugging
- List and explain the 4 key steps of debugging.
- List (and remember) some common HTML errors.

Be prepared to debug HTML documents both on the computer and on paper.

# COSC 122
# Computer Fluency

## Information Representation

**Dr. Ramon Lawrence**
**University of British Columbia Okanagan**
**ramon.lawrence@ubc.ca**



## Survey
## Reading the Notes

***Question: HONESTLY,*** how often do you read the notes before class?

**A)** never
**B)** up to 25% of the time
**C)** up to 50% of the time
**D)** all the time
**E)** This class has notes?





## Survey
## Class Still Easy?

***Question: HONESTLY,*** rate the course difficulty so far from 1 (easy) to 5 (difficult).

**A)** easy
**B)** below normal
**C)** normal
**D)** above normal
**E)** difficult





## Key Points

1) Representing data digitally means to represent it using discrete units.

2) The lowest level of data representation on a computer is a single bit that represents either 0 or 1.

3) Bits are combined to allow more information to be represented including characters and numbers.

4) More complex information like documents, spreadsheets, and databases (all of which we will see later) are simply compositions and higher-level abstractions of bits.





## Everything is digital - Is that good?

Almost all of our music, movies, data, and pictures are digital.
- Most people believe digital is better. What does digital mean?

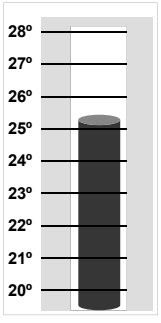
Representing something ***digitally*** means to store the data in discrete units. A unit is ***discrete*** if it is distinct or separate from other units. The smallest unit of data depends on what we are representing.

Digital differs from ***analog*** where the information is encoded on a continuous signal (spectrum of values).
- Note that sound and images are analog by nature.





## Analog versus Digital
## Thermometer Example



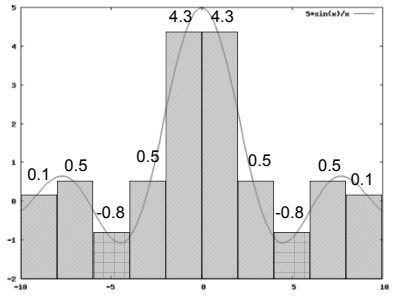

A thermometer contains mercury which expands and contracts in response to temperature changes.

The mercury level is analog, and its expansion continuous over the temperature range.

By adding marks and units to the thermometer, we are digitizing the information.

## Conversion from Analog to Digital

How would you digitize this analog data into 10 discrete points?








## Why are electronics digital?

1) Computers are digital and many home electronics are interfacing with computers.

2) Analog signals are more susceptible to noise that degrades the quality of the signal (sound, picture, etc.). The effect of noise also makes it difficult to preserve the quality of analog signals across long distances.

3) Reading data stored in analog format is susceptible to data loss and noise. Copying analog data leads to declining quality.





## Digitizing Discrete Information Phone Numbers

A simple example of digital data is a phone number. A phone number consists of multiple units of information called digits (the numbers 0 through 9).

Although numbers are used to represent the values of different digits, it is possible to use any collection of 10 distinct symbols to represent the 10 possible different values.

However, using numbers is nice because they have a natural ordering (0 < 1 < 2 < 3 < … < 9).





## Digitizing Discrete Information Phone Numbers

| Standard Phone Keys | Phone Keys (with shift) | Phone Keys as Musical Symbols |
|---|---|---|
| 1 2 3 | ! @ # | ▶ ▼ ◀ |
| 4 5 6 | $ % ^ | ▶▶ ‖ ◀◀ |
| 7 8 9 | & * ( | ♫ ▲ ♯ |
| 0 | ) | ■ |

Question: Represent the phone number 254-123-6789 using both alternative digitization methods.





## Digital Phone Numbers

***Question:*** Using the symbol encoding for phone numbers, what is this number: `$#% - **()`

**A)** 615 - 8809

**B)** 435 - 8800

**C)** 453 - 8899

**D)** 435 - 8890

| | | |
|---|---|---|
| 1 2 3 | ! @ # | |
| 4 5 6 | $ % ^ | |
| 7 8 9 | & * ( | |
| 0 | ) | |





## Encoding Information with Dice

We will see how much information we can encode using six-sided dice.

Quick question: If a dice has six unique sides, how many different values/states can it encode?

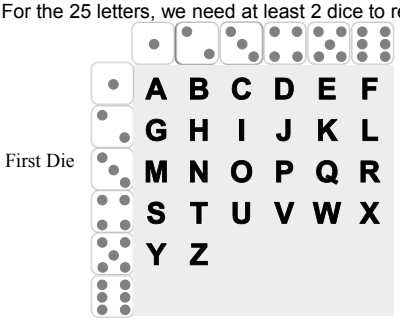
Answer: 6

By using more dice, we can encode more data:

1 die = 6 states
2 dice = 6 x 6 = 36 states
3 dice = 6 x 6 x 6 = 216 states
N dice = $6^N$ states

## Encoding Information with Dice (2)

For the 25 letters, we need at least 2 dice to represent a symbol:




Question: Spell your name using our dice representation.





## Encoding Information with Dice (3)

The extra 10 states could be used to encode numbers. However, what if we need to encode other symbols as well?

One solution is to use 3 dice per symbol which gives us 216 possible symbols.

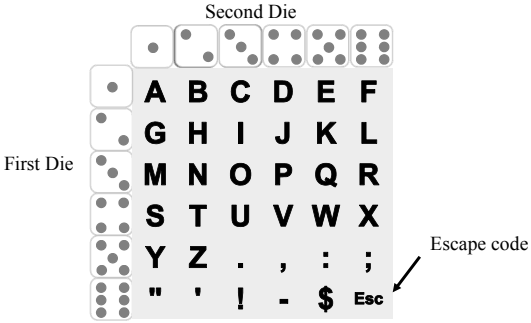
Another way is to have one special symbol be an escape character. It does not match any legal character, so it will never be needed for normal text digitization. An escape character indicates that the digitization is "escaping from the basic representation" and applying a secondary representation.

Question: What escape character have we already seen and in what context?





## Encoding Information with Dice (4)

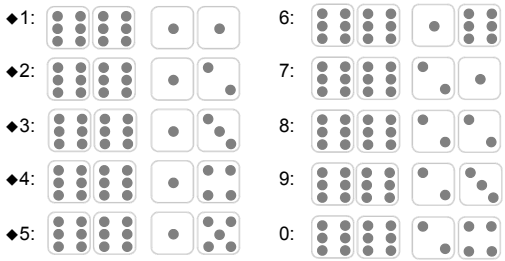







## Encoding Information with Dice (6)

We will use the escape and two dice to represent numbers:




Question: How would we encode the number 198 in this notation?





## Representing Data using Dice

***Question:*** Using the dice encoding, what is this:

**A)** SIMYR

**B)** DICER

**C)** DICYR

**D)** SNMYO






## Aside: The Time versus Space Tradeoff

A fundamental challenge in computer science is encoding information efficiently both in terms of space and time.

- We just saw an example where we could save space (only need 2 dice instead of 3) by using the escape symbol.

At all granularities (sizes) of data representation, we want to use as little space (memory) as possible. However, saving space often makes it harder to figure out what the data means (think of compression or abbreviations). In computer terms, the data takes longer to process.

The ***time versus space tradeoff*** implies that we can often get a faster execution time if we use more memory (space). Thus, we must strive for a balance between time and space.

## Representing Binary Data

***Data*** is information before it has been given any context, structure and meaning.

Binary data has two states and is represented in a computer using a bit. A ***bit*** can either be 0 or 1.

- The word bit is short for "binary digit".

A computer memory consists of billions of bits which allows for an almost limitless number of possible states.





## What is a Byte?

A ***byte*** is a sequence of 8 bits.

Historical note: Byte is spelled with a "y" because engineers at IBM were looking for a word for a quantity of memory between a bit and a ***word*** (usually 32 bits). Bite seemed appropriate, but they changed the "i" to a "y", to minimize typing errors.





## Converting Binary to Decimal

To convert a binary number *B* to a decimal number *D*:

Let *B* have *n* bits of the form $b_{n-1}b_{n-2}\ldots b_3b_2b_1b_0$ then

$D = b_{n-1}*2^{n-1} + b_{n-2}*2^{n-2} + \ldots + b_3*2^3 + b_2*2^2 + b_1*2^1 + b_0*2^0$

Base 10 (decimal) example:

- $765 = 7 * 10^2 + 6 * 10^1 + 5 * 10^0$

Example: binary value is 10010111

- $= 1 * 2^7 + 0 * 2^6 + 0 * 2^5 + 1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 1 * 2^1 + 1 * 2^0$
- = 151

Question:

1) Compute the decimal value of 1011.

2) Compute the decimal value of 00101010.





## Converting Decimal to Binary

To convert a decimal number *D* to a binary number *B*:

- Repeat until *D* = 0
  - ⇨IF D is odd THEN append a 1 bit to the front of B
  - ⇨ELSE IF D is even THEN append a 0 bit to the front of B
  - ⇨Set D equal to D / 2

Example: Decimal value of D = 19

- 19 is odd — B = 1
- 9 is odd — B = 11
- 4 is even — B = 011
- 2 is even — B = 0011
- 1 is odd — B = 10011

Question: Compute the binary value of 115.





## Aside: Adding Binary Numbers

Just like regular addition, we can add binary numbers. The rules are the same:

- Work from right to left, adding corresponding digits in each place position.
- If adding the two digits is bigger than the maximum digit value (9 in base 10 and 1 in base 2), we carry to the next position.

Example:

```
          1 1      (carries)
  1 0 0 1 0 1 1 1
+ 0 1 1 0 0 1 1 0
-----------------
  1 1 1 1 1 1 0 1
```





## Hex Explained

Previously we specified custom colors in HTML using hex digits

- e.g., <font color ="#FF8E2A">
- *Hex* is short for hexadecimal (base 16)

We use hex as it is easier than writing sequences of bits. Each hex digit corresponds to a 4-bit sequence.

- e.g. 1011 (binary) = 11 (decimal) = B (hexadecimal)

Question:

Convert this binary sequence to hexadecimal:

0000 0101 1000 0001 1111 1110

## Decimal to Binary to Hex Conversion Table

| Decimal | Binary | Hexadecimal |
|---|---|---|
| 0 | 0000 | 0 |
| 1 | 0001 | 1 |
| 2 | 0010 | 2 |
| 3 | 0011 | 3 |
| 4 | 0100 | 4 |
| 5 | 0101 | 5 |
| 6 | 0110 | 6 |
| 7 | 0111 | 7 |
| 8 | 1000 | 8 |
| 9 | 1001 | 9 |
| 10 | 1010 | **A** |
| 11 | 1011 | **B** |
| 12 | 1100 | **C** |
| 13 | 1101 | **D** |
| 14 | 1110 | **E** |
| 15 | 1111 | **F** |





## Review Binary to Decimal

***Question:*** Convert this binary number to decimal: **01001111**.

**A)** 143

**B)** 78

**C)** 79

**D)** 47





## Review Decimal to Binary

***Question:*** Convert this decimal number to binary: **123**.

**A)** 1011011

**B)** 1111011

**C)** 11111011

**D)** 1110011





## Review Binary to Hexadecimal

***Question:*** Convert this binary number to hexadecimal:

**0111 1000 1111 1110 1001**

**A)** 78ACD

**B)** 58FED

**C)** 78FE9

**D)** 78FFD





## Review Questions Decimal to Binary to Hexidecimal

1) Convert 163 (decimal) to binary and hexadecimal.

2) Covert 10101010 to decimal and hexadecimal.

3) Convert EF (hexadecimal) to binary and decimal.





## Representing Characters using Bits

In total, there are 95 basic character symbols which would require 7 bits to encode.

- 26 uppercase and 26 lowercase Roman letters, 10 Arabic numerals, 10 arithmetic characters, 20 punctuation characters, and 3 non-printable characters (tab, backspace, new line).

The standard 7-bit code for characters is called ***ASCII*** *(American Standard Code for Information Interchange).*

- Later, the ASCII code was extended (*extended ASCII*) to 8 bits to handle additional characters.

Just like the dice encoding, each 8-bit sequence maps to a particular character. We use an ASCII table to determine what each bit sequence means.

## ASCII Table

| First 4 bits (most significant) | Next 4 bits (least significant) 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0000 | ␀ | ␁ | ␂ | ␃ | ␄ | ␅ | ␆ | ␇ | ␈ | ␉ | ␊ | ␋ | ␌ | ␍ | ␎ | ␏ |
| 1 | 0001 | ␐ | ␑ | ␒ | ␓ | ␔ | ␕ | ␖ | ␗ | ␘ | ␙ | ␚ | ␛ | ␜ | ␝ | ␞ | ␟ |
| 2 | 0010 | | ! | " | # | $ | % | & | ' | ( | ) | * | + | , | - | . | / |
| 3 | 0011 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | : | ; | < | = | > | ? |
| 4 | 0100 | @ | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
| 5 | 0101 | P | Q | R | S | T | U | V | W | X | Y | Z | [ | \ | ] | ^ | _ |
| 6 | 0110 | ` | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o |
| 7 | 0111 | p | q | r | s | t | u | v | w | x | y | z | { | \| | } | ~ | ␡ |
| 8 | 1000 | ␀ | ␁ | ␂ | ␃ | ␄ | ␅ | ␆ | ␇ | ␈ | ␉ | ␊ | ␋ | ␌ | ␍ | ␎ | ␏ |
| 9 | 1001 | ␐ | ␑ | ␒ | ␓ | ␔ | ␕ | ␖ | ␗ | ␘ | ␙ | ␚ | ␛ | ␜ | ␝ | ␞ | ␟ |
| A | 1010 | ␠ | ¡ | ¢ | £ | ¤ | ¥ | ¦ | § | ¨ | © | ª | « | ¬ | - | ® | ¯ |
| B | 1011 | ° | ± | ² | ³ | ´ | µ | ¶ | · | ¸ | ¹ | º | » | ¼ | ½ | ¾ | ¿ |
| C | 1100 | À | Á | Â | Ã | Ä | Å | Æ | Ç | È | É | Ê | Ë | Ì | Í | Î | Ï |
| D | 1101 | Ð | Ñ | Ò | Ó | Ô | Õ | Ö | × | Ø | Ù | Ú | Û | Ü | Ý | Þ | ß |
| E | 1110 | à | á | â | ã | ä | å | æ | ç | è | é | ê | ë | ì | í | î | ï |
| F | 1111 | ð | ñ | ò | ó | ô | õ | ö | ÷ | ø | ù | ú | û | ü | ý | þ | ÿ |

Question: Represent the phone #: 254-123-6789 using ASCII.





## Representing Text Beyond ASCII - Unicode

Although ASCII is suitable for English text, many world languages, including Chinese, require a larger number of symbols to represent their basic alphabet.

The ***Unicode standard*** uses patterns of 16-bits (2 bytes) to represent the major symbols used in all languages.

- First 256 characters exactly the same as ASCII.
- Maximum # of symbols: 65,536.



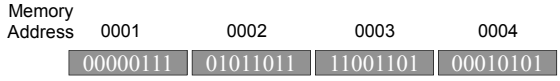


## Representing Data in Memory Integers

A integer is a whole number. It is encoded in a computer using a fix sized number of bits (usually 32).

- The first bit is a sign bit (0=positive, 1=negative).
- Negative numbers are represented in *two's complement notation*. The "largest" bit pattern FFFFFFFF is -1.

Example: 123,456,789 as a 32-bit integer:








## Representing Data in Memory Doubles and Floats

A number with a decimal may be either stored as a *double* or *float* value. On 32-bit machines, a double is usually 8 bytes long.

- ⇨A float is normally half the size of a double value and has less precision.

Double values are stored using a ***mantissa*** and an ***exponent***:

- Represent numbers in scientific format: $N = m * 2^e$
  - ⇨m - mantissa, e - exponent, 2 - radix
  - ⇨Note that converting from base 10 to base 2 is not always precise, since real numbers cannot be represented precisely in a fixed number of bits.
- There are many standards for representing numbers in a fixed number of bits. The most common is **IEEE 754 Format**:
  - ⇨32 bits - 1-bit sign; 8-bit exponent; 23-bit mantissa
  - ⇨64 bits - 1-bit sign; 11-bit exponent; 52-bit mantissa

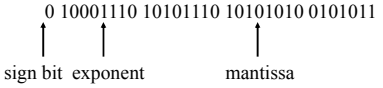




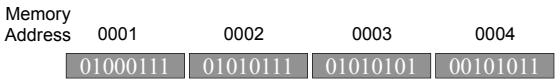
## Representing Data in Memory Doubles (2)

The number 55,125.17 stored as 4 consecutive bytes is:

- Hexadecimal value is: 4757552B Stored value is: 55125.168




- Divided into bytes looks like this:








## Aside: Can you really get rich by stealing fractions of a penny?

Have you ever seen a movie (e.g. Office Space) where the plot was to steal fractions of a penny lost due to rounding?

Can that really happen?

- Called ***salami slicing*** as stealing money in very small quantities by always rounding down fractions of a penny.

Consider the salary in the previous example: $55,125.17 that had an actual value of 55,125.16**8** where stored in the computer.

- That imprecision can be serious when we are talking about millions of numbers and operations.
- Idea: Round ***down*** to 55,125.16 and take the extra penny

Good code would not store monetary values as doubles because they are imprecise or make sure to round appropriately.

## Representing Data in Memory Strings from Characters

A ***string*** is a sequence of characters allocated in consecutive memory bytes.

The first character of the string is at the first location of memory. The last character can be known by either:

- ***Null-terminated string*** - last byte value is 0 to indicate end of string.
- ***Byte-length string*** - length of string in bytes is specified (usually in the first few bytes before string starts).





## Representing Data in Memory Dates

A ***date*** value can be represented in multiple ways:

- Integer representation - number of days past since a given date
  - ⇨Example: # days since Jan 1, 1900
- String representation - represent a date's components (year, month, day) as individual characters of a string
  - ⇨Example: YYYYMMDD or YYYYDDD
  - ⇨Please do not reinvent Y2K by using YYMMDD!!

A ***time*** value can also be represented in similar ways:

- Integer representation - number of seconds since a given time
  - ⇨Example: # of seconds since midnight
- String representation - hours, minutes, seconds, fractions
  - ⇨Example: HHMMSSFF





## Encoding Higher-Level Information

We have seen how we can encode characters, numbers, and strings using only sequences of bits (and translation tables).

The documents, music, and videos that we commonly use are much more complex. However, the principle is exactly the same. We use sequences of bits and ***interpret*** them based on the ***context*** to represent information.

As we learn more about representing information, always remember that everything is stored as bits, it is by interpreting the context that we have information.





## Encoding an HTML Document

Here is our first HTML document:

```
<HTML><HEAD><TITLE>Hello World using HTML</TITLE></HEAD>
<BODY>
<P>Hello world!</P>
</BODY></HTML>
```

Here is its hexadecimal encoding:

```
3C 48 54 4D 4C 3E 3C 48 45 41 44 3E 3C 54 49 54 4C 45 3E
48 65 6C 6C 6F 20 57 6F 72 6C 64 20 75 73 69 6E 67 20 48
54 4D 4C 3C 2F 54 49 54 4C 45 3E 3C 2F 48 45 41 44 3E 0A
3C 42 4F 44 59 3E 0A 3C 50 3E 48 65 6C 6C 6F 20 77 6F 72
6C 64 21 3C 2F 50 3E 0A 3C 2F 42 4F 44 59 3E 3C 2F 48 54
4D 4C 3E
```

Some key hex digits: 3C = "<" 3E = ">" 20 = space 2F = "/" 0A = new line





## Encoding Higher-Level Information (2)

Note that the tag instructions to HTML are encoded in ASCII characters just like the text of the document. However, when the web browser processes the document they are treated as the special instructions that they are.

What we have is ***layers of abstraction*** or context to the bit sequence:

- Raw data – sequence of bits (or hexadecimal digits)
- Character level – Each 8 bit sequence represents a character encoded using ASCII.
- Document level – The document consists of text and tags. Tags are instructions to tell the browser how to display the document.

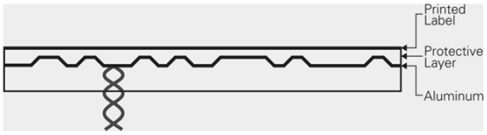




## Aside: Encoding Data on CDs and DVDs

How the present and absent states of bits are encoded depends on the medium on which the information is stored.

A CD consists of several different material layers. In one of those layers, indentations (or ***pits***) are created. Areas between pits are called ***lands***. The transition between a pit to a land represents 1 and no change represents 0.




- DVDs store more information as they have smaller pit sizes and more tracks (smaller distance between tracks).

## Aside: How do CD-R and CD-RW work?

The medium for encoding is different for CD-R and CD-RW.

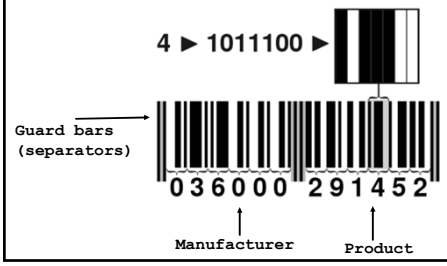
- CD-R/DVD-R – use *photosensitive dye* and are initially "blank". The write-laser of a CD writer changes the color of the dye at desired locations to make the CD appear to have pits and lands.
  - ⇨ Note that the dye will fade over time causing read errors.
- CD-RW/DVD-RW – are re-recordable by using a *metallic alloy* that has its reflectivity changed by the heat of the write laser.
  - ⇨ There is not as great a difference in lands and pits with CD-RW, hence they sometimes are not readable by all players.





## UPC Barcodes

**Universal Product Codes** (UPC) encode manufacturer on left side and product on right side. Each digit uses 7 bits with different bit combinations for each side (can tell if upside down).








## QR Codes

A **QR** (**Q**uick **R**esponse) code is a 2D optical encoding developed in 1994 by Toyota with support for error correction.

Hello World!

First Page of Syllabus

Make your own codes at: www.qrstuff.com.





## NATO Broadcast Alphabet

The code for broadcast communication is purposefully inefficient, to be distinctive when spoken amid noise.

| | | | | | |
|---|---|---|---|---|---|
| A | Alpha | J | Juliet | S | Sierra |
| B | Bravo | K | Kilo | T | Tango |
| C | Charlie | L | Lima | U | Uniform |
| D | Delta | M | Mike | V | Victor |
| E | Echo | N | November | W | Whiskey |
| F | Foxtrot | O | Oscar | X | X-ray |
| G | Golf | P | Papa | Y | Yankee |
| H | Hotel | Q | Quebec | Z | Zulu |
| I | India | R | Romeo | | |

Question: Pick a partner. Pretend to be a pilot and broadcast your name to your partner using the NATO broadcast alphabet.





## Conclusion

The ability to ***represent information*** is fundamental to the functions of a computer system.

There are multiple ways to represent information, the most basic of which is the presence and absence of information. A bit, which has the values 0 or 1, are used in computers.

Sequences of bits are combined to represent characters, numbers, and other data items. Larger data items are produced by combining these basic units.

Bits are just data until the necessary context is provided. There may be multiple levels of context (***abstraction***) needed to understand the meaning of a bit sequence.





## Objectives

- Compare and contrast: digital versus analog
- Give one reason why electronics are increasing digital.
- Explain how we can encode states and characters using dice.
- Explain the usefulness of the escape symbol.
- Define: data, bit, byte, word
- Convert from decimal to binary and binary to decimal.
- Convert from binary to hexadecimal and hexadecimal to binary.
- Explain why ASCII table is required for character encoding.
- Convert characters to binary using ASCII table.
- Briefly explain how integers, doubles, and strings are encoded.
- Encode using the NATO broadcast alphabet.
- Explain why context and interpretation produces information from data.

# COSC 122
# Computer Fluency

## Computer Organization

**Dr. Ramon Lawrence**
**University of British Columbia Okanagan**
**ramon.lawrence@ubc.ca**



## Key Points

1) The standard computer (von Neumann) architecture consists of a central processing unit (CPU) and memory that stores both instructions and data.

2) Understand the Fetch/Execute cycle performed by the CPU.





## What Computers Can and Cannot Do

Computers can only deterministically perform or execute instructions to process information. The computer must have instructions to follow.

A computer has no imagination, intuition, or emotions. It has no intelligence, free will, or its own purpose.

We must specify precisely what the computer should do in the form of instructions and the associated data. We will see how a computer processes our instructions.

A computer is useful not because it is "smart" but because it can do simple operations very quickly.

- A CPU at 1 GHz can perform about 1 billion operations/second.





## Central Processing Unit

A computer contains a ***Central Processing Unit*** (***CPU***) with several components:

- ***control unit*** – responsible for fetching, decoding, and executing instructions
- ***arithmetic/logic unit*** (***ALU***) – responsible for performing mathematical functions (addition, subtraction, etc.) and logical operations (AND, OR, NOT)
- ***registers*** – specialized memory locations used during execution

The CPU is connected to input/output devices and main memory using the motherboard ***system bus*** (set of wires).

This architecture is referred to as a ***von Neumann architecture*** because it uses memory to hold both instructions and data.

- The term came from the work of John von Neumann in 1945.



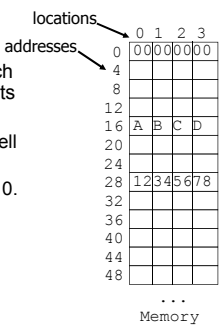


## Memory

***Memory*** stores programs and data.

Memory is divided into ***locations***. Each location has an ***address*** (usually 32-bits long) and can store 1 byte.

- We will show memory with each big cell consisting of 4 bytes (one ***word***).
- Addresses are in decimal and start at 0.




Questions:
1) What byte value is at address 0?
2) What byte value is at address 17?
3) What word value is at address 16?
4) What word value is at address 28?



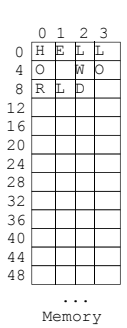


## Aside: Storing Characters in Memory

Recall that an ASCII character occupies one byte of space.

In the memory is the string HELLO WORLD stored at address 0.

- Remember that the computer stores things in binary. To the computer, this looks like: 48 45 4C 4C 4F 20 57 4F 52 4C 44 00.
  - ⇨Note that A = 65 = 41 (hex), B = 66 = 42 (hex), etc. Also, we terminated the string with NULL (00).




Questions:
1) Store the string "COMPUTER" in memory starting at address 20. End string with 00.
2) Store your name as a string starting at 16.

## Computer Instructions

The CPU has hardwired only a very few basic operations or instructions that it can perform:

- read a memory location into a register
- write a register value to a memory location
- add, subtract, multiply, divide values stored in registers
- shift bits left or right in a register
- test if a bit is zero or non-zero and jump to new set of instructions based on the outcome
- sense signals from input/output devices

All programs are composed of these basic operations.





## The Fetch/Execute Cycle

The computer performs the following cycle of operations to process instructions:

- Instruction Fetch (IF) - retrieve instruction from memory
- Instruction Decode (ID) - lookup meaning of instruction
- Data Fetch (DF) - fetch data for instruction
- Instruction Execution (EX) - execute instruction
- Result Return (RR) - return result to register

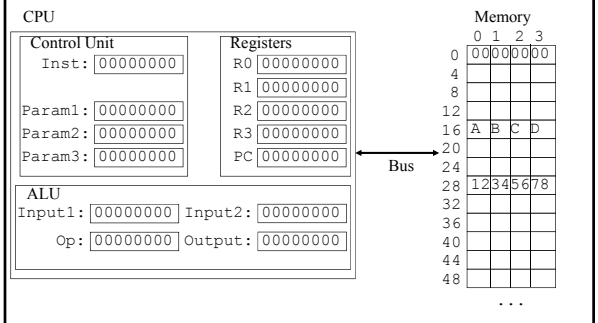
A special register called the ***program counter*** (PC) stores the address of the next instruction to execute.

- Since each instructor is 4 bytes long, the PC is incremented by 4 every time an instruction is executed unless a *branch* is performed.





## CPU and Memory Diagram








## Decoding Computer Instructions

How the instructions are encoded in bits depends on the processor and computer architecture. Just like with the ASCII lookup table, each bit sequence represents some instruction.

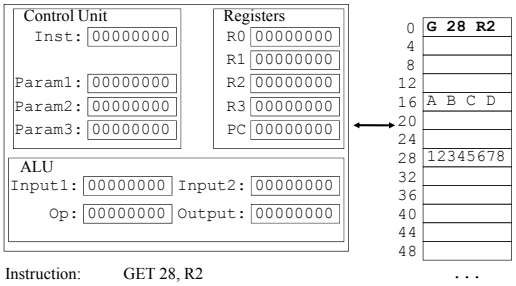
We will encode instructions using simple character strings.

- Each instruction has one, two, or three *parameters* or *operands*.
- G [address] [register] – Get data from memory at address and put in to the register
  - ⇨e.g. G 16 R1 – Get address 16 and put in register 1
- P [address] [register] – Put data in register to memory at address
  - ⇨e.g. P 20 R2 – Put data in register 2 into memory address 20
- + [reg1] [reg2] [reg3] – Store in reg3 result of reg1+ reg2
- - [reg1] [reg2] [reg3] – Store in reg3 result of reg1- reg2
  - ⇨e.g. + R0 R1 R2 – Store in register 2 result of register 0 + register 1





## Example: Executing Move Instruction

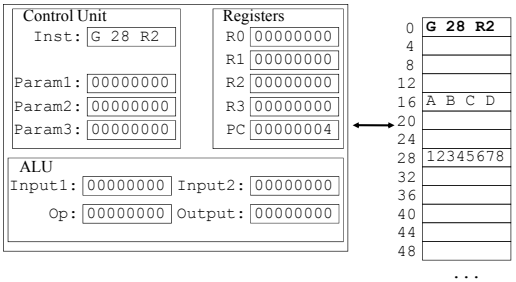



Instruction: GET 28, R2



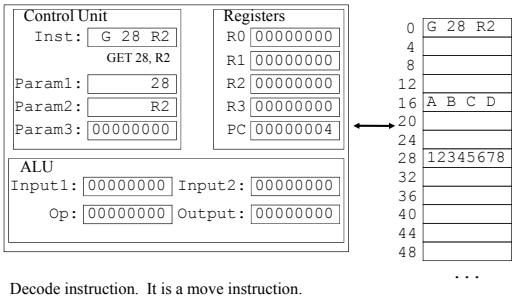


## Example: Executing Move Instruction - Fetch




Program counter is for address 0. Fetch from memory. Increment program counter by 4.

## Example: Executing Move Instruction - Decode

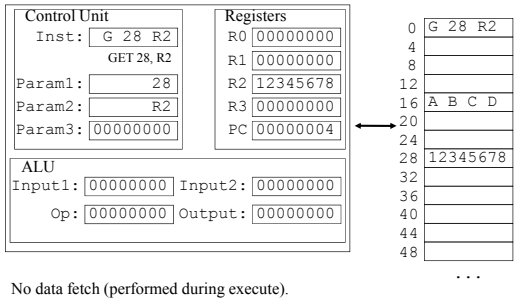



Decode instruction. It is a move instruction.
Set param1 to be 28 for memory address and param2 to be register 2.





## Example: Executing Move Instruction – Execute

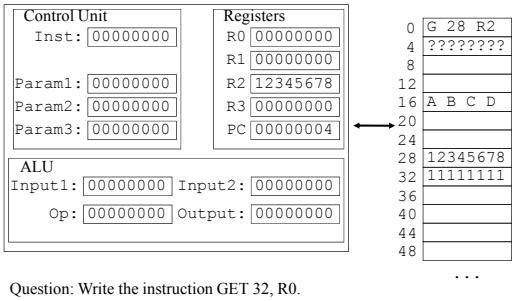



No data fetch (performed during execute).
Instruction execution: Fetch memory location 28 and put in R2.
No result return.





## Question: Instruction Execution

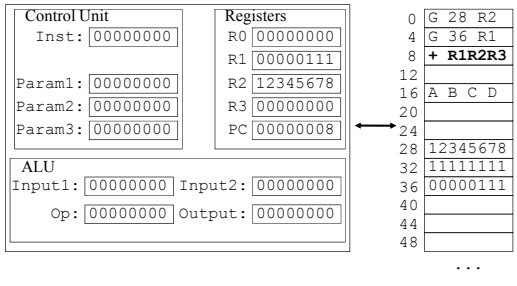



Question: Write the instruction GET 32, R0.
Put this instruction in location 4 and explain how it gets executed.





## Example: Add Instruction Execution

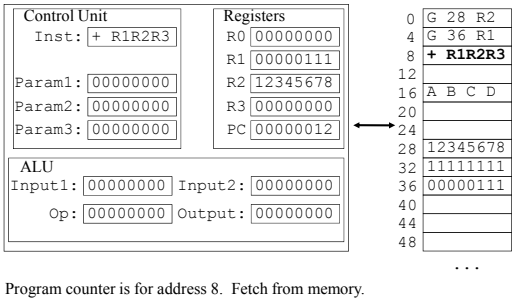



Instruction: ADD R1, R2, R3





## Example: Add Instruction Execution - Fetch

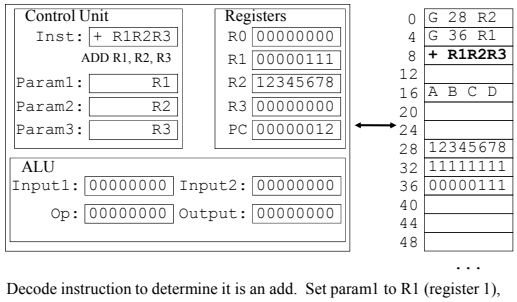



Program counter is for address 8. Fetch from memory.
Increment program counter by 4.



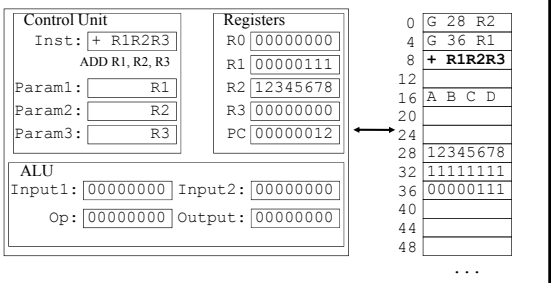


## Example: Add Instruction Execution - Decode




Decode instruction to determine it is an add. Set param1 to R1 (register 1), param2 to R2 (register 2), and param3 to R3 (register 3).
Prepare ALU to receive command and inputs.

## Example: Add Instruction Execution – Data Fetch

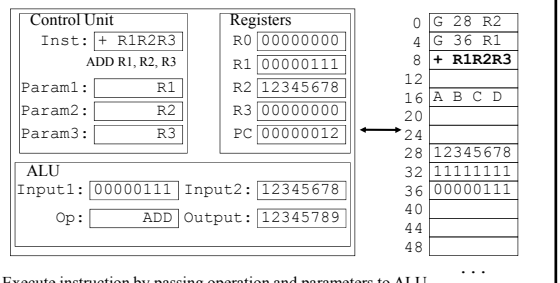



No data must be fetched from memory. Nothing to do in this step.





## Example: Add Instruction Execution – Execute

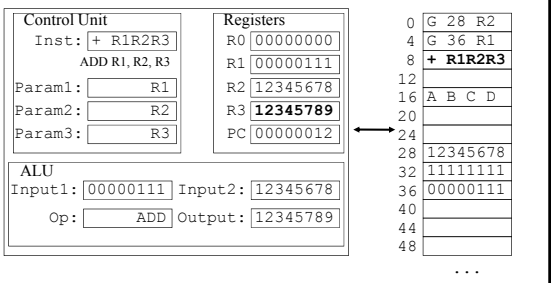



Execute instruction by passing operation and parameters to ALU.
Assume ALU knows operation is an ADD.
ALU executes the add (which may take some time) and result is in output.





## Example: Add Instruction Execution – Return

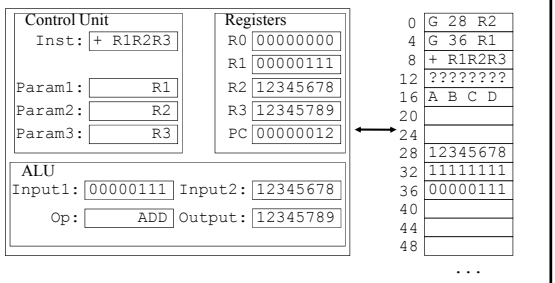



Output result from ALU is returned into register 3 as required.





## Question: Instruction Execution




Question: Encode the instruction to put the data in register 3 into memory address 40. Instruction goes in address 12.
Explain how this instruction gets executed.





## CPU

***Question:*** Which of these is ***NOT*** a component of the CPU?

**A)** control unit

**B)** arithmetic logic unit

**C)** bus

**D)** registers





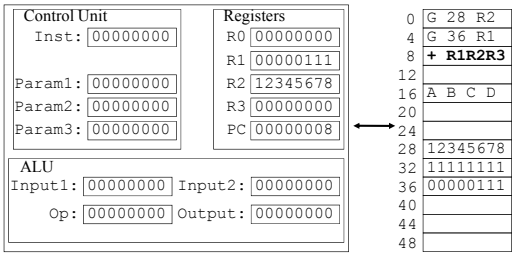
## Fetch/Execute Cycle

***Question:*** Put the steps in order for the Fetch/Execute cycle:

**1)** Data Fetch (DF)
**2)** Result Return (RR)
**3)** Instruction Execution (IE)
**4)** Instruction Fetch (IF)
**5)** Instruction Decode (ID)

**a)** IF,ID,DF,IE,RR
**b)** ID,IF,DF,RR,IE
**c)** IF,IE,ID,DF,RR
**d)** IF,DF,IE,ID,RR

## How many fetch/execute cycle steps?



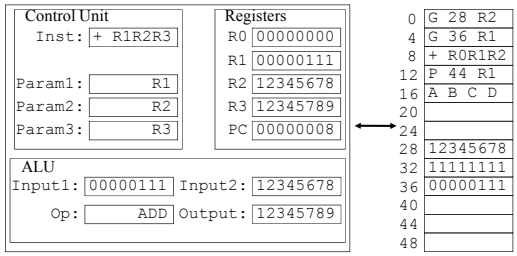

How ***many*** of the 5 fetch/execute steps are performed when executing the statement at memory address 8?

**a)** 1 **b)** 2 **c)** 3 **d)** 4 **e)** 5





## Instruction Execution Result




Question: What is the value of R2 after executing the statement at location 8?

**A)** 0 **B)** 111 **C)** 12345678 **D)** 12345789 **E)** None of the above





## Challenge Question: Writing a Simple Program

Write a simple program that computes the following:

- result = (A + B) * C
- Assume A is at location 52, B is at 56, C is at 60.
- Let A=5, B=2, C=10 then the result should be 70.
- Store result at location 64.
- Your program instructions should begin at address 0.

Be prepared to explain how the program works when executed.

**HINT:** You will need 6 instructions (3 `GET`, 1 `ADD`, 1 `MULTIPLY`, 1 `PUT`). Use the "*" to denote a multiply expression.





## Assembly and Machine Programming

The previous examples are similar to assembly programming.

***Machine programming*** involves specifying commands directly in binary form. ***Assembly language*** is a slightly higher level of commands which look more like English commands (MOVE, ADD) that are then translated to machine language before execution.

Most programmers do not write code in assembly or machine language because it is too low-level and time-consuming.





## Higher-Level Programming

Higher-level programming languages (such as HTML and JavaScript) are more powerful and easier to use because they have more powerful features and functions.

- The programmer does not have to specify all the details at a low-level and can use more general commands.
- Note that this is another form of *abstraction*.

Every language for communicating instructions to the computer must ultimately be translated to machine language for execution.

- The tools that translate to machine language are called ***compilers***. Compilers verify that code has correct syntax before performing the translation.





## Branch and Jump Instructions

One type of instruction that is available in all languages is called a ***branch*** or ***jump instruction***.

A branch instruction allows the program to execute different parts of code depending on certain conditions. Example:

```
IF hungry THEN
   eat something
ELSE
   go work
```

A branch instruction is implemented by making a decision whether or not to branch (usually a comparison) then setting the program counter to the address of the next instruction.

## Computer Speed

The speed that a computer can execute a program depends on many things:

- the speed of the CPU
- the speed of the bus, memory, and other devices
- the type of program and its characteristics
- the amount of parallelism and pipelining in the CPU

Historical example:

Apollo Guidance Computer had 2.048 MHz processor, 32KB of RAM, 4KB of ROM, and 8 16-bit registers.





## Computer Speed in GHz

The most basic measurement is the speed of the CPU clock because it is a rough estimate of the number of instructions that can be executed per second.

CPU speed is measured in hertz or cycles per second. The clock of typical CPUs perform billions (giga-) cycles per second, so the measurement is in giga-hertz (GHz).

- A computer with a 2 GHz CPU has the potential for executing 2 billion instructions per second.

Note that measuring computer performance simply on clock speed has been used as a marketing tool. As computers have become faster and more complex, CPU clock speed in GHz is not the best measurement.





## Aside: Advanced Processor Issues

Our explanation of how a processor works is a high-level abstraction of how they work in practice.

Processors may have multiple dedicated hardware, complex pipelining features, cache memory, and other optimizations.

Some other terminology:

- ***dual/quad core*** – means that there are two/four processing units on the same chip. The units may share subcomponents.
- ***dual processor*** – means that there are two separate processing units on different chips. Each processor appears distinct to the operating system.
- ***32-bit or 64-bit*** – describes the size of the basic memory unit and is also related to the bus size.





## Operating Systems

An operating system is software written to perform the basic operations that are necessary for the effective use of the computer that are not built into the hardware.

Three most widely used Operating Systems:

- Microsoft Windows
- Apple's Mac OS X
- Linux/Unix

The operating system performs booting, memory management, device management, Internet connection, file management, and provides a platform for the execution and development of programs.





## Computers and Electricity

Computer components consist of gates and circuits that control the flow of electricity.

A ***gate*** is a device that performs a basic operation on electrical signals.

- Common gates: `AND`, `OR`, `NOT`, `XOR`

A ***circuit*** is a combination of gates that performs a more complicated task.





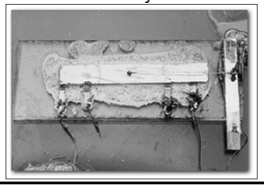
## Constructing Gates using Transistors

A ***transistor*** may either conduct or block flow of electricity based on input voltage (functions like a switch).

- Made of ***semiconductor*** material such as silicon.

An ***integrated circuit*** is contains both transistors and wires that connect them. manufactured during same process.

- Invented by Jack Kilby and others at Texas Instruments in 1958. They received the Nobel Prize in Physics in 2000.
- First integrated circuit:



## Summary: Putting it All Together

- An application is written by a programmer to solve a task using a programming language.
- The application uses features of the operating system to perform certain functions.
- The program is translated (compiled) into machine language for the computer to use. This form is simply a sequence of bytes.
- The byte sequence (binary file) is read from the hard drive into memory by the operating system when executed.
- The commands are executed using the fetch/execute cycle.
- The commands are implemented in hardware on silicon on integrated circuits that are produced using photolithography.
- The CPU contains a control unit and arithmetic logic unit that performs the basic operations. By controlling the flow of electricity, different states and operations are performed.





## Conclusion

The standard computer (von Neumann) architecture consists of a CPU, memory, a bus, and input/output devices.

The five basic steps of the ***fetch/execute cycle*** are:

- Instruction Fetch
- Instruction Decode
- Data Fetch
- Instruction Execution
- Result Return

Hardware commands are encoded on integrated circuits using gates that consist of transistors etched on silicon (semiconductor).





## Objectives

- Describe the von Neumann architecture (computer anatomy). Draw the diagram, and list and explain its main components.
- Explain the organization of memory in terms of locations and addresses.
- Define and list examples of: input/output device, peripheral
- List and explain the three major components of the CPU.
- Advanced: Explain the key feature of the von Neumann architecture.
- List some of the basic CPU instructions.
- List and explain the five steps of the fetch/execute cycle.
- Explain the purpose of the program counter register.
- Advanced: Explain how instruction decoding works and be able to decode an instruction using our format.





## Objectives (2)

- Be able to explain and demonstrate the fetch/execute cycle for a small program.
- Define: machine language, assembly language
- Explain the difference between a high-level programming language and assembly/machine language.
- Define: compiler
- Define: branch instruction
- List some factors in determining a computer's speed.
- Define: gate, circuit, integrated circuit, transistor, semiconductor

# COSC 122
# Computer Fluency

## Algorithmic Thinking

**Dr. Ramon Lawrence**
**University of British Columbia Okanagan**
**ramon.lawrence@ubc.ca**



## Key Points

1) There are five essential properties for algorithms.

2) The five basic steps of development are a general approach for solving problems using a computer.





## Algorithm

An ***algorithm*** is a precise, systematic method for producing a specified result.

We use algorithms all the time to complete tasks.

A common example is following assembly directions or using a recipe. Simpler examples include how to perform arithmetic or look up a person's name in a list.

Some algorithms are so simple or ingrained that we do not consciously remember the steps. However, precision is required when communicating the algorithm to others.





## Five Essential Properties of Algorithms

1) ***Inputs specified*** – must specify the type, amount, and form of data to be used during the algorithm
2) ***Outputs specified*** – must describe the result of the algorithm (it is possible to have no output).
3) ***Precision*** – specify precisely the sequence of steps to be performed including how to handle errors.
4) ***Reasonable Operations*** - The operations are doable.
5) ***Finite*** – The algorithm must eventually stop (terminate).





## Five Essential Properties of Algorithms

***Question:*** The algorithm on the shampoo bottle says: "Apply shampoo. Lather. Rinse. Repeat." Which one of the five essential properties does this algorithm not meet?

**A)** inputs specified

**B)** outputs specified

**C)** precision

**D)** reasonable operations

**E)** finiteness





## Group Discussions

Provide an algorithm for brushing your teeth.

## Specifying Algorithms using Language

An algorithm must be written using a language understood by both the writer of the algorithm and the reader who will use it.

For computer algorithms, the writer is a human programmer, and the reader is the computer. Natural languages like English are easy for humans, but are ambiguous and often require domain knowledge and context. Instead, we use precise programming languages (e.g. HTML/JavaScript).

A common barrier for students with programming is that the language is unfamiliar and that the computer requires precision. Remember, have patience!

- Learning a computer language is similar to learning a foreign language like Spanish.





## Divide-and-Conquer Grouping Instructions

The idea of ***abstraction*** is so important in programming that considerable effort is made to group similar instructions together so that they can be re-used.

Once a set of instructions is well-defined and tested, it is put in its own component that can be re-used to solve other problems.

***Grouping instructions*** allows programmers to use algorithms to solve other problems without re-creating them. It also allows programmers to focus on one task at a time (divide-and-conquer) and build up solutions to larger problems.





## Algorithm Performance

There is a whole area of computer science studying the performance of algorithms.

The goal is to find the algorithms that solve the problems in the least amount of time and use the least amount of memory.

Algorithms are usually compared based on the number of operations they perform or the amount of space they use. This way it does not matter what computer is actually running the algorithm.

The best algorithms **WIN** – both in performance and in business.





## Examined your own algorithms lately?

***Question:*** Productive and successful people continually examine their daily routines and activities to determine ways to do things better (finish tasks quicker, make more money, be more productive, have more free time, etc.).

Have you examined any of these areas recently? (select one)

**A)** Your time spent traveling and routes taken.

**B)** How you divide your time between work, school, and play.

**C)** Determine more effective ways to study.

**D)** Improving your efficiency around your home.

**E)** Other or none of the above





## The 5 Basic Steps of Software Development

1) Specification
   - Determine the scope of your problem and **what** you want your program to do.
2) Design
   - Determine the structures and algorithms necessary (**how**) to solve your problem at a high-level of abstraction.
3) Implementation
   - Start implementing your algorithms/structures on the computer.
4) Testing, Execution, and Debugging
   - Test your program on various data sets and fix any problems.
5) Maintenance
   - Over time, modify your program as necessary to handle new data or more complicated problems.





## Software Development Steps

***Question:*** Which of the 5 steps is most often the cause of projects (and your own assignments) being unsuccessful?

**A)** Specification

**B)** Design

**C)** Implementation

**D)** Testing

**E)** Maintenance

## Programming - Art or Science?

There is a debate whether programming is an art or a science.

- It is similar to a science because algorithms and data structures can be analyzed for performance and chosen with respect to their relevance to a particular problem.
- It is like an art or craft because skills of programmers vary widely, even with similar training, and the "best" solution to the problem is often open to debate.

In computer science, we teach you the "science" component.

- We want you to understand the choices you make and the reasons for them.
- However, students will all have different natural abilities and talents with respect to programming.
  - ⇨ If it is easy or natural for you, great! If not, then fall back on the science and the techniques we teach to help you!





## Programming: Art or Science?

***Question:*** What do you think programming is most like?

**A)** Art (creativity)

**B)** Science (experimentation)

**C)** Engineering (construction)

**D)** All of the above

**E)** Other or none of the above





## Programming: Experience

***Question:*** What is your programming experience?

**A)** I have never programmed before.

**B)** I have wrote instructions, recipes, manuals, or other precise information before (maybe not electronic).

**C)** I have wrote HTML or created web sites before this class.

**D)** I have experimented on my own with programming.

**E)** I have taken a programming class in high school or university.





## Conclusion

An ***algorithm*** is a precise sequence of steps to produce a result that is encoded in a language to produce a ***program***.

The five essential properties of an algorithm are:

- Inputs specified
- Output specified
- Precision
- Reasonable operations
- Finite

Following the five basic steps for developing solutions to problems on a computer will make you more successful and efficient while programming.





## Objectives

- Define: algorithm, program
- List and explain the five essential properties of an algorithm.
- Explain why special programming languages are used to communicate algorithms to the computer instead of English.
- List and explain the five basic steps of software development.

# COSC 122
# Computer Fluency

## Programming Basics

**Dr. Ramon Lawrence**
**University of British Columbia Okanagan**
**ramon.lawrence@ubc.ca**



## Key Points

1) We will learn JavaScript to write instructions for the computer.
- The fundamental programming concepts apply to all languages.

2) The key programming concepts covered:
- variables, values, and locations
- initialization and assignment
- expressions
- decisions and Boolean conditions





## History: The First Programmers

Did you know that the first programmers were almost all women?
- Women worked on the first computer - the ENIAC (Electronic Numerical Integrator and Calculator) developed for the US Army in 1946 by J. Eckert and John Mauchley.
- These women were recruited from the ranks of "computers", humans that used mechanical calculators to solve complex math problems before the invention of computers.
- These pioneer programmers laid the foundation of many of the original ideas including compilers and programming languages.





## Introduction to Programming

Remember that an ***algorithm*** is a precise sequence of steps to produce a result. A ***program*** is an encoding of an algorithm in a ***language*** to solve a particular problem.

There are numerous languages that programmers can use to specify instructions. Each language has its different features, benefits, and usefulness.

The language we will use is called JavaScript. However, our focus will be understanding the primary programming concepts that apply to all languages.





## Introduction to JavaScript

JavaScript is a *scripting* language used primarily for web pages.
- JavaScript was developed in 1995 and released in the Netscape web browser (since renamed to Mozilla Firefox).
- JavaScript is standardized and supported by most browsers.

Despite the name, JavaScript is not related to Java, although its syntax is similar to other languages like C, C++, and Java.
- There are some major differences between JavaScript and Java that will not concern us here.
- Aside: The term ***scripting*** means the language is interpreted (processed when needed) instead of compiled (converted to machine language directly). The difference is irrelevant to us.





## Some Quotes

If you can't write it down in English, you can't code it.
-- Peter Halpern

If you lie to the computer, it will get you.
-- Peter Farrar

## JavaScript: Basic Rules

To program in JavaScript you must follow a set of rules for specifying your commands. This set of rules is called a ***syntax***.

- Just like any other language, there are rules that you must follow if you are to communicate correctly and precisely.

Important general rules of JavaScript syntax:

- JavaScript is ***case-sensitive***.
  - ⇨ Main() is not the same as main() or MAIN()
- JavaScript accepts ***free-form layout***.
  - ⇨ Spaces and line breaks are not important except to separate words.
  - ⇨ You can have as many words as you want on each line or spread them across multiple lines.
  - ⇨ However, you should be consistent and make your code easy to read.





## Our Running Example
## Do you want fries with that?

We will use an example program for our discussion that calculates the total cost of a fast food order.

Inputs:

- `burger` – may be "none", "hamburger", or "cheeseburger"
- `fries` – may be "none", "small", or "large"
- `drink` – may be "none", "small", or "large"

Output:

- the `total` in dollars of the order including tax (7%)





## Fast Food Example






## Fast Food Example Code

```
var total;
var taxRate = 0.07;

total = 0;

if (burger == "hamburger" || burger == "cheeseburger")
   total = 0.99;

if (fries == "small")
   total = total + 1.19;
if (fries == "large")
   total = total + 1.79;

if (drink == "small")
   total = total + 1.49;
else if (drink == "large")
   total = total + 1.89;

total = total + total * taxRate;
```

Declare and initialize variables

Assignment

Decision using IF

Expression

**Flow of Execution**
-Start at first statement at top and proceed down executing each statement.
-For `if` statement only execute one of the possibilities if condition is true otherwise go to next statement after `if`.



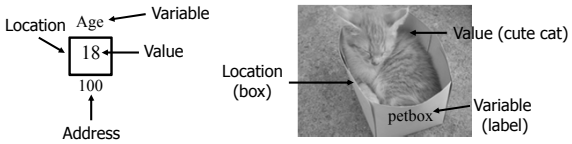


## Values, Variables, and Locations

A ***value*** is a data item that is manipulated by the computer.

A ***variable*** is the name that the programmer uses to refer to a location in memory.

A ***location*** has an address in memory and stores a value.




**IMPORTANT:** The *value* at a given location in memory (named using a variable name) can change using initialization or assignment.





## Values, Variables, and Locations Example

We want to store a number that represents the total order value.

Step #1: Declare the variable by giving it a name

```
var total;
```

- The computer allocates space for the variable in memory (at some memory address). Every time we give the name `total`, the computer knows what data item we mean.

Variable Name Lookup Table

| Name | Location | Type |
|---|---|---|
| total | 16 | number |

Memory

| | |
|---|---|
| 16 | ???????? |
| 20 | |
| 24 | |
| 28 | |

## Values, Variables, and Locations Example (2)

Step #2: Initialize the variable to have a starting value

- If you do not initialize your variable to a starting value when you first declare it, the value of the variable is ***undefined***.

Example:

```
total = 1;
```

Variable Name Lookup Table

| Name | Location | Type |
|---|---|---|
| total | 16 | number |

Memory

| | |
|---|---|
| 16 | 1 |
| 20 | |
| 24 | |
| 28 | |





## Values, Variables, and Locations Example (3)

Step #3: Value stored in location can be changed throughout the program to whatever we want using ***assignment*** ("**=**" symbol).

```
total = total * 5 + 20;
```

Variable Name Lookup Table

| Name | Location | Type |
|---|---|---|
| total | 16 | number |

Memory

| | |
|---|---|
| 16 | 25 |
| 20 | |
| 24 | |
| 28 | |





## Variable Rules

Variables are also called identifiers. An ***identifier*** is a name that ***must begin with a letter*** and cannot contain spaces.

The keyword **`var`** is used to declare to the computer that you want a variable created. This declaration is a type of ***statement***.

Rules:

- Every variable used in a program must be declared.
- Variables can be declared anywhere in the program, but usually should be declared right at the start.
- Variable names **ARE** case-sensitive. Numbers are allowed (but not at the start). Only other symbol allowed is underscore ('_');
- Beware of declaring two variables with the same name.
- The syntax of the language allows you to declare and initialize multiple variables in the same statement:

```
var total = 0, taxRate = 0.07;
```





## Aside: Good Variable Names

As a programmer you have flexibility on the names that you assign to your variables.

- However, names should be meaningful and explain how the variable is actually used in your program.

Example:

```
var t = 0;
var total = 0;
```

Avoid naming variables as reserved words. A ***reserved word*** is a string that has special meaning in the language.

- e.g. `if`, `var`, `else`





## Variables – Basic Terminology

***Question:*** Of the following three terms, what is most like a ***box***?

**A)** value

**B)** variable

**C)** location





## Variables - Definitions

***Question:*** Which of the following statements is correct?

**A)** The location of a variable may change during the program.

**B)** The name of a variable may change during the program.

**C)** The value of a variable may change during the program.

## Variables – Correct Variable Name

***Question:*** Which of the following is a valid JavaScript variable?

**A)** `aBCde123`

**B)** `123test`

**C)** `t_e_s_t!`





## General Syntax Rules

A program is a list of statements (instructions).

**PRIMARY RULE:** Every statement must be terminated by a semi-colon ";".

- Note the statement terminator character varies by language.

Other rules:

- You may have multiple statements on a line as long as each ends with a semi-colon.
- A statement may cross multiple lines.





## General Syntax Rules: Comments

***Comments*** are used by the programmer to document and explain the code. Comments are ignored by the computer.

There are two choices for commenting:

- 1) One line comment: put "`//`" before the comment and any characters to the end of line are ignored by the computer.
- 2) Multiple line comment: put "`/*`" at the start of the comment and "`*/`" at the end of the comment. The computer ignores everything between the start and end comment indicators.

Example:

```
/* This is a multiple line
     comment.
With many lines. */

// Single line comment
// Single line comment again
d = 5.0; // Comment after code
```





## Variable Types

A variable has a ***name*** for a data item and a ***type***.

- JavaScript is different than most languages because you do not have to tell the computer what type the variable is when you declare it. The variable can store any type (although it is not recommended to change types).

The data types that we will use are:

- numbers – both integers and float/doubles
- strings – sequences of characters
- Boolean – true or false





## Strings

***Strings*** are sequences of characters that are surrounded by either single or double quotes.

Example:

```
var personName = "Ramon Lawrence";
personName = "Joe Smith";
```

Question: What is the difference between these two statements?





## Rules for Strings in JavaScript

String rules:

- Must be surrounded by single or double quotes.
- Can contain most characters except enter, backspace, tab, and backslash.
  - ⇨ These special characters must be escaped by using an initial "`\`".
  - ⇨ e.g. `\n` – new line, `\'` – single quote, `\\` - backslash, `\"` – double quote
- Double quoted strings can contain single quoted strings and vice versa.
- Any number of characters is allowed.
- The minimum number of characters is zero "", which is called the *empty string*.
- String *literals* (values) have the quotation marks removed when displayed.

## Practice Questions

1) Write the statements to create two variables: one called `hourlyRate` and the other called `hoursWorked`.

2) Are the following variable names valid or invalid:

```
var A;
var A123;
var 123A;
var aReallyLongName;
```

3) Using your statements from question #1, write the code to calculate and store a person's salary by multiplying their `hoursWorked` times their `hourlyRate`.

4) Create a string variable that has an initial value of `'Joe's Place'`.





## The Assignment Statement

An ***assignment statement*** changes the value of a variable.

- ⇨The variable on the left-hand side of the **=** is assigned the value from the right-hand side.
- ⇨The value may be changed to a constant, to the result of an expression, or to be the same as another variable.
- ⇨The values of any variables used in the expression are always their values before the start of the execution of the assignment.

Examples:

```
var A, B;

A = 5;
B = 10;
A = 10 + 6 / 2;
B = A;
A = 2*B + A - 5;
```

Question: What are the values of A and B?





## Expressions

An ***expression*** is a sequence of operands and operators that yield a result. An expression contains:

- ◆***operands*** - the data items being manipulated in the calculation
  - ⇨e.g. 5, "Hello, World", myDouble
- ◆***operators*** - the operations performed on the operands
  - ⇨e.g. +, -, /, *, % (modulus - remainder after integer division)

An operator can be:

- ◆***unary*** - applies to only one operand
  - ⇨e.g. d = - 3.5; // "-" is a unary operator, 3.5 is the operand
- ◆***binary*** - applies to two operands
  - ⇨e.g d = e * 5.0; // "*" is binary operator, e and 5.0 are operands





## Expressions - Operator Precedence

Each operator has its own priority similar to their priority in regular math expressions:

- ◆1) Any expression in parentheses is evaluated first starting with the inner most nesting of parentheses.
- ◆2) Unary + and unary - have the next highest priorities.
- ◆3) Multiplication and division (*, /, %) are next.
- ◆4) Addition and subtraction (+,-) are then evaluated.





## String Operators: Concatenation

The ***concatenation operator*** is used to combine two strings into a single string. The notation is a plus sign '**+**'.

Example:

```
var string1 = "Hello";
var string2 = " World!";
var result = string1 + string2; //result = "Hello World!"
```

The plus sign is used for addition, but it makes sense as the symbol for string concatenation as well.

Using the same symbol as a operator in multiple different ways is called ***operator overloading***.





## Assignment

***Question:*** What are the values of `A` and `B` after this code?

```
var A, B;

A = 2;
B = 4;
A = B + B / A;
B = A * 5 + 3 * 2;
```

**A)** `A = 6, B = 36`

**B)** `A = 4, B = 26`

**C)** `A = 6, B = 66`

## String Concatentation

***Question:*** What is the value of result after this code?

```
var st1="Joe", st2="Smith";
var result = st1 + st2;
```

**A)** "Joe Smith"

**B)** "JoeSmith"





## String Concatentation (2)

***Question:*** What is the result after this code?

```
var st1="123", st2="456";
var result = st1 + st2;
```

**A)** 579

**B)** "579"

**C)** "123456"





## Running a JavaScript Program

We run JavaScript programs within a web browser.

This means several things:

- 1) The file that stores the program will be an HTML document. It should have a name like `myProgram.html`.
- 2) The JavaScript program is part of the HTML file.
- 3) Edit the document using a text editor. Test the document by opening it in Internet Explorer, FireFox, Chrome, or Safari.





## Hello World Example JavaScript Code

```
<html>
<head>
<title>HelloWorld using JavaScript</title>
</head>

<body>

<h1>

   <script type="text/javascript">
      document.write("Hello, world!");
   </script>
</h1>

</body>
</html>
```

`<script>` tag indicating code

JavaScript code

document is HTML document
document.write() puts that text into the document at this location





## Getting Input into a JavaScript Program

There are two ways to get input from the user into your program:

- 1) Make the user fill in form fields and get the value of those fields when the user clicks a button.
  - ⇨ We will see how to do this later.
- 2) Prompt the user with a separate window asking them for a value.





## Getting Input Using JavaScript Code

```
<html>
<head>
<title>Prompt for a Value using JavaScript</title>
</head>

<body>
<h1>
   <script type="text/javascript">
      var val = window.prompt("Enter a value: ");
      document.write(val);
   </script>
</h1>
</body>
</html>
```

Prompt for value from user

write out value retrieved

## Outputting from a JavaScript Program

There are three ways to output information to the user:

- 1) Have your code set the value of a form field.
- 2) Have your code write out text directly into the HTML document.
- 3) Open an alert output window to the user with a message.





## Outputting Data from JavaScript Code

```
<html>
<head>
<title>Display a Value using an Alert Window</title>
</head>

<body>
   <script type="text/javascript">
      var val = window.prompt("Enter a value: ");
      window.alert("You said: "+val);
   </script>
</body>
</html>
```

Prompt for value from user

Open up new window with message and value that the user just entered.

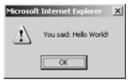




## Prompt and Output Example

Prompt window:


Alert (output) window:








## Input/Output Question

***Question:*** Assume the user typed in **10** when prompted. What is shown in the HTML document after this code?

```
var val = window.prompt("Enter a value: ");
window.alert("You said: "+val);
```

**A)** Nothing

**B)** `You said: 10`

**C)** Error





## Input/Output Question (2)

***Question:*** Assume the user typed in **10** when prompted. What is shown in the HTML document after this code?

```
var val;
window.prompt("Enter a value: ");
document.write("You said: "+val);
```

**A)** Nothing

**B)** `You said: 10`

**C)** `You said: undefined`

**D)** Error





## Practice Questions

For these questions, use slide #33 as an example. Do not copy the HTML code, just write the JavaScript statements.

1) Write the JavaScript code to print:

Hello, World!

Goodbye, World!

2) Write the JavaScript code to print:

Testing…

1..2..3..

1+2+3 = 6

1*2*3 = 6

Note: You must calculate 6 in both cases not just print it!

3) Write a program to calculate and print: (a=5, b=10)

c = 25*a+b-32

## Making Decisions

***Decisions*** allow the program to perform different actions in certain conditions.

- For example, if a person applies for a driver's license and is not 16, then the computer should not give them a license.

To make a decision in a program we must:

- 1) Determine the ***condition*** in which to make the decision.
  - ⇨ In the license example, we will not give a license if the person is under 16.
- 2) Tell the computer what actions to take if the condition is true or false.
  - ⇨ A decision always has a *Boolean* or true/false answer.

The syntax for a decision uses the ***if*** statement.





## Making Decisions Performing Comparisons

A ***comparison operator*** compares two values. Examples:

- 5 < 10
- N > 5 // N is a variable. Answer depends on what is N.

Comparison operators in JavaScript:

- > - Greater than
- >= - Greater than or equal
- < - Less than
- <= - Less than or equal
- == - Equal (Note: Not "=" which is used for assignment!)
- != - Not equal

The result of a comparison is a ***Boolean value*** which is either ***true*** or ***false***.





## Making Decisions Example Comparisons

```
var j=25, k = 45;
var d = 2.5, e=2.51;

// Determine if these comparisons are true or false

(j == k)          // false
(j <= k);         // true
(d == e);         // ??
(d != e);         // ??
(k >= 25);        // ??
(25 == j);        // ??
(j > k);          // ??
(e < d);          // ??

 j = k;
(j == k);         // ??
```





## Valid Comparison Operators Question

***Question:*** Select the operator that is invalid (not allowed).

**A)** !=

**B)** ==

**C)** <=

**D)** ≥





## Making Decisions If Statement

To make decisions with conditions, we use the ***if*** statement.

- If the condition is true, the statement(s) after **`if`** are executed otherwise they are skipped.
- If there is an **`else`** clause, statements after **`else`** are executed if the condition is false.

Syntax:

```
if (condition)
   statement;
```

**OR**

```
if (condition)
   statement;
else
   statement;
```

Example:

```
if (age > 17)
   alert("Adult!");
```

**OR**

```
if (age > 17)
   alert("Adult!");
else
   alert("Kid!");
```





## Making Decisions Block Syntax

Currently, using our if statement we are only allowed to execute one line of code (one statement).

- What happens if we want to have more than one statement?

We use the ***block syntax*** for denoting a multiple statement block. A block is started with a "**{**" and ended with a "**}**".

- All statements inside the brackets are grouped together.

Example:

```
if (age > 17)
{  window.alert("You are an adult");
   window.alert("You can vote!");
   ...
}
```

We will use block statements in many other situations as well.

## Making Decisions
## If Statement Example

```
var age;
var teenager, hasLicense;
age = window.prompt("Enter your age: ");

if (age > 19)
{  teenager = false;
   hasLicense = true;
}
else if (age < 13)
{  teenager = false;
   hasLicense = false;
}
else
{  teenager = true;  // Do not know if have license
   hasLicense = false;
}
document.write("Is teenager: "+teenager);
document.write("Has license? "+hasLicense);
```





## Making Decisions

***Question:*** What is the output of this code?

```
var num=10;

if (num > 10)
   document.write("big");
else
   document.write("small");
```

**A)** `big`

**B)** `small`

**C)** `bigsmall`





## Making Decisions (2)

***Question:*** What is the output of this code?

```
var num=10;

if (num != 10)
   document.write("big");
document.write("small");
```

**A)** `big`

**B)** `small`

**C)** `bigsmall`





## Making Decisions (3)

***Question:*** What is the output of this code?

```
var num=10;

if (num == 10)
{  document.write("big");
   document.write("small");
}
```

**A)** `big`

**B)** `small`

**C)** `bigsmall`





## Decision Practice Questions

1) Write a program that reads an integer *N*.

- If *N* < 0, print "Negative number", if *N* = 0, print "Zero", If *N* > 0, print "Positive Number".

2) Write a program that reads in a number for 1 to 5 and prints the English word for the number. For example, 1 is "one".

3) Write a program to read in your name and age and print them.

- a) Modify your program to also print "Not a teenager" if your age is greater than 19 otherwise print "Still a teenager".





## Nested Conditions and Decisions
## Nested If Statement

We ***nest*** `if` statements for more complicated decisions.

- Verify that you use blocks appropriately to group your code!

Example:

```
if (age > 16)
{  if (gender == "male")
   {  document.write("Fast driver!");
   }
   else
   {  document.write("Great driver!");
   }
}
else
{  document.write("Sorry! Too young to drive.");
}
```

## Making Decisions
## Nested If Statement Example

```
var salary, tax;
var married;

married = window.prompt("Enter M=married, S=single: ");
salary = window.prompt("Enter your salary: ");

if (married == "S")
{  // Single person
   if (salary > 50000)
      tax = salary*0.5;
   else if (salary > 35000)
      tax = salary*0.45;
   else
      tax = salary*0.30;
}  // End if single person
```





## Making Decisions
## Nested If Statement Example (2)

```
else if (married == "M")
{  // Married person
   if (salary > 50000)
      tax = salary*0.4;
   else if (salary > 35000)
      tax = salary*0.35;
   else
      tax = salary*0.20;
}  // End if married person
else  // Invalid input
   tax = -1;

if (tax != -1)
{  document.write("Salary: "+salary+"<br/>");
   document.write("Tax: "+tax+"<br/>");
}
else
   document.write("Invalid input!");
```





## Nested Conditions and Decisions
## Dangling Else Problem

The ***dangling else problem*** occurs when a programmer mistakes an else clause to belong to a different if statement than it really does.

- Brackets determine which statements are grouped together, not indentation! By default, an else with no brackets matches the closest if statement regardless of indentation.

Example:

**Incorrect**

```
if (country == "US"))
   if (state == "HI"))
      shipping = 10.00;
else  // Belongs to 2nd if!
   shipping = 20.00; // Wrong!
```

**Correct**

```
if (country == "US")
{  if (state == "HI")
      shipping = 10.00;
}
else
   shipping = 20.00;
```





## Nested Conditions and Decisions
## Boolean Expressions

A ***Boolean expression*** is a sequence of conditions combined using AND (**&&**), OR (**||**), and NOT (**!**).

- Allows you to test more complex conditions
- Group subexpressions using parentheses

Syntax:

| | |
|---|---|
| *(expr1)* **&&** *(expr2)* | - expr1 AND expr2 |
| *(expr1)* **||** *(expr2)* | - expr1 OR expr2 |
| !(expr1) | - NOT expr1 |

Examples:

```
var b;

1) b = (x > 10) && !(x < 50);
2) b = (month == 1) || (month == 2) || (month == 3);
3) if (day == 28 && month == 2)
4) if !(num1 == 1 && num2 == 3)
5) b = ((10 > 5 || 5 > 10) && ((10>5 && 5>10));// False
```





## Boolean Expressions

***Question:*** Is `result` `true` or `false`?

```
var x = 10, y = 20;
var result = (x > 10) || (y < 20);
document.write(result);
```

**A)** `true`

**B)** `false`





## Boolean Expressions (2)

***Question:*** Is `result` `true` or `false`?

```
var x = 10, y = 20;
var result = !(x != 10) && (y == 20);
document.write(result);
```

**A)** `true`

**B)** `false`

## Boolean Expressions (3)

***Question:*** Is `result` `true` or `false`?

```
var x = 10, y = 20;
var result = (x >= y) || (y <= x);
document.write(result);
```

**A)** `true`

**B)** `false`





## Making Decisions (4)

***Question:*** What is the output of this code?

```
var num=12;

if (num >= 8)
   document.write("big");
   if (num == 10)
      document.write("ten");
else
   document.write("small");
```

**A)** `big`

**B)** `small`

**C)** `bigsmall`

**D)** `ten`

**E)** `bigten`





## Making Decisions (5) Boolean Expressions

***Question:*** What is the output of this code?

```
var x = 10, y = 20;

if (x >= 5)
{  document.write("bigx");
   if (y >= 10)
      document.write("bigy");
}
else if (x == 10 || y == 15)
   if (x < y && x != y)
      document.write("not equal");
```

**A)** `bigx`

**B)** `bigy`

**C)** `bigxnot equal`

**D)** `bigxbigynot equal`

**E)** `bigxbigy`





## Practice Questions

1) Create the Boolean expressions in JavaScript for:

- a) x does not equal y OR y is greater than z
- b) x is greater than 0 AND less than 100
- c) x is not less than 0 OR greater than 100

2) Write a program that reads two numbers and prints them in sorted, descending order. Challenge: Do it for three numbers.

3) Challenge: Write a program that translates a letter grade into a number grade.

- Letter grades are A,B,C,D,F possibly followed by + or - with values 4,3,2,1, and 0. There is no F+ or F-. A + increases the value by 0.3, a - decreases it by 0.3. An A+ equals 4.0.
- You need to use two functions:
  - ⇨<variableName>.length – length of string given by variableName
  - ⇨<variableName>.charAt(0) – character at position 0 in string





## Review: Key Programming Concepts

Some key concepts in programming:

- ***variables*** – names for data items to be manipulated
- ***locations*** – addresses of data items in memory
- ***values*** – the value stored at a particular location and referenced using a given variable name
- ***initialization*** – setting beginning values for variables
- ***assignment*** – general form of initialization where the value of a variable is set to another value
- ***decisions*** – performing different actions based on testing a condition
- ***expressions*** – consist of operands and operators and yield a result





## Conclusion

We learned the basics of the JavaScript language to communicate instructions to the computer including:

- declaring and using variables
- initialization and assignment of values to variables
- expressions
- decisions and Boolean conditions

## Objectives

- Compare and contrast: algorithm and program
- List and define the key programming concepts covered.
- Explain the difference between variables, values, and locations.
- Remember the rules for variables, comments, and statements.
- Remember the rules for declaring and using strings.
- Understand and explain assignment operator.
- Define: operator, operand, unary, binary
- Remember operator precedence for expressions.
- Recall the string concatenation operator.
- Be able to write and execute JavaScript code in HTML files.
- Define: operator overloading





## Objectives (2)

- Know how to get input and send output to and from the user.
- Write decisions using the if statement.
- Define: Boolean, condition
- List and use the comparison operators.
- Explain the dangling else problem.
- Construct and evaluate Boolean expressions using AND, OR, and NOT.

# COSC 122
# Computer Fluency

## Iteration and Arrays

**Dr. Ramon Lawrence**
**University of British Columbia Okanagan**
**ramon.lawrence@ubc.ca**



## Key Points

1) A loop repeats a set of statements multiple times until some condition is satisfied.

2) Arrays are a data structure for storing multiple items using the same name. Individual items are referenced by index.





## Iteration & Looping Overview

A computer does simple operations extremely quickly.

If all programs consisted of simple statements and decisions as we have seen so far, then we would never be able to write enough code to use a computer effectively.

To make a computer do a set of statements multiple times we use ***looping structures***.

A ***loop*** repeats a set of statements multiple times until some condition is satisfied.

- Each time a loop is executed is called an ***iteration***.





## The While Loop

The most basic looping structure is the ***while*** loop.

A while loop continually executes a set of statements **while** a condition is true.

Syntax:

```
while (<condition>)
{  <statements>
}
```

Example:

```
var j=0;
while (j <= 5)
{  j=j+1;
   document.write(j);
}
```

Question: What does this print?





## The ++ and -- Operators

It is very common to subtract 1 or add 1 from the current value of an integer variable.

There are two operators which abbreviate these operations:

- ++- add one to the current integer variable
- -- - subtract one from the current integer variable

Example:

```
var j=0;

j++;     // j = 1;  Equivalent to j = j + 1;
j--;     // j = 0;  Equivalent to j = j - 1;
```





## The For Loop

The most common type of loop is the ***for loop***. Syntax:

```
for (<initialization>; <continuation>; <next iteration>)
{  <statement list>
}
```

Explanation:

- 1) initialization section - is executed once at the start of the loop
- 2) continuation section - is evaluated *before* every loop iteration to check for loop termination
- 3) next iteration section - is evaluated *after* every loop iteration to update the loop counter

## Iteration & Looping
## The For Loop

Although JavaScript will allow almost any code in the three sections, there is a typical usage:

```
for (i = start; i < end; i++)
{  statement
}
```

Example:

```
var i;

for (i = 0; i < 5; i++)
{  document.write(i);      // Prints 0 to 4
}
```





## For Loop and While Loop

The `for` loop is like a short-hand for the `while` loop:

```
var i=0;

while (i < 10)
{  document.write(i);
   i++;
}
```

```
var i;

for (i=0; i < 10; i++)
{  document.write(i);
}
```





## JavaScript Rules for Loops

The loop variable `i` must be declared.

- `i`, `j`, and `k` are used by convention, but you can pick any name you want.

The starting point of the iteration can begin anywhere, including negative numbers.

The continuation test must be an expression that results in a Boolean value. It should contain the loop variable to avoid an ***infinite loop***.

The next iteration usually changes the value of the loop variable by 1. It does not always have to be one, and it can be positive (such as +2) or negative (-1).





## Common Problems – Infinite Loops

***Infinite loops*** are caused by an incorrect loop condition or not updating values within the loop so that the loop condition will eventually be false.

Examples:

```
var i;

for (i=0; i < 10; i--)      // Should have been i++
{  document.write(i);       // Infinite loop: 0,-1,-2,..
}

i = 0;
while (i < 10)
{  document.write(i);       // Infinite loop: 0,0,0,..
}                           // Forgot to change i in loop
```





## Common Problems – Using Brackets

A one statement loop does not need brackets, but we will ***always use brackets***. Otherwise problems may occur:

```
var i=0;
while (i <= 10)
   document.write(i);       // Prints 0 (infinite loop)
   i++;                     // Does not get here…
// Forgot brackets { and } - i++ not in loop!
```

Do not put a semi-colon at the end of the loop:

```
for (i=0; i <= 10; i++);    // Causes empty loop
{  document.write(i);       // Prints 11
}
```





## Common Problems – Off-by-one Error

The most common error is to be "***off-by-one***". This occurs when you stop the loop one iteration too early or too late.

Example:

- This loop was supposed to print 0 to 10, but it does not.

```
for (i=0; i < 10; i++)
   document.write(i);  // Prints 0..9 not 0..10
```

Question: How can we fix this code to print 0 to 10?

## Looping Review

A loop structure makes the computer repeat a set of statements multiple times.

- `for` loop is used when you know exactly how many iterations to perform
- `while` loop is used when you keep repeating the loop until a condition is no longer true

When constructing your loop structure make sure that:

- you have the correct brackets to group your statements
- you do not add additional semi-colons that are unneeded
- make sure your loop terminates (no infinite loop)

Remember the operators ++ and -- as short-hand notation.





## For Loops

***Question:*** What is the output of this code?

```
for (i=0; i < 10; i++)
{  document.write(i);
}
```

**A)** nothing
**B)** error
**C)** The numbers 0, 1, 2, …, 9
**D)** The numbers 0, 1, 2, …, 10





## For Loops

***Question:*** What is the output of this code?

```
for (i=2; i < 10; i--)
{  document.write(i);
}
```

**A)** nothing
**B)** infinite loop
**C)** The numbers 2, 3, 4, …, 9
**D)** The numbers 2, 3, 4, …, 10





## For Loops

***Question:*** What is the output of this code?

```
for (i=0; i <= 10; i++);
{  document.write(i);
}
```

**A)** nothing
**B)** error
**C)** `11`
**D)** The numbers 0, 1, 2, …, 10





## Practice Questions: Iteration

1) How many times does each loop execute:

```
a) for(j=0; j <= 10; j--)
b) for(j=0; j <= 10; j++)
c) for(j=0; j < 10; j++)
d) for(j=-10; j <= 10; j++)
e) for(j=0; j <= 20; j=j+2)
```

2) Write a program to print the numbers from 1 to *N*.

- a) Modify your program to only print the even numbers.

3) Write a program to calculate and print the sum of the numbers from 1 to *N*. E.g. If *N* = 4, print 10 (1+2+3+4=10).



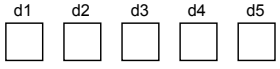


## Arrays Overview

Suppose you need many variables in your program.

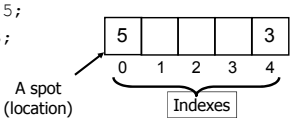
You could either create a separate name for each variable:

- `var d1, d2, d3, d4, d5;`




Or you could create an array that has multiple spots (indexes):

- `var myArray = new Array(5);`
- `myArray[0] = 5;`
- `myArray[4]= 3;`



## Arrays

An ***array*** is a collection of data items of the same type.

- Technically, JavaScript allows an array to contain different types, but that is not common in other programming languages.

The name of the array is an identifier like any other variable.

However, until you actually create the space for the array using ***new***, no array exists in memory.

- `var strings = new Array(10);`





## Array Indexing

When creating an array using `new`, the number in parentheses is the number of spots in the array:

- `var a = new Array(20);` // 20 elements

Note that the first spot of the array has index 0 instead of 1.

- In the previous example, the first index is 0 and the last is 19.

When an array is first created, all its values are undefined.

To access or set a value in an array, use its index:

- `a[0] = 10;` // Sets first spot to 10
- `a[19] = a[0];` // Sets last element same as 1st





## Array Details

If you provide an array index outside of the valid range, JavaScript will automatically re-size the array for you (for updates) or returned undefined (for reads).

- This is not common behavior. Most languages generate an error called an ***exception***.

To get the length of an array in your program:

- `var a = new Array(20);`
- `var size = a.length;` // Returns 20





## Arrays

***Question:*** What is the size of this array?

```
var myArray = new Array(10);
```

**A)** error
**B)** 10
**C)** 9
**D)** 11





## Arrays

***Question:*** What is the size of this array?

```
var myArray = new Array[10];
```

**A)** error
**B)** 10
**C)** 9
**D)** 11





## Arrays

***Question:*** What are the contents of this array?

```
var myArray = new Array(4);
myArray[3] = 1;
myArray[2] = 2;
myArray[1] = 3;
myArray[0] = 4;
```

**A)** error
**B)** 0, 1, 2, 3
**C)** 1, 2, 3, 4
**D)** 4, 3, 2, 1

## Application of Arrays

We will use arrays to create a web page that shows a slideshow of our favorite pictures.






## Practice Questions: Arrays

1) Create an array with name `myArray` that has 20 spots.

2) Set the value of the 1st spot to 10.

3) Set the value of the last spot to 1.

4) How do you know how many spots are in an array?

5) Create an array that has 10 spots. Put the numbers from 1 to 10 in the array.





## Conclusion

A ***loop*** allows the repetition of a set of statements multiple times until some condition is satisfied.

- We will primarily use for loops that have 3 components:
  - initialization - setup iteration variable start point
  - continuation - use iteration variable to check if should stop
  - next iteration - increment/decrement iteration variable

***Arrays*** are a data structure for storing multiple items using the same name. Arrays are often used with loops, as a loop can access each individual item by its index.





## Objectives

- Define: loop, iteration
- Explain the difference between the while and for loops.
- Explain what ++ and -- operators do.
- Be able to use a for loop structure to solve problems.
- Define: infinite loop
- Define: array
- Be able to use arrays to solve problems.

# COSC 122
# Computer Fluency

## Functions and Events

**Dr. Ramon Lawrence**
**University of British Columbia Okanagan**
**ramon.lawrence@ubc.ca**



## Key Points

1) Functions are used to group statements that perform a particular task so that they can be easily used.

2) Forms are used to input and receive output from the computer. A form consists of elements such as buttons, sliders, lists, and boxes.

3) Events are notifications that something occurs. Your program contains event handlers to indicate what to do when an event is detected.





## Important: Programming Incrementally

**NEVER** write code in a *monolithic* fashion.

**ALWAYS** write code by adding only a few lines or features at a time and then testing.

Thus, coding is an ***incremental process***.

- Write some code.
- Test in browser. Fix errors.
- Repeat (until done).

***Problem decomposition*** involves breaking down a large problem into subproblems which are easier to solve. Dividing problems into subproblems is called ***divide and conquer***.

Suggestion: Complete HTML document tags before writing complicated JavaScript code and event handling.





## Functions and Procedures

A ***procedure*** (or ***method***) is a sequence of program statements that have a specific task that they perform.

- The statements in the procedure are mostly independent of other statements in the program.

A ***function*** is a procedure that returns a value after it is executed.

We use functions so that we do not have to type the same code over and over. We can also use functions that are built-in to the language or written by others.





## Defining and Calling Functions and Procedures

**Creating** a function involves writing the statements and providing a ***function declaration*** with:

- a name (follows the same rules as identifiers)
- list of the inputs (called parameters) and their data types
- the output (return value) if any

**Calling** (or executing) a function involves:

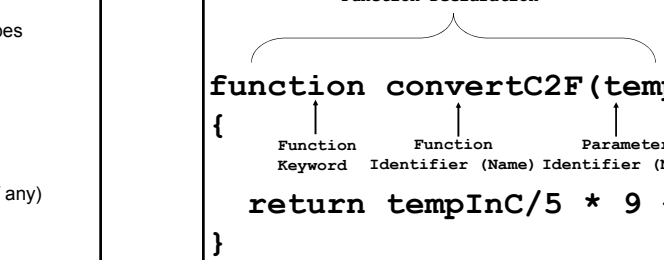
- providing the name of the function
- providing the values for all parameters (inputs) if any
- providing space (variable name) to store the output (if any)





## Defining a Function

Consider a method that converts a temperature in Celsius to Fahrenheit:



## Creating a Function

***Question:*** This function is supposed to take two numbers as input and return their sum. What is wrong with it?

```
function addTwoNum(num1)
{  var result = num1 + num2;
}
```

**A)** The two numbers are not added together.
**B)** The result of the addition is not returned back.
**C)** Only one number to add is passed into the function.
**D)** The name of the function is not correct.





## Creating a Function (2)

***Question:*** We want to create a function that multiplies two numbers together. Which of these functions is correct?

**A)**
```
multTwoNum(num1, num2)
{  return num1 * num2;
}
```

**B)**
```
function multTwoNum(num1, num2, num3)
{  return num1 * n2;
}
```

**C)**
```
function multTwoNum(num1, num2)
{  var result = num1 * num2;
}
```

**D)**
```
function multTwoNum(num1, num2)
{  return num1 * num2;
}
```





## Example: Calling Convert Function

This is how to call our `convertC2F` function:

```
var myCTemp = 50;                          1
var myFTemp;                               2

myFTemp = convertC2F(myCTemp);             3
alert(myCTemp +"C is = "+myFTemp+"F");     6

function convertC2F( tempInC )             4
{
   return tempInC / 5 * 9 + 32;            5
}
```

**Order of Operations**

What happens if we move the function convertC2F to the top of the code?





## Functions and Procedures Notes

- When declaring a function, you must put the parenthesis "`()`" after the name even if the function has no parameters.
- If a function returns nothing, you can just say "`return;`".
- *Parameter* is the term used for input when viewing from inside the function (function's perspective). *Argument* is the term used for input when viewing from outside the function.
- Functions are declared only once, but can be called as many times as you want.
- Execution of the method halts at the return statement and any value in the statement is passed back to the caller.
- You may have multiple return statements in a method, but only one will ever be executed for a given execution.





## Functions

***Question:*** What is the output of this code?

```
var num=9;

var result = doubleNum(num);
document.write(result);

function doubleNum(n)
{  return n*2; }
```

**A)** nothing
**B)** error
**C)** `9`
**D)** `18`





## Functions (2)

***Question:*** What is the output of this code?

```
function subtractNum(a, b)
{  return a-b; }

var x=5, y=8;

var result = subtractNum(x,y);
result = result + subtractNum(y,x);
document.write(result);
```

**A)** error
**B)** `3`
**C)** `-3`
**D)** `0`

## Functions (3)

***Question:*** What is the output of this code?

```
var num=9;

var result = doubleNum(doubleNum(num));
document.write(result);

function doubleNum(n)
{  return n*2; }
```

**A)** 36

**B)** 18

**C)** 9

**D)** error





## Functions (4)

***Question:*** What is the output of this code?

```
function evenOrOdd(n)
{  if (n % 2 == 0)
   {  return "even";
   }
   else
   {  return "odd";
   }
}
var num = 10;
document.write(evenOrOdd(11));
document.write(evenOrOdd(num));
```

**A)** oddodd

**B)** oddeven

**C)** evenodd

**D)** eveneven





## Built-In Functions

JavaScript has many built-in functions that you can use. These methods are grouped into objects.

- An ***object*** is a related group of code and data.

(Some of the) pre-defined objects in JavaScript:

- Array
- Date
- Math
  - ⇨ Functions: abs(x), floor(x), min(x,y), max(x,y), random(), sqrt(x)
- Number
- String
  - ⇨ Functions:
    - substring(start, end) - start is first character index, end is last index (not inc.)
    - charAt(index) - character at particular location in string (starting at 0)
    - Others: toUpperCase(st), toLowerCase(st)





## Built-in Function Example

```
var str = "hello, world!";
str = str.toUpperCase();
str = str.substring(0,5);
window.alert(str);
```

System realizes your data is a string and converts it to a String object automatically for you.

```
var num = 49;
window.alert(Math.sqrt(num));
```

← Use functions in built-in Math object

```
// Create two random numbers between 0 and 1
window.alert("My random number: "+Math.random());
window.alert("My next random number: "+Math.random());
```





## Advanced: Calling Object Methods

A method is called on an object by supplying an object instance and the name and arguments to the method.

Syntax:

*objectInstance.methodName*(*arguments*)

Remember:

- Each object has its own methods that it can perform.
- Each object has its own area of memory storing its data.

Tricky: A String object may be created for us automatically, so we do not always have to create String objects. We use the already created Math object for math functions.





## Practice Questions: Functions

1) Write a function that returns the sum of three numbers.

2) Write a function that returns the largest of two numbers.

3) Write a function that writes out the numbers from 1 to N where N is its input parameter.

4) Write a function that determines leap years:

```
function isLeapYear(year)
```

- A leap year is a year divisible by 4, except years divisible by 100 and not by 400. (i.e. 1900 is not a leap year, 2000 is.)

For each function, provide a sample function call.

## Input Forms

A ***form*** is an input page that contains controls such as buttons, lists, and boxes that allow the user to input information.

Below is an example form that we will create using HTML:



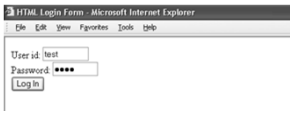



## HTML Forms

A ***form*** is created in HTML using a **`form`** tag and **`input`** tags (one for each input control).

Below is a login form with user id and password boxes:

```
<form name="MyForm" method="post" action="validateLogin.html">
User id: <input type="text" name="username" size="8"/><br/>
Password: <input type="password" name="password" size="8"/><br/>
<input class="submit" type="submit" name="Sub" value="Log In"/>
</form>
```








## HTML Input Types

An ***input type*** is a control that allows the user to communicate information with the computer.

An input control is specified using the **`input`** tag. The other key attributes are a **`type`** and a **`name`** for the control.

First name:
Last name:

```
<form name="fname" method="post" action="http://www.google.ca">
First name: <input type="text" name="firstname"/> <br/>
Last name:  <input type="text" name="lastname"/>
</form>
```

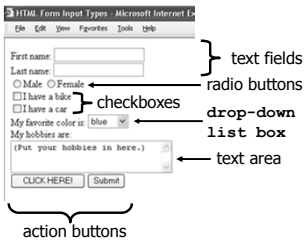
What type of control is it?

What is its name? Use the name in code like a variable.





## HTML Input Types








## HTML Input Types (2)

- Text Fields - for text (use password type for hidden)

User id: test
Password: ••••

```
User id: <input type="text" name="username"/><br/>
Password: <input type="password" name="password"/>
```

- Radio Buttons - for distinct (only one) choice

Male Female

```
<input type="radio" name="gender" value="male"/>Male
<input type="radio" name="gender" value="female"/>Female
```

- Checkboxes - for yes/no

I have a bike
I have a car

```
<input type="checkbox" name="bike"/>I have a bike  <br/>
<input type="checkbox" name="car"/>I have a car
```





## HTML List

A drop-down list is created in HTML by using the **`select`** tag and an **`option`** tag for each item in the list.

- Each item in the list has a display text label and a value.

```
<select name="colors">
<option value="R">red</option>
<option value="B" selected="selected">blue</option>
<option value="Y">yellow</option>
<option value="G">green</option>
</select>
```


- Notes:
  - ⇨ `selected` attribute indicates the list item is selected.
  - ⇨ Depending on the list, multiple items may be selected at the same time.
  - ⇨ size attribute of `SELECT` tag indicates how many items are visible.

## HTML Text Area

A text area is a text field that has a certain size in rows and columns. It is created in HTML by using the **textarea** tag.

(Put your hobbies in here.)

```
<textarea name="hobbies" rows="3" cols="30">
(Put your hobbies in here.)
</textarea>
```

- Notes:
  - Use `readonly` attribute to not allow the user to edit the field.





## HTML Buttons

A button performs an ***action*** when clicked.

There are some default buttons with special names:

- `reset` - clears all form fields
- `submit` - sends all data in form fields to web server

how and where to send data when submit button is clicked

```
<form name="myForm" action="doAction.html" method="get">
<input type="reset" value="Reset"/>
<input type="submit" value="Submit to Server"/>
</form>
```

Reset | Submit to Server





## HTML User-Defined Buttons

User buttons can be created using either the **button** tag or **input** tag with type as button.

```
<input type="button" value="Click Button 1" name="button1"/>
```

Click Button 1

```
<button name="button2">My Button 2</button>
```

My Button 2

```
<button name="button2"><img src="redball.gif"></button>
```

We must specify what to do when a user clicks on a button. A user click is one type of ***event***.





## Label Tag

There is a **label** tag that will associate a label with an input tag and clicking on the label gives that input focus.

Example:

```
<label for="firstName">First Name:</label>
<input type="text" id="firstName" />
```





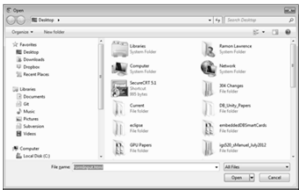
## File Input (Browse for file button)

There is a **input** type called **file** that will create a text field and button to allow user to select a file on their computer.

Example:

```
<input type="file" id="fileselect" />
```

Choose File HTMLFormInput.html







## HTML Inputs

***Question:*** A form has three radio buttons on it in a group (all have the same name). How many radio buttons can the user select of the three? (Select ***one*** of the correct answers.)

**A)** 0
**B)** 1
**C)** 2
**D)** 3

```
<!DOCTYPE html>
<html>
<head><title>HTML Form Question</title></head>
<body style="background-color:white">
<form name="myForm" action="http://people.ok.ubc.ca" method="get">
Name: <input type="text" name="name"> <br>
Student id: <input type="password" name="studentid"/> <br>

Year:
<input type="radio" name="year" value="1"/>1
<input type="radio" name="year" value="2"/>2
<input type="radio" name="year" value="3"/>3
<input type="radio" name="year" value="4"/>4<br>

My major is:
<select name="major">
<option value="ARTS">Arts</option>
<option value="SCIENCE">Science</option>
<option value="COSC" selected="selected">Computer Science</option>
<option value="ENGL">English</option>
<option value="PSYC">Psychology</option>
</select><br>

My minors are:<br>
<input type="checkbox" name="Arts">Arts<br>
<input type="checkbox" name="English">English<br>
<input type="checkbox" name="Psychology">Psychology<br>
<input type="checkbox" name="Science">Science<br>

Other notes:<br/> <textarea rows="4" cols="30"></textarea><br>

<input type="button" value="Register">
<input type="reset" value="Reset">
<input type="submit" value="Submit">
</form>
</body></html>
```

Given this HTML code:

1) Draw what the page looks like.

2) Indicate what each button does.





## GUI Programming Philosophy

In ***graphical applications***, the programmer must ***react*** instead of ***dictate*** the events that occur in a program.

As a programmer, you design a graphical user interface with windows, buttons, and components that the user can interact with. You do not know the order or the sequence of events the user will generate, but you must be able to react to them.





## Events and Event Handling Overview

An ***event*** is a notification to your program that something has occurred.

- For graphical events (mouse click, data entry), the browser notifies your program that an event occurred.
  - ⇨ There are different ***kinds*** of events such as keyboard events, mouse click events, mouse movement events, etc.

An ***event handler*** is part of your program that is responsible for "listening" for event notifications and handling them properly.

An ***event source*** is the user interface component that generated the event.

- A button, a window, and scrollbars are all event sources.
- The event source is **NOT** the user, the mouse, or the keyboard.





## HTML Input Element Events

Form elements in HTML can be made to listen and perform actions when events occur. The type of event that can be detected depends on the input type.

When you declare the form element, you can also provide:

- the ***events*** that you are interested in
- the ***actions*** (code functions) that should be run when the event occurs

Example: A button that pops up a window when clicked:

Event source

```
<input type="button" value="Button 1" name="button1"
          onclick="window.alert('You clicked me!');"/>
```

Event — Event handler





## HTML Input Element Events (2)

Buttons can be made to handle these events (and others):

- `onclick` - occurs when button is clicked
- `ondblclick` - button is double-clicked
- `onmouseover` - mouse pointer passed over button

Common events for text fields and areas:

- `onkeypress` - key has been pressed on the keyboard

For all form elements:

- `onchange` - value of element has been changed
- `onselect` - element has been selected by user
- `onsubmit` - occurs before form data is submitted





## HTML Event Example

Below is an example of a form that detects and alters its appearance using HTML events:

## HTML Event Code

```
<html>
<head><title>HTML Events</title></head>

<body style="background-color:white">
<form name="myForm" action="doAction.html" method="get"
                              onsubmit="formSubmit()">

First name: <input type="text" name="firstname"
              onkeypress="keyPress(event)"> <br>
Last name: <input type="text" name="lastname"
              onkeypress="return noNumbers(event)"> <br>

<input type="radio" name="sex" value="male"
              onclick="changeFontUp()">Male
<input type="radio" name="sex" value="female"
              onclick="changeFontDown()">Female<br>

<input type="checkbox" name="bike" onclick="noCar()">I have a bike<br>
<input type="checkbox" name="car" onclick="checkCar(this)">I have a
car<br>
```

Keypress events

OnClick events





## HTML Event Code (2)

```
My favorite color is:
<select name="colors" onchange="changeColor(this)">
<option value="red">red</option>
<option value="blue" selected="selected">blue</option>
<option value="yellow">yellow</option>
<option value="green">green</option>
</select><br>

My hobbies are:<br>
<textarea rows="5" cols="30" name="ta">
</textarea><br>

<input type="button" value="CLICK HERE!" onclick="count++;
myForm.ta.value=myForm.ta.value+'You clicked '+count+' times!\n';">
<input type="reset" value="Reset"
    onclick="return confirm('Are you sure?');"/>
<input type="submit" value="Submit" name="submitButton"
    onmouseover="overSubmit()" onmouseout="offSubmit()">
</form>
```

onChange event

Mouse moves over and off events.





## HTML Event Code (3)

```
<script type="text/javascript">

var count=0;                    // Global variable

function formSubmit()
{   window.alert("You are trying to submit the form!"); }

function getKey(ev)
{   // Handles differences between IE and other browsers
    var keyVal;
    if (window.event) // IE
        keyVal = ev.keyCode;
    else if (e.which) // Netscape/Firefox/Opera
        keyVal = ev.which;
    return keyVal;
}

function keyPress(ev)
{   window.alert("You pressed: "+String.fromCharCode(getKey(ev)));  }
```





## HTML Event Code (4)

```
function noNumbers(ev)
{
    var keyVal = getKey(ev);
    if (keyVal >= 48 && keyVal <= 57) // Key pressed is a number
        return false;
}

function changeFontUp()
{   document.body.style.fontSize="120%"; }

function changeFontDown()
{   document.body.style.fontSize="80%"; }

function changeColor(el)
{   document.body.bgColor = el.value; }

function checkCar(el)
{   if (el.checked)
        window.alert('Get some exercise!  Ride a bike!');
    else
        window.alert("You didn't need a car anyways!");
}
```

Changes font size of all text in document.

Change background color of document.





## HTML Event Code (5)

```
function overSubmit()
{   myForm.ta.value = "Time to submit the form, eh?";}

function offSubmit()
{   myForm.ta.value = "Changed your mind.  Chicken!";}

function noCar()
{   if (myForm.car.checked && myForm.bike.checked)
    {
        myForm.car.checked = false;
        myForm.ta.value = "Sold the car to get the bike.";
    }
}
</script>
</body>
</html>
```

A checkbox has a checked property.

myForm (form) has a text area called ta with a property value





## Advanced: HTML Document Objects

Your JavaScript program has access to all parts (called ***objects***) of your HTML document.

The entire document is represented by a **`document`** object.

- You can change foreground and background colors using it.
- When you name your document parts (such as forms and inputs), you can later refer to and change the properties of these elements using your JavaScript code.

A ***property*** is information about an object.

- Properties include `value`, `fgcolor`, `bgcolor`, `name`, and `type`. Other properties depend on the type of object.
- To change a property value using assignment, provide the name of the object then "." then the property name.

**Watch for:** The keyword "**`this`**" refers to the current object and its properties.

## Advanced: How are HTML pages created?

1) An HTML page can be created once (***statically***) and saved on a server. Every request for the page returns it exactly as it was originally created.

2) An HTML page can be produced ***dynamically*** by program code running on the server.

- The server-side code can access databases, run functions, or change the appearance or function of the page in response to user input and preferences.





## Advanced: When is JavaScript code executed?

JavaScript code is executed:

- 1) While the page is being loaded
  - ⇨A browser builds a page by reading through the HTML file, figuring out all tags and preparing to build page.
  - ⇨Then, it removes JavaScript tags and all text between them, and does whatever the JavaScript tells it to do.
  - ⇨Example: We have used `document.write()` to tell the browser to put text into the HTML document.
- 2) Interactively after the page is displayed (most common).
  - ⇨Example: HTML elements (such as buttons) may have events associated with them that run JavaScript code.





## Aside: New Windows

It is easy to open up a new browser window in JavaScript.

Use the **`window.open()`** method and provide the file URL.

```
<input type="button" value="Open a Window"
  onclick="window.open('HTMLCalculator.html', 'calcName', 'resizable=yes');">
```

**File URL** **Window name** **resizable?**





## Events

***Question:*** What is the event, event source, and event handler in this code?

```
<input type="button" value="Button 1" name="button1"
                                onclick="doButtonClick()">
```

**A)** event – `button`, event source – `button`, event handler - `onclick`

**B)** event – `onclick`, event source – `the mouse`, event handler - `onclick`

**C)** event – `onclick`, event source – `button`, event handler – `doButtonClick()`

**D)** event – `onclick`, event source – `button1`, event handler - `doButtonClick()`





## Event Names

***Question:*** Find the names of the three types of events below. Select the appropriate order of event names.

- happen when the mouse is clicked
- happen when a key is pressed
- happen when the mouse passes over something

**A)** `onmouseclick, onkeypress, onmouseover`

**B)** `onclick, onkeyboardpress, onmouseout`

**C)** `onmouseclick, onkeyboardpress, onmouseover`

**D)** `onclick, onkeypress, onmouseover`





## Events

***Question:*** Philosophical challenge: If an event occurs but there is no code to handle it, did it actually happen?

- Example: You click on a button and see nothing change. Did something happen?

**A)** Yes, the event happened, but it was ignored by the operating system.

**B)** Yes, the event happened, but it was ignored by our program.

**C)** No, the event did not happen because our program was not listening for it.

## Events and Case-Sensitivity

***Question:*** TRUE or FALSE: Event names are case-sensitive.

**A)** TRUE
**B)** FALSE





## Conclusion

***Functions*** and ***procedures*** are used to group statements that perform a particular task so that they can be easily used.

- Functions must be declared before they can be called (used).

***Forms*** are used to send input and receive output from the computer. A form consists of elements such as buttons, sliders, lists, and boxes.

- HTML forms use the `form` and `input` tags.

***Events*** are notifications that something occurs. ***Event handlers*** are code statements that you write indicating to the computer what should be done when an event happens.





## Objectives

- Define: function, procedure
- Explain the difference between creating and calling a function.
- Explain the difference between an argument and a parameter.
- Define: form. Be able to draw forms from code.
- List the different types of buttons. Define a button action.
- Define: event, event handler, event source

# COSC 122
# Computer Fluency

## Spreadsheets

**Dr. Ramon Lawrence**
**University of British Columbia Okanagan**
**ramon.lawrence@ubc.ca**



## Key Points

1) Spreadsheets are programs for storing and manipulating data that is represented as a table of cells.

2) Each cell has a row number and column label which combine to represent its address.

3) Spreadsheets allow you to organize data and write formulas to do computations. They are a powerful tool for data storage and analysis.





## Spreadsheet Overview

A ***spreadsheet*** organizes information into a two-dimensional array of cells (a *table*).

A ***cell*** has two components:

- an address - specified given a row and column number
- a location - that can store a number, text, or formula

The power of a spreadsheet is that we can write simple formulas (commands) to perform calculations and immediately see the results of those calculations.

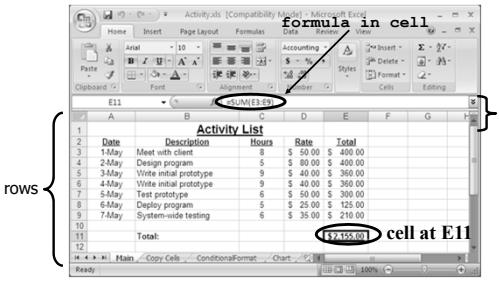
Spreadsheets are very common in accounting and reporting applications.





## Spreadsheet Addressing

A ***cell*** is identified by a row number and column letter.








## Spreadsheet Addressing

The rows in a spreadsheet are numbered starting from 1.

The columns are represented by letters.

- A is column 1, B is column 2, …, Z is column 26, AA is column 27, …

A cell is identified by putting the column letter first then the row number.

- e.g. B3 is the 2nd column and the 3rd row.

Question: What column number is AD? How about BAD?

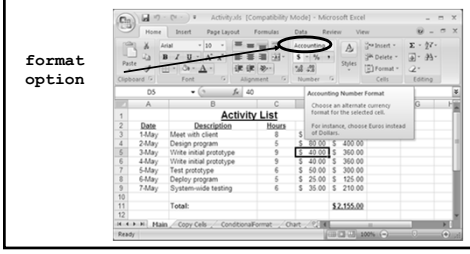




## Spreadsheet Data Entry

An entry can be added to a cell by clicking on it and typing in the data. The data may be a number, text, or a date.

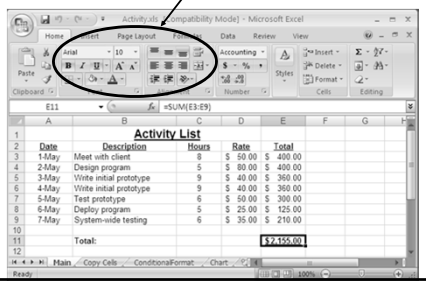
- The spreadsheet attempts to detect the data type and format it accordingly. It is also possible to manually format the data.



## Spreadsheet Formatting

We can format cells in italics, underline, and bold similar to a text editor. It is also possible to justify data and change fonts.

**format and justify shortcuts**








## Spreadsheet Selecting Cells

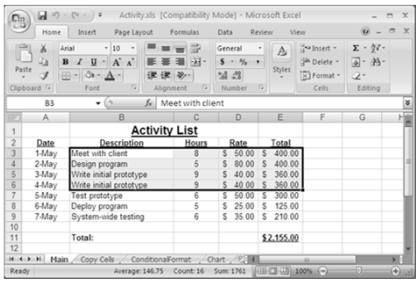
Multiple ways of selecting cells:

- 1) With the mouse, (left) click and drag mouse to select a rectangle region of cells.
- 2) With keyboard, hold `SHIFT` key and use arrow keys to select a rectangle region of cells.
- 3) With mouse and keyboard, while holding `CTRL` key, (left) click on individual cells to select non-contiguous cells.
- 4) Click on a row number to select a whole row.
- 5) Click on a column header to select a whole column.





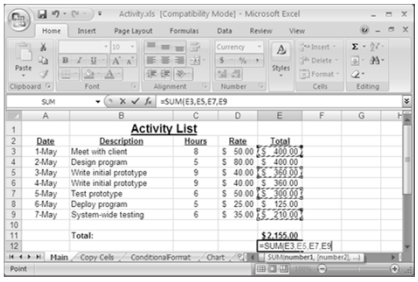
## Range Selecting Cells Example








## Selecting Individual Cells Example








## Manipulating Cells

Once you have selected one or more cells, there are several common actions you can perform:

- 1) DELETE
  - ⇨ delete the contents of all cells by pressing delete key
  - ⇨ delete the contents and the cell locations (then shift remaining) by selecting `Edit` menu, `Delete...` or `Delete…` from pop-up menu (brought up by right click).
- 2) Cut, Copy, Paste
  - ⇨ cut - copies selected cells to clipboard and removes from document
  - ⇨ copy - copies selected cells to clipboard
  - ⇨ paste - copies cells in clipboard to sheet starting at currently selected cell
- 3) Add selected cells to a formula (requires that you were previously constructing a formula before selecting the cells).





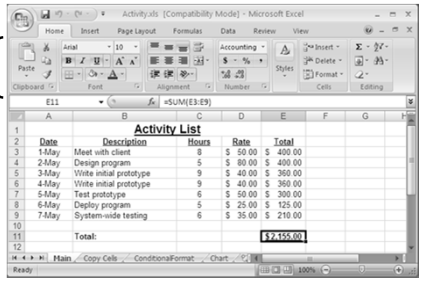
## Manipulating Cells - Filling

***Filling*** combines copy and paste.

There is a small box or tab beyond the cell's lower right corner (fill handle). Grab it with the cursor and pull to other cells.

## Cut, Copy, Paste

cut, copy, paste

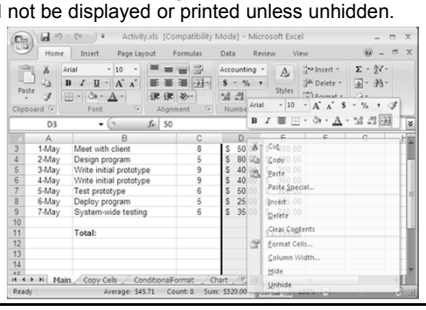







## Hiding Columns and Rows

You can ***hide*** a column or row by right-clicking on the column or row header and selecting **`Hide`**. The column/row still exists but will not be displayed or printed unless unhidden.



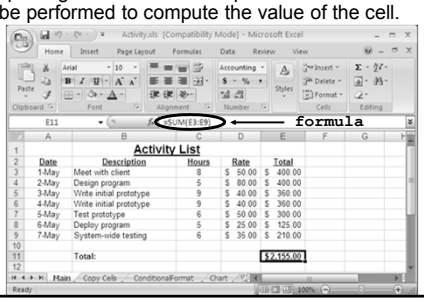




## Entering Formulas

A ***formula*** is any expression that begins with an equal sign ("**=**").

The equal sign indicates to the spreadsheet that a calculation must be performed to compute the value of the cell.








## Formula Expressions

A ***formula*** expression can consist of literals (numbers, text strings), operators, functions, and cell references.

Simple mathematical expressions:

- `= 1 + 5`
- `= 1.5 * 3.14 + 42`

Common functions:

- `= ROUND(PI,2)` // Result is 3.14
- `= CONCATENATE("Hello", " World")` // Hello World
- Other common functions for trigonometry, dates, and financial.





## Formula Expressions

The power of formulas comes from using cell references (similar to variable names in programming).

Cell reference examples:

- `= A1 + A2`
- `= B1 + A3 - A4`





## Spreadsheets
## Selecting Cells

***Question:*** Which method allows you to select non-contiguous cells in a spreadsheet?

**A)** hold `SHIFT` key and use arrow keys

**B)** With the mouse left click on a cell and drag mouse

**C)** hold `CTRL` key and use arrow keys

**D)** hold `CTRL` key and left click on cells

## Spreadsheets Formulas

***Question:*** A cell contains the following: **=3+5*2** What is the value of the cell?

**A)** 13

**B)** 16

**C)** =3+5*2





## Spreadsheets Formulas

***Question:*** A cell contains the following: **'ABC'+'DEF'** What is the value of the cell?

**A)** error

**B)** ABCDEF

**C)** 'ABC'+'DEF'





## Advanced Spreadsheet Addressing

The dollar sign "**$**" is a special symbol that indicates an absolute address.

- By default, addresses are "relative" in the sense that if they are in a formula that is copied to another cell, they will be changed relative to where they were copied from their origin.

Example:

- Cell `A1` has the formula `=A2+B1`
- Copy contents of cell `A1` to cell `C4`.
- Formula changes to `=C5+D4` because moved down three rows and over two columns.
- If cell `A1` had the formula `=$A$2+$B$1`, then the same formula would be in cell `C4`.
- Question: What if formula was `=$A2+B$1`?





## Spreadsheets Formulas and References

***Question:*** Cell **A1** contains the following: **=$B2+D$4** What is the formula if the cell is copied to cell **D3**?

**A)** error

**B)** =$B2+D$4

**C)** =$B4+F$4

**D)** =$B4+G$4



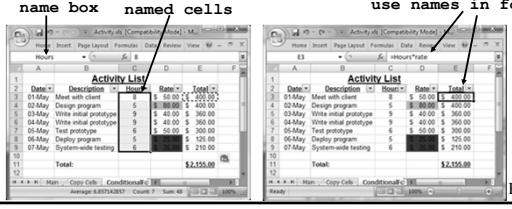


## Naming Cells

Instead of referring to cells by their address, you can give a cell a name and use that name in cell formulas.

- This makes it easier to read and understand formulas.
  - ⇨ Like programming variables where we use names instead of addresses to refer to data locations.

Example: Refer to columns by name `Hours` and `Rate`.








## Aggregate Formulas

An ***aggregate formula*** computes a summary function over a range of cells. The values can either be literals or cell locations.

Common functions are:

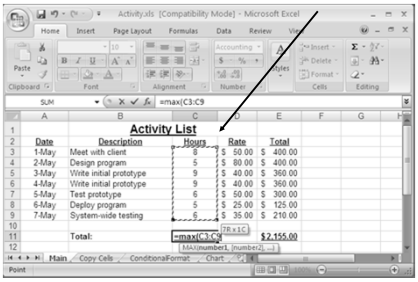
- `MIN(<value list>)` - returns minimum value in list
- `MAX(<value list>)` - returns maximum value in list
- `SUM(<value list>)` - returns sum of all values in list
- `AVERAGE(<value list>)` - returns average of values in list
- `COUNT(<value list>)` - returns count of values in list
- `MEDIAN(<value list>)` - returns median value of list

If specifying a cell rectangle, give the upper left and lower right corners, separated by a colon.

- e.g. `=average(A3:E6)` - rectangle of 4 rows and 5 columns

## Aggregate Formula Example

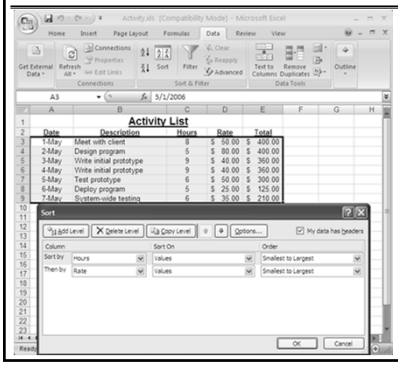
`building formula by selection`








## Sorting Data




Data can be sorted by selecting the **`Sort`** option under the **`Data`** menu.

Select the column(s) to sort on.





## Spreadsheets Aggregate Formulas

***Question:*** Assume the three cells in the range A1:C1 contain numbers. Which of these formulas is **ALWAYS** the largest?

**A)** MAX(A1:C1)

**B)** MIN(A1:C1)

**C)** COUNT(A1:C1)

**D)** SUM(A1:C1)

**E)** none of the above are always guaranteed to be the largest





## Charts

A ***chart*** is a graphical representation of spreadsheet data.

A chart is of a particular type (line, bar, etc.) and requires the user to supply the data that will be displayed in the chart.

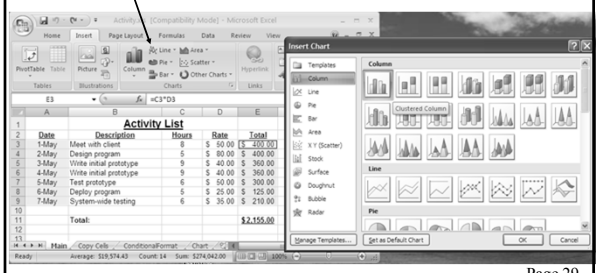




## Chart: Step #1 - Select Data and Type

Select **Insert**, then click **`Chart`** Icon, and pick the chart type.

`chart shortcut`

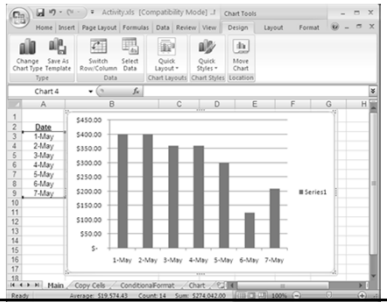






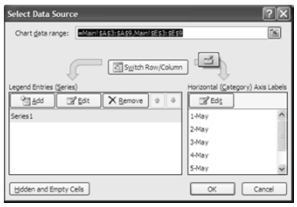

## Chart Options

Chart design tools allows you to modify the data in the chart, change the chart type, and move the chart in the Worksheet.



## Chart: Step #2 - Verify Data

You may modify the data displayed in the chart using the `Select Data` option. This includes adding legends.

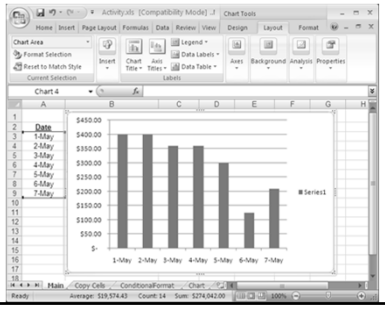







## Chart: Step #3 - Chart Options

Under **`Layout`** (**`Chart Tools`**) you can set the title, legend, and colors. There are more format options under **`Format`**.

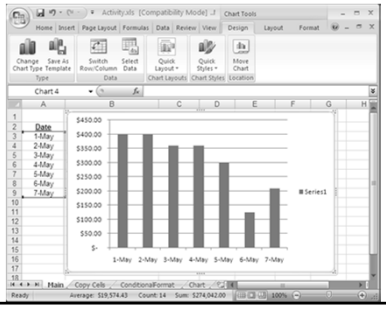






## Chart: Step #4 - Chart Location

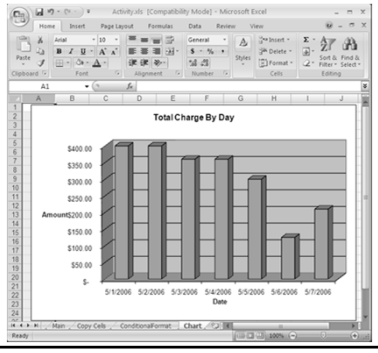
Put chart on an existing sheet or on its own sheet by selecting **`Move Chart`** in the **`Design`** area.







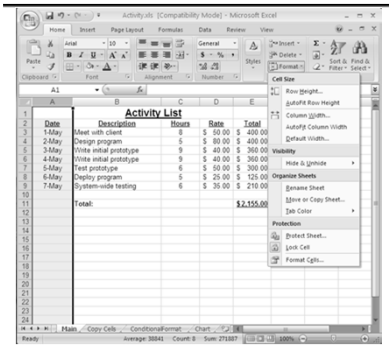
## Final Chart








## Other Formatting: Column Width




Resizing columns:

Auto-resize by double clicking on border between columns or using the `Format` option.

May also right-click on column to get `Format` option in the pop-up menu.

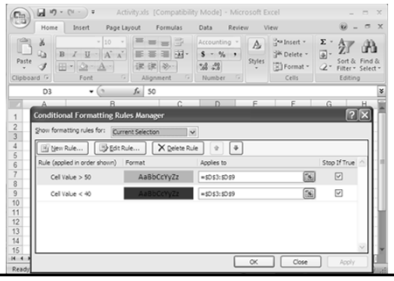




## Conditional Formatting

***Conditional formatting*** allows you to change the cell format based on data values. This is accessible under **`Styles`**.

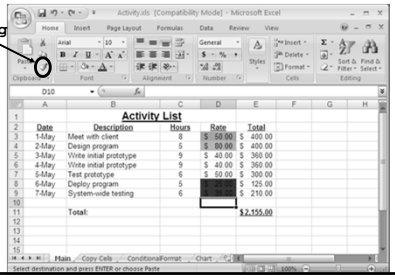
- Other options: data bars, color scales



## Conditional Formatting Result

The paint format button allows you to copy formatting to many cells. Select the cell, click paint button, then highlight cells to have identical formatting.

**paint formatting button**








## Spreadsheets for Data Management

A spreadsheet is often used as a simple form of a "database". A database is an organized representation of information.

- Examples: schedules and calendars, timesheets, expenses and finances, records, notes, and recipes, data research/analysis

We can use a spreadsheet as a database by:

- Using a row to store all the information about something we want to represent.
- Giving each column a meaningful name. A column represents a property or feature of the object stored in the row.
- Using the formulas to calculate new facts from the data.
- Using sorting to organize the data by key features.
- Using simple filtering (querying) to only show the most important data or data of interest.





## Filtering

A ***filter*** shows a subset of the rows in the spreadsheet by only showing rows that pass a given condition (test).

For our purposes, the **`Auto Filter`** under the **`Data`** then **`Filter`** menu is sufficient.

Once you select `Auto Filter`, each column heading has a drop-down list. By selecting a filtering criteria from the list, you can limit the rows that are displayed.

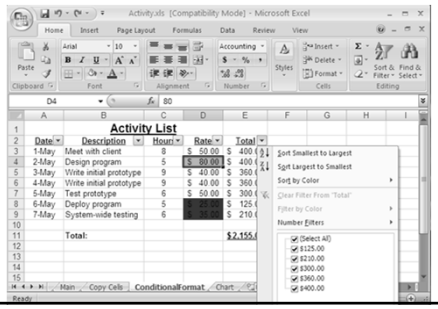
It is possible to filter on more than one column at the same time.





## Filter Example

Filter on `Total` column: Can select a value, Top 10 items, or write a custom filter.








## Custom Filter Example

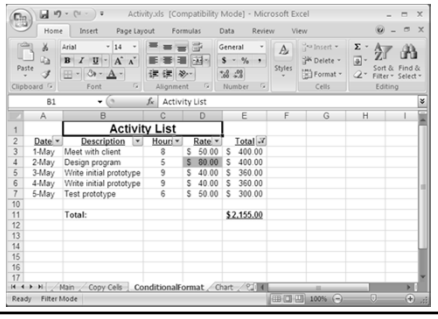
Filter on `Total` column: Custom filter with **`Total > 250`**






## Custom Filter Result

Filter on `Total` column: Custom filter result with **`Total > 250`**



## Conclusion

***Spreadsheets*** are programs for storing and manipulating data that is represented as a table of cells.

Each ***cell*** has a row number and column label which combine to represent its address. A cell can contain a number, text, date, or a formula that calculates its value.

Spreadsheets allow you to organize data and write formulas to do computations. They are a powerful tool for data storage and analysis.





## Objectives

- Define: spreadsheet
- Explain how cells are addressed in a spreadsheet.
- List some of the ways to select cells in a spreadsheet.
- Explain: filling
- Define and explain: formula
- Explain how an aggregate function works. List some examples.
- Explain the usefulness of charts.
- Define: conditional formatting
- Explain how spreadsheets can be used as a database.

# COSC 122
# Computer Fluency

## Databases

**Dr. Ramon Lawrence**
**University of British Columbia Okanagan**
**ramon.lawrence@ubc.ca**



## Key Points

1) Databases allow for easy storage and retrieval of large amounts of information.

2) Relational databases organize data into tables consisting of rows and columns.

3) SQL is the common language to query a database for results.





## What is a database?

A ***database*** is a collection of logically related data for a particular domain.

A ***database management system*** (***DBMS***) is software designed for the creation and management of databases.

- e.g. Oracle, DB2, Microsoft Access, MySQL, SQL Server

Bottom line: A ***database*** is the *data* stored and a ***database system*** is the *software* that manages the data.





## Databases in the Real-World

Databases are everywhere in the real-world even though you do not often interact with them directly.

- $20 billion dollar annual industry

Examples:

- Retailers manage their products and sales using a database.
  - ⇨ Wal-Mart has one of the largest databases in the world!
- Online web sites such as Amazon, eBay, and Expedia track orders, shipments, and customers using databases.
- The university maintains all your registration information and marks in a database that is accessible over the Internet.

Can you think of other examples?

What data do you have?





## DBMS

A database management system provides *efficient*, *convenient*, and *safe multi-user* storage and access to *massive* amounts of *persistent* data.

***Efficient*** - Able to handle large data sets and complex queries without searching all files and data items.

***Convenient*** - Easy to write queries to retrieve data.

***Safe*** - Protects data from system failures and hackers.

***Massive*** - Database sizes in gigabytes and terabytes.

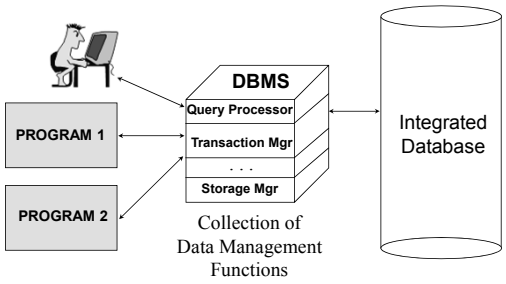
***Persistent*** - Data exists even if have a power failure.

***Multi-user*** - More than one user can access and update data at the same time while preserving consistency.





## Database System Approach



## Advanced: Databases and Abstraction

One of the major advantages of databases is they provide data abstraction. ***Data abstraction*** allows the implementation of an object to change without affecting programs that use the object through an external definition.

That is, as a database user or programmer, you do not have to worry about how the data is stored or organized.

A DBMS achieves data abstraction by allowing users to define the database and then handling all the low-level details of how to store it, retrieve it, and handle concurrent access to it.





## The Relational Model: Terminology

The ***relational model*** organizes database information into tables called relations.

- The relational model was developed by E. F. Codd in 1970 and is used by almost all commercial database systems.

Terminology:

A ***relation*** is a table with columns and rows.

An ***attribute*** is a named column of a relation.

A ***tuple*** is a row of a relation.

A ***domain*** is a set of allowable values for one or more attributes.

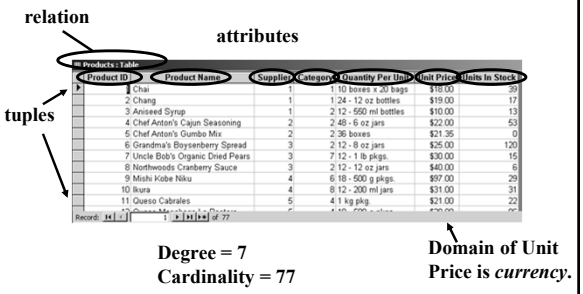
The ***degree*** of a relation is the number of attributes it contains.

The ***cardinality*** of a relation is the number of tuples it contains.





## Relation Example



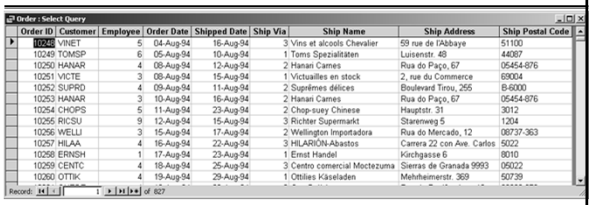

**Degree = 7**
**Cardinality = 77**

**Domain of Unit Price is *currency*.**





## Relation Practice Questions


Order : Select Query

| Order ID | Customer | Employee | Order Date | Shipped Date | Ship Via | Ship Name | Ship Address | Ship Postal Code |
|---|---|---|---|---|---|---|---|---|
| 10248 | VINET | 5 | 04-Aug-94 | 16-Aug-94 | 3 | Vins et alcools Chevalier | 59 rue de l'Abbaye | 51100 |
| 10249 | TOMSP | 6 | 05-Aug-94 | 10-Aug-94 | 1 | Toms Spezialitäten | Luisenstr. 48 | 44087 |
| 10250 | HANAR | 4 | 08-Aug-94 | 12-Aug-94 | 2 | Hanari Carnes | Rua do Paço, 67 | 05454-876 |
| 10251 | VICTE | 3 | 08-Aug-94 | 15-Aug-94 | 1 | Victuailles en stock | 2, rue du Commerce | 69004 |
| 10252 | SUPRD | 4 | 09-Aug-94 | 11-Aug-94 | 2 | Suprêmes délices | Boulevard Tirou, 255 | B-6000 |
| 10253 | HANAR | 3 | 10-Aug-94 | 16-Aug-94 | 2 | Hanari Carnes | Rua do Paço, 67 | 05454-876 |
| 10254 | CHOPS | 5 | 11-Aug-94 | 23-Aug-94 | 2 | Chop-suey Chinese | Hauptstr. 31 | 3012 |
| 10255 | RICSU | 9 | 12-Aug-94 | 15-Aug-94 | 3 | Richter Supermarkt | Starenweg 5 | 1204 |
| 10256 | WELLI | 3 | 15-Aug-94 | 17-Aug-94 | 2 | Wellington Importadora | Rua do Mercado, 12 | 08737-363 |
| 10257 | HILAA | 4 | 16-Aug-94 | 22-Aug-94 | 3 | HILARIÓN-Abastos | Carrera 22 con Ave. Carlos | 5022 |
| 10258 | ERNSH | 1 | 17-Aug-94 | 23-Aug-94 | 1 | Ernst Handel | Kirchgasse 6 | 8010 |
| 10259 | CENTC | 4 | 18-Aug-94 | 25-Aug-94 | 3 | Centro comercial Moctezuma | Sierras de Granada 9993 | 05022 |
| 10260 | OTTIK | 4 | 19-Aug-94 | 29-Aug-94 | 1 | Ottilies Käseladen | Mehrheimerstr. 369 | 50739 |

Record: 1 of 827

1) What is the name of the relation?

2) What is the cardinality of the relation?

3) What is the degree of the relation?

4) What is the domain of order date? What is the domain of order id?





## Databases
## Database and Database System

***Question:*** Which of these two definitions below are an example of software?

**A)** database

**B)** database system





## Databases
## Database Properties

***Question:*** True or False: The data in a database is lost when the power to the computer is turned off.

**A)** true

**B)** false

## Databases
## Database Properties (2)

***Question:*** True or False: More than one user can use the database managed by the DBMS at the same time.

**A)** true

**B)** false





## Databases
## Definition Matching

***Question:*** Given the three definitions, select the ordering that contains their related definitions.

**1)** relation
**2)** tuple
**3)** attribute

**A)** column, row, table
**B)** row, column, table
**C)** table, row, column
**D)** table, column, row





## Databases
## Cardinality and Degree

***Question:*** A database table has 10 rows and 5 columns. Select ***one*** true statement.

**A)** The table's degree is 50.

**B)** The table's cardinality is 5.

**C)** The table's degree is 10.

**D)** The table's cardinality is 10.





## Relational Keys

Keys are used to uniquely identify a tuple in a relation.

A ***superkey*** is a set of attributes that uniquely identifies a tuple in a relation.

A ***key*** is a *minimal* set of attributes that uniquely identifies a tuple in a relation.

Question:

- What is a key to identify a student in this class?





## Databases
## Keys and Superkeys

***Question:*** True or false: A key is always a superkey.

**A)** true

**B)** false





## Databases
## Keys and Superkeys (2)

***Question:*** True or false: It is possible to have more than one key for a table and the keys may have different numbers of attributes.

**A)** true

**B)** false

## Example Relations

Relations:

emp (eno, ename, bdate, title, salary, supereno, dno)
proj (pno, pname, budget, dno)
dept (dno, dname, mgreno)
workson (eno, pno, resp, hours)

`Emp` - one row per employee storing name, birth date, supervisor, and department that they are in

`Proj` - one row per project storing name and its department

`Dept` - one row per department storing name and manager

`WorksOn` - stores that an employee works on a particular project for a certain amount of time in a given role

Note: Key fields are underlined.





## Example Relation Instances

Emp Relation

| eno | ename | bdate | title | salary | supereno | dno |
|---|---|---|---|---|---|---|
| E1 | J. Doe | 01-05-75 | EE | 30000 | E2 | null |
| E2 | M. Smith | 06-04-66 | SA | 50000 | E5 | D3 |
| E3 | A. Lee | 07-05-66 | ME | 40000 | E7 | D2 |
| E4 | J. Miller | 09-01-50 | PR | 20000 | E6 | D3 |
| E5 | B. Casey | 12-25-71 | SA | 50000 | E8 | D3 |
| E6 | L. Chu | 11-30-65 | EE | 30000 | E7 | D2 |
| E7 | R. Davis | 09-08-77 | ME | 40000 | E8 | D1 |
| E8 | J. Jones | 10-11-72 | SA | 50000 | null | D1 |

WorksOn Relation

| eno | pno | resp | hours |
|---|---|---|---|
| E1 | P1 | Manager | 12 |
| E2 | P1 | Analyst | 24 |
| E2 | P2 | Analyst | 6 |
| E3 | P3 | Consultant | 10 |
| E3 | P4 | Engineer | 48 |
| E4 | P2 | Programmer | 18 |
| E5 | P2 | Manager | 24 |
| E6 | P4 | Manager | 48 |
| E7 | P3 | Engineer | 36 |

Proj Relation

| pno | pname | budget | dno |
|---|---|---|---|
| P1 | Instruments | 150000 | D1 |
| P2 | DB Develop | 135000 | D2 |
| P3 | Budget | 250000 | D3 |
| P4 | Maintenance | 310000 | D2 |
| P5 | CAD/CAM | 500000 | D2 |

Dept Relation

| dno | dname | mgreno |
|---|---|---|
| D1 | Management | E8 |
| D2 | Consulting | E7 |
| D3 | Accounting | E5 |
| D4 | Development | null |





## A Simple Query Language: Keyword Searching

***Keyword*** (or English-language) ***search*** allows a user to type keywords or phrases and returns a best answer estimate.


This works fairly well for web searches, although we lack precision. Precision is required for many applications.

- Example: How would you return all employees with salary greater than 30,000 using keyword search?





## SQL Overview

Structured Query Language or SQL is the standard database query language to retrieve *exact answers*.

- SQL is a ***declarative language*** (non-procedural). A SQL query specifies *what* to retrieve but not *how* to retrieve it.
- SQL is used by Microsoft Access.

Some basic rules for SQL statements:

- 1) There is a set of *reserved words* that cannot be used as names for database fields and tables.
  - ⇨ `SELECT`, `FROM`, `WHERE`, etc.
- 2) SQL is generally *case-insensitive*.
  - ⇨ Only exception is string constants. 'FRED' not the same as 'fred'.
- 3) SQL is *free-format* and white-space is ignored.





## SQL Queries

A query in SQL has the form:

**SELECT (list of attributes)**
**FROM (list of tables)**
**WHERE (filter *conditions*)**

Notes:

- 1) Separate the list of attributes and list of tables by **commas**.
- 2) The "`*`" is used to select all attributes.





## SQL Retrieving Only Some of the Columns

The ***projection operation*** creates a new table that has some of the columns of the input table. In SQL, provide the table in the `FROM` clause and the fields in the output in the `SELECT`.

Example: Return only the `eno` field from the `Emp` table:

```
SELECT eno
FROM   emp
```

Emp Relation

| eno | ename | bdate | title | salary | supereno | dno |
|---|---|---|---|---|---|---|
| E1 | J. Doe | 01-05-75 | EE | 30000 | E2 | null |
| E2 | M. Smith | 06-04-66 | SA | 50000 | E5 | D3 |
| E3 | A. Lee | 07-05-66 | ME | 40000 | E7 | D2 |
| E4 | J. Miller | 09-01-50 | PR | 20000 | E6 | D3 |
| E5 | B. Casey | 12-25-71 | SA | 50000 | E8 | D3 |
| E6 | L. Chu | 11-30-65 | EE | 30000 | E7 | D2 |
| E7 | R. Davis | 09-08-77 | ME | 40000 | E8 | D1 |
| E8 | J. Jones | 10-11-72 | SA | 50000 | null | D1 |

Result

| eno |
|---|
| E1 |
| E2 |
| E3 |
| E4 |
| E5 |
| E6 |
| E7 |
| E8 |

## SQL Projection Examples

Emp Relation

| eno | ename | title | salary |
|---|---|---|---|
| E1 | J. Doe | EE | 30000 |
| E2 | M. Smith | SA | 50000 |
| E3 | A. Lee | ME | 40000 |
| E4 | J. Miller | PR | 20000 |
| E5 | B. Casey | SA | 50000 |
| E6 | L. Chu | EE | 30000 |
| E7 | R. Davis | ME | 40000 |
| E8 | J. Jones | SA | 50000 |

```
SELECT eno,ename
FROM   emp
```

| eno | ename |
|---|---|
| E1 | J. Doe |
| E2 | M. Smith |
| E3 | A. Lee |
| E4 | J. Miller |
| E5 | B. Casey |
| E6 | L. Chu |
| E7 | R. Davis |
| E8 | J. Jones |

```
SELECT title
FROM   emp
```

| title |
|---|
| EE |
| SA |
| ME |
| PR |
| SA |
| EE |
| ME |
| SA |

Note: Duplicates are not removed during SQL projection.





## Databases Projection

***Question:*** Given this table and the query:

```
SELECT eno, ename, salary
FROM   emp
```

How many columns are returned?

**A)** 0
**B)** 1
**C)** 2
**D)** 3
**E)** 4

Emp Relation

| eno | ename | title | salary |
|---|---|---|---|
| E1 | J. Doe | EE | 30000 |
| E2 | M. Smith | SA | 50000 |
| E3 | A. Lee | ME | 40000 |
| E4 | J. Miller | PR | 20000 |
| E5 | B. Casey | SA | 50000 |
| E6 | L. Chu | EE | 30000 |
| E7 | R. Davis | ME | 40000 |
| E8 | J. Jones | SA | 50000 |





## Databases Projection (2)

***Question:*** Given this table and the query:

```
SELECT salary
FROM   emp
```

How many rows are returned?

**A)** 0
**B)** 2
**C)** 4
**D)** 8

Emp Relation

| eno | ename | title | salary |
|---|---|---|---|
| E1 | J. Doe | EE | 30000 |
| E2 | M. Smith | SA | 50000 |
| E3 | A. Lee | ME | 40000 |
| E4 | J. Miller | PR | 20000 |
| E5 | B. Casey | SA | 50000 |
| E6 | L. Chu | EE | 30000 |
| E7 | R. Davis | ME | 40000 |
| E8 | J. Jones | SA | 50000 |





## SQL Projection Questions

WorksOn Relation

| eno | pno | resp | dur |
|---|---|---|---|
| E1 | P1 | Manager | 12 |
| E2 | P1 | Analyst | 24 |
| E2 | P2 | Analyst | 6 |
| E3 | P3 | Consultant | 10 |
| E3 | P4 | Engineer | 48 |
| E4 | P2 | Programmer | 18 |
| E5 | P2 | Manager | 24 |
| E6 | P4 | Manager | 48 |
| E7 | P3 | Engineer | 36 |
| E7 | P5 | Engineer | 23 |
| E8 | P3 | Manager | 40 |

Write the SQL statement that:
1) Returns only attributes *resp* and *dur*.
2) Returns only *eno*.
3) Returns only *pno*.

List the number of result rows and columns in each case.





## One Table Query Example Retrieving Only Some of the Rows

The ***selection operation*** creates a new table with some of the rows of the input table. A condition specifies which rows are in the new table. The condition is similar to an `if` statement.

Example: Return the projects in department `'D2'`:

```
SELECT pno, pname, budget, dno
FROM   proj
WHERE  dno = 'D2';
```

Proj Relation

| pno | pname | budget | dno |
|---|---|---|---|
| P1 | Instruments | 150000 | D1 |
| P2 | DB Develop | 135000 | D2 |
| P3 | Budget | 250000 | D3 |
| P4 | Maintenance | 310000 | D2 |
| P5 | CAD/CAM | 500000 | D2 |

→

Result

| pno | pname | budget | dno |
|---|---|---|---|
| P2 | DB Develop | 135000 | D2 |
| P4 | Maintenance | 310000 | D2 |
| P5 | CAD/CAM | 500000 | D2 |

Algorithm: Scan each tuple and check if matches condition in `WHERE` clause.





## Retrieving Only Some of the Rows Selection Conditions

The condition in a selection statement specifies which rows are included. It has the general form of an if statement.

The condition may consist of attributes, constants, comparison operators (<, >, =, !=, <=, >=), and logical operators (`AND`, `OR`, `NOT`).

## SQL Selection Examples

Emp Relation

| eno | ename | title | salary |
|---|---|---|---|
| E1 | J. Doe | EE | 30000 |
| E2 | M. Smith | SA | 50000 |
| E3 | A. Lee | ME | 40000 |
| E4 | J. Miller | PR | 20000 |
| E5 | B. Casey | SA | 50000 |
| E6 | L. Chu | EE | 30000 |
| E7 | R. Davis | ME | 40000 |
| E8 | J. Jones | SA | 50000 |

```
SELECT eno, ename, title, salary
FROM   emp
WHERE  salary > 35000 OR
         title = 'PR'
```

| eno | ename | title | salary |
|---|---|---|---|
| E2 | M. Smith | SA | 50000 |
| E3 | A. Lee | ME | 40000 |
| E4 | J. Miller | PR | 20000 |
| E5 | B. Casey | SA | 50000 |
| E7 | R. Davis | ME | 40000 |
| E8 | J. Jones | SA | 50000 |

```
SELECT *
FROM   emp
WHERE  title = 'EE'
```

| eno | ename | title | salary |
|---|---|---|---|
| E1 | J. Doe | EE | 30000 |
| E6 | L. Chu | EE | 30000 |





## Databases Selection

***Question:*** Given this table and the query:

```
SELECT *
FROM   emp
WHERE  title='EE'
```

How many rows are returned?

**A)** 0
**B)** 1
**C)** 2
**D)** 3

Emp Relation

| eno | ename | title | salary |
|---|---|---|---|
| E1 | J. Doe | EE | 30000 |
| E2 | M. Smith | SA | 50000 |
| E3 | A. Lee | ME | 40000 |
| E4 | J. Miller | PR | 20000 |
| E5 | B. Casey | SA | 50000 |
| E6 | L. Chu | EE | 30000 |
| E7 | R. Davis | ME | 40000 |
| E8 | J. Jones | SA | 50000 |





## Databases Selection

***Question:*** Given this table and the query:

```
SELECT *
FROM   emp
WHERE  salary > 50000 or title='PR'
```

How many rows are returned?

**A)** 0
**B)** 1
**C)** 2
**D)** 3

Emp Relation

| eno | ename | title | salary |
|---|---|---|---|
| E1 | J. Doe | EE | 30000 |
| E2 | M. Smith | SA | 50000 |
| E3 | A. Lee | ME | 40000 |
| E4 | J. Miller | PR | 20000 |
| E5 | B. Casey | SA | 50000 |
| E6 | L. Chu | EE | 30000 |
| E7 | R. Davis | ME | 40000 |
| E8 | J. Jones | SA | 50000 |





## Databases Selection

***Question:*** Given this table and the query:

```
SELECT *
FROM   emp
WHERE  salary > 50000 or title='PR'
```

How many columns are returned?

**A)** 0
**B)** 2
**C)** 3
**D)** 4

Emp Relation

| eno | ename | title | salary |
|---|---|---|---|
| E1 | J. Doe | EE | 30000 |
| E2 | M. Smith | SA | 50000 |
| E3 | A. Lee | ME | 40000 |
| E4 | J. Miller | PR | 20000 |
| E5 | B. Casey | SA | 50000 |
| E6 | L. Chu | EE | 30000 |
| E7 | R. Davis | ME | 40000 |
| E8 | J. Jones | SA | 50000 |





## SQL Selection Questions

WorksOn Relation

| eno | pno | resp | dur |
|---|---|---|---|
| E1 | P1 | Manager | 12 |
| E2 | P1 | Analyst | 24 |
| E2 | P2 | Analyst | 6 |
| E3 | P3 | Consultant | 10 |
| E3 | P4 | Engineer | 48 |
| E4 | P2 | Programmer | 18 |
| E5 | P2 | Manager | 24 |
| E6 | P4 | Manager | 48 |
| E7 | P3 | Engineer | 36 |
| E7 | P5 | Engineer | 23 |
| E8 | P3 | Manager | 40 |

Write the SQL statement that:
1) Returns all rows with a project `P2`.
2) Returns all rows with responsibility of a `Manager`.
3) Returns all rows with a responsibility of `Manager` and duration of more than `40` months.

List the number of result rows for each case.





## One Table Query Example Retrieving Some of the Rows/Columns

Return the employee name and salary of all employees whose title is 'EE':

```
SELECT ename, salary
FROM   emp
WHERE  title = 'EE';
```

Emp Relation

| eno | ename | bdate | title | salary | supereno | dno |
|---|---|---|---|---|---|---|
| E1 | J. Doe | 01-05-75 | EE | 30000 | E2 | null |
| E2 | M. Smith | 06-04-66 | SA | 50000 | E5 | D3 |
| E3 | A. Lee | 07-05-66 | ME | 40000 | E7 | D2 |
| E4 | J. Miller | 09-01-50 | PR | 20000 | E6 | D3 |
| E5 | B. Casey | 12-25-71 | SA | 50000 | E8 | D3 |
| E6 | L. Chu | 11-30-65 | EE | 30000 | E7 | D2 |
| E7 | R. Davis | 09-08-77 | ME | 40000 | E8 | D1 |
| E8 | J. Jones | 10-11-72 | SA | 50000 | null | D1 |

Result

| ename | salary |
|---|---|
| J. Doe | 30000 |
| L. Chu | 30000 |

## One Table Query Examples

Return the birth date and salary of employee 'J. Doe':

```
SELECT bdate, salary
FROM   emp
WHERE  ename = 'J. Doe'
```

Return all information on all employees:

```
SELECT *          ← * returns all attributes
FROM   emp
```

Return the employee number, project number, and number of hours worked where the hours worked is > 50:

```
SELECT eno, pno, hours
FROM   workson
WHERE  hours > 50
```





## Databases
## Projection and Selection

***Question:*** Given this table and the query:

```
SELECT eno, salary
FROM   emp
WHERE  salary >= 40000
```

What is the degree of the result?

**A)** 2
**B)** 3
**C)** 4
**D)** 5

Emp Relation

| eno | ename | title | salary |
|---|---|---|---|
| E1 | J. Doe | EE | 30000 |
| E2 | M. Smith | SA | 50000 |
| E3 | A. Lee | ME | 40000 |
| E4 | J. Miller | PR | 20000 |
| E5 | B. Casey | SA | 50000 |
| E6 | L. Chu | EE | 30000 |
| E7 | R. Davis | ME | 40000 |
| E8 | J. Jones | SA | 50000 |





## Databases
## Projection and Selection (2)

***Question:*** Given this table and the query:

```
SELECT eno, salary
FROM   emp
WHERE  salary >= 40000
```

What is the cardinality of the result?

**A)** 2
**B)** 3
**C)** 4
**D)** 5

Emp Relation

| eno | ename | title | salary |
|---|---|---|---|
| E1 | J. Doe | EE | 30000 |
| E2 | M. Smith | SA | 50000 |
| E3 | A. Lee | ME | 40000 |
| E4 | J. Miller | PR | 20000 |
| E5 | B. Casey | SA | 50000 |
| E6 | L. Chu | EE | 30000 |
| E7 | R. Davis | ME | 40000 |
| E8 | J. Jones | SA | 50000 |





## SQL Projection/Selection
## One Table Questions

Relations:

emp (eno, ename, bdate, title, salary, supereno, dno)
proj (pno, pname, budget, dno)
dept (dno, dname, mgreno)
workson (eno, pno, resp, hours)

1) Returns all `employees` making more than $`50,000`.

2) Show the `WorksOn` records with less than `20` hours but more than `10` hours.

3) Return only the `pno` and `dno` for each `project`.

4) Return the name for each employee in department `'D1'`.

5) **Challenge:** Display the employees who make less than $`40,000` or have title `'EE'` and are born after June 1, 1970.

◆Dates are in `YYYY-MM-DD` format. e.g. `'1970-06-01'`





## Join

A join combines two tables into a single table.

If the join has no condition that specifies which rows are in the result, all possible combinations of rows are in the result.

This is called a ***Cartesian or cross product***.

◆If table *R* has *N* rows and *X* columns and table *S* has *M* rows and *Y* columns, then there are *N*M* rows and *X+Y* columns in the cross product result.

In SQL, a cross product is done automatically if you put more than one table in the `FROM` clause and do not specify a condition on how to combine them.

◆In most cases, this is **NOT** what you want to do!





## Cartesian Product
## SQL Example

*Emp* Relation

| eno | ename | title | salary |
|---|---|---|---|
| E1 | J. Doe | EE | 30000 |
| E2 | M. Smith | SA | 50000 |
| E3 | A. Lee | ME | 40000 |
| E4 | J. Miller | PR | 20000 |

*Proj* Relation

| pno | pname | budget |
|---|---|---|
| P1 | Instruments | 150000 |
| P2 | DB Develop | 135000 |
| P3 | CAD/CAM | 250000 |

```
SELECT *
FROM   emp, proj
```

| eno | ename | title | salary | pno | pname | budget |
|---|---|---|---|---|---|---|
| E1 | J. Doe | EE | 30000 | P1 | Instruments | 150000 |
| E2 | M. Smith | SA | 50000 | P1 | Instruments | 150000 |
| E3 | A. Lee | ME | 40000 | P1 | Instruments | 150000 |
| E4 | J. Miller | PR | 20000 | P1 | Instruments | 150000 |
| E1 | J. Doe | EE | 30000 | P2 | DB Develop | 135000 |
| E2 | M. Smith | SA | 50000 | P2 | DB Develop | 135000 |
| E3 | A. Lee | ME | 40000 | P2 | DB Develop | 135000 |
| E4 | J. Miller | PR | 20000 | P2 | DB Develop | 135000 |
| E1 | J. Doe | EE | 30000 | P3 | CAD/CAM | 250000 |
| E2 | M. Smith | SA | 50000 | P3 | CAD/CAM | 250000 |
| E3 | A. Lee | ME | 40000 | P3 | CAD/CAM | 250000 |
| E4 | J. Miller | PR | 20000 | P3 | CAD/CAM | 250000 |

## Databases
## Cartesian Product

***Question:*** R is a relation with 10 rows and 5 columns. S is a relation with 8 rows and 3 columns.

What is the degree and cardinality of the cartesian product?

**A)** degree = 8, cardinality = 80

**B)** degree = 80, cardinality = 8

**C)** degree = 15, cardinality = 80

**D)** degree = 8, cardinality = 18





## Equijoin

In most cases, you only want to combine two tables and have rows in the result that satisfy a certain condition.

The most common type of join is an ***equijoin*** that combines two tables by matching columns that have the same value.

- Equijoin gets its name because the columns are compared using the equality operator (=).
- e.g. `WorksOn.pno = Proj.pno`





## Equijoin Example

WorksOn Relation

| eno | pno | resp | dur |
|---|---|---|---|
| E1 | P1 | Manager | 12 |
| E2 | P1 | Analyst | 24 |
| E2 | P2 | Analyst | 6 |
| E3 | P4 | Engineer | 48 |
| E5 | P2 | Manager | 24 |
| E6 | P4 | Manager | 48 |
| E7 | P3 | Engineer | 36 |
| E7 | P4 | Engineer | 23 |

Proj Relation

| pno | pname | budget |
|---|---|---|
| P1 | Instruments | 150000 |
| P2 | DB Develop | 135000 |
| P3 | CAD/CAM | 250000 |
| P4 | Maintenance | 310000 |
| P5 | CAD/CAM | 500000 |

```
SELECT *
FROM   WorksOn, Proj
WHERE  WorksOn.pno = Proj.pno
```

| eno | pno | resp | dur | P.pno | pname | budget |
|---|---|---|---|---|---|---|
| E1 | P1 | Manager | 12 | P1 | Instruments | 150000 |
| E2 | P1 | Analyst | 24 | P1 | Instruments | 150000 |
| E2 | P2 | Analyst | 6 | P2 | DB Develop | 135000 |
| E3 | P4 | Engineer | 48 | P4 | Maintenance | 310000 |
| E5 | P2 | Manager | 24 | P2 | DB Develop | 135000 |
| E6 | P4 | Manager | 48 | P4 | Maintenance | 310000 |
| E7 | P3 | Engineer | 36 | P3 | CAD/CAM | 250000 |
| E7 | P4 | Engineer | 23 | P4 | Maintenance | 310000 |

What is the meaning of this join?





## Equijoin in SQL

There are two ways of using equijoin in SQL.

In `WHERE` clause:

```
SELECT *
FROM   WorksOn, Proj
WHERE  WorksOn.pno = Proj.pno
```

In `FROM` clause:

```
SELECT *
FROM   WorksOn JOIN Proj ON WorksOn.pno = Proj.pno
```

Can simplify syntax by using alias to shorten table name:

```
SELECT *
FROM   WorksOn AS W, Proj AS P
WHERE  W.pno = P.pno
```



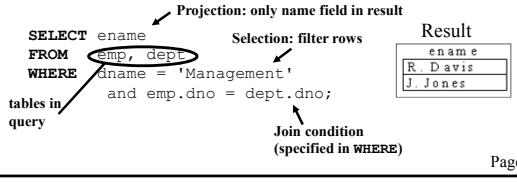


## Join Query with Selection Example

You can use join, selection, and projection in the same query.

- Recall: Projection returns columns listed in `SELECT`, selection filters out rows using condition in `WHERE`, and join combines tables in `FROM` using condition specified in `FROM` or `WHERE`.

Example: Return the employee names who are assigned to the 'Management' department.








## Join Query Examples

Return the department names and the projects in each department:

```
SELECT dname, pname
FROM   dept, proj
WHERE  dept.dno = proj.dno
```

Return the employees and the names of their department:

```
SELECT ename, dname
FROM   emp JOIN dept ON emp.dno=dept.dno
```

Return all projects who have an employee working on them whose title is 'EE':

```
SELECT pname
FROM   emp, proj, workson
WHERE  emp.title = 'EE' and workson.eno=emp.eno
        and workson.pno = proj.pno
```

## Join Practice Questions

Emp Relation

| eno | ename | title | salary |
|---|---|---|---|
| E1 | J. Doe | EE | 30000 |
| E2 | M. Smith | SA | 50000 |
| E3 | A. Lee | ME | 40000 |
| E4 | J. Miller | PR | 20000 |
| E5 | B. Casey | SA | 50000 |
| E6 | L. Chu | EE | 30000 |
| E7 | R. Davis | ME | 40000 |
| E8 | J. Jones | SA | 50000 |

WorksOn Relation

| eno | pno | resp | dur |
|---|---|---|---|
| E1 | P1 | Manager | 12 |
| E2 | P1 | Analyst | 24 |
| E2 | P2 | Analyst | 6 |
| E3 | P3 | Consultant | 10 |
| E3 | P4 | Engineer | 48 |
| E4 | P2 | Programmer | 18 |
| E5 | P2 | Manager | 24 |
| E6 | P4 | Manager | 48 |
| E7 | P3 | Engineer | 36 |
| E7 | P5 | Engineer | 23 |
| E8 | P3 | Manager | 40 |

Proj Relation

| pno | pname | budget |
|---|---|---|
| P1 | Instruments | 150000 |
| P2 | DB Develop | 135000 |
| P3 | CAD/CAM | 250000 |
| P4 | Maintenance | 310000 |
| P5 | CAD/CAM | 500000 |

Compute the following joins (how many tuples?):

```
1) SELECT * FROM Emp JOIN WorksOn
                ON Emp.eno = WorksOn.eno
2) SELECT * FROM Emp, Proj, WorksOn
   WHERE Emp.eno = WorksOn.eno AND
      Proj.pno = WorksOn.pno
```





## Ordering Result Data

The query result returned is not ordered on any column by default. We can order the data using the **ORDER BY** clause:

```
SELECT   ename, salary, bdate
FROM     emp
WHERE    salary > 30000
ORDER BY salary DESC, ename ASC;
```

- 'ASC' sorts the data in ascending order, and 'DESC' sorts it in descending order. The default is 'ASC'.
- The order of sorted attributes is significant. The first column specified is sorted on first, then the second column is used to break any ties, etc.





## More Advanced Querying

There are many more queries that we can ask a database:

- compute expressions and functions
- group data by value and meaning
- compute summary (aggregate) functions (max, min, sum, etc.)
- subqueries (queries within queries)

We will not study the notation for this advanced querying.





## Putting it All Together

The steps to write an English query in SQL are:

- 1) Find the columns that you need and put in SELECT clause.
- 2) List the tables that have the columns in the FROM clause. If there is more than one, join them together.
- 3) If you must filter rows, add a filter criteria in WHERE clause.

Example: List project name and budget where a 'Manager' is working on the project.

```
SELECT pname, budget
FROM   WorksOn, Proj
WHERE  resp='Manager' AND WorksOn.pno = Proj.pno
```

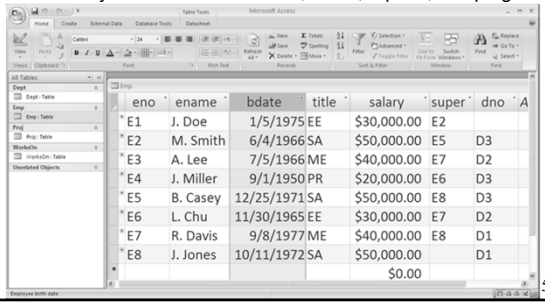




## Microsoft Access

Microsoft Access is a simple database management system.

- It allows you to create databases, forms, reports, and programs.

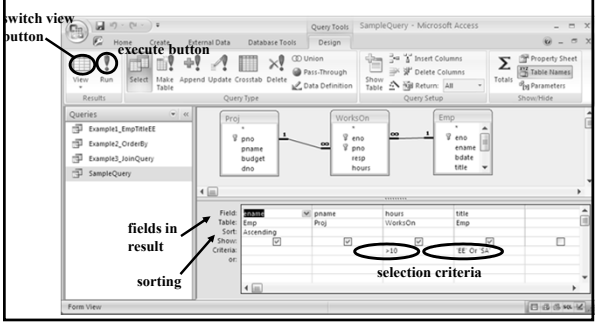







## Microsoft Access Query Interface

Tables are boxes. Relationships are lines. Condition specified on bottom.



## Microsoft Access Querying Basics

1) Projection is performed by selecting the fields in the output in the field row in the table at the bottom of the screen.

2) Selection is performed by entering the condition in the criteria box. The criteria applies to the field in that column.

3) The tables used are added to the query by the **`Show Table`**... option.

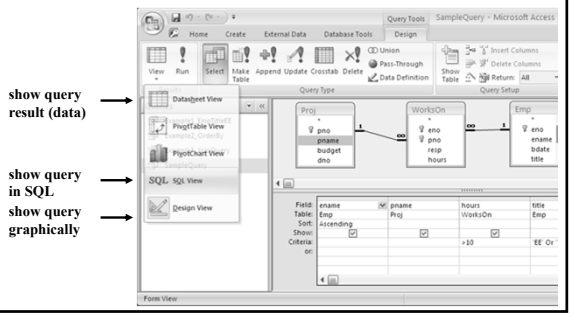
4) Joins (based on relationships) are often automatically added, but if not, you can add them by selecting the join field in one table, holding the mouse button, then dragging to the join field in the other table.





## Microsoft Access Query Views

You may view your data, your query graphically, or your query in SQL.






## Practice Questions

Relational database schema:

emp (eno, ename, bdate, title, salary, supereno, dno)
proj (pno, pname, budget, dno)
dept (dno, dname, mgreno)
workson (eno, pno, resp, hours)

1) Return the project names that have a budget > 250000.
2) List all project names in department with name 'Accounting'.
3) For employee 'M. Smith' list the project number and hours for all projects that he worked on.
4) Return a list of all department names, the names of the projects of that department, and the name of the manager of each department.





## Conclusion

A ***database*** is a collection of related data. A ***database system*** allows storing and querying a database.

The basic query operations are selection (subset of rows), projection (subset of columns), and join (combine two or more tables).

***SQL*** is the standard query language for databases, although Microsoft Access also provides a graphical user interface.





## Objectives

- Define: database, database system
- Explain how a DBMS achieves data abstraction.
- Define: relation, attribute, tuple, domain, degree, cardinality, superkey, key
- Given a relation, know its cardinality, degree, domains, and keys.

Given a relational schema and instance be able to translate very simple English queries into SQL.

# COSC 122
# Computer Fluency

## Social Implications of IT

**Dr. Ramon Lawrence**
**University of British Columbia Okanagan**
**ramon.lawrence@ubc.ca**



## Key Points

1) Information technology improves our lifestyle and our society, but also introduces challenges related to its ethical use and management.

2) We must be aware of potential violations of our privacy and our computer by malicious programs and companies.

3) Copyright protects intellectual property from unauthorized distribution and modification.





## Implications of Technology

Information technology, like any technological advance, can be used both for the benefit and the destruction of society.

As we become increasingly reliant on information technology, it is important that it be used appropriately and ethics guide its development and use.

As individuals, information technology is pervasive in our lives. Although this leads to new opportunities and experiences, we also must deal with the associated problems as well.





## IT and Communications

IT has allowed for real-time, inexpensive, readily-accessible communications across the globe.

The benefits of reliable communication are enormous - both personally and economically to our society.

However, communication tools can also be a time-consuming distraction that invades other aspects of our life. Further, we also must deal with limitations and challenges with the tools themselves.





## Crowdsourcing

***Crowdsourcing*** involves solving a problem or goal by using a large volunteer population.


Examples:

- Wikipedia http://www.wikipedia.org/
- Be a Martian (NASA) http://beamartian.jpl.nasa.gov/
- Foldit http://fold.it
- Freerice http://freerice.com
- Kickstarter http://www.kickstarter.com







## Email Benefits and Challenges

Email and instant/text messaging are the most commonly used electronic communication methods.

Text conversation introduces some issues:

- Conveying emotion - happy faces (*emoticons*) sometimes used
- Emphasis - hard to highlight what is important without other cues
- Conversational pace - asynchronous nature makes dialog hard
- Ambiguity - poor formatting causes misinterpretations
- Flame-a-thons - due to impersonal nature, harsh and inflammatory messages are easier to write.
- Spam - junk messages sent by automatic programs
- Size limits – condense dialog and introduce ambiguities
- History – can be kept forever and may be public

## Internet Etiquette

Internet etiquette are rules that civilized people use when communicating and interacting on the Internet that makes the interactions more personable, enjoyable, and acceptable.

- **Act as if you are there in person and that you were being recorded for everyone to see.**

Some email etiquette rules:

- Keep messages short and on a single topic.
- Always include context (question with your answer).
- Use an automated reply if unable to answer for a period of time.
- Answer a backlog of emails in reverse order.
- Get the sender's permission before forwarding email.
- Use targeted distribution lists (don't send that joke to everyone).
- Do not write in all capital letters.
- **Emails should still look "professional".**





## Dealing with the Uncivilized

As in any society, people do not follow the rules all the time.

The best policy is to show the respect and grace in a virtual world that you would in the real-world.

- No harassment, slander, rudeness, etc.
- Remember: **The virtual world is not anonymous.**

Dealing with spam email and companies is mostly out of your control. To avoid spam, limit how you give out your e-mail address and do not post it on a web site.

- Use real-world and technical savvy to detect scams and marketing. Watch for non-professional e-mails, strange e-mail addresses, etc. Be very careful with personal data.
- Aside: Why do spam e-mails have many spelling mistakes?
  - ⇨To avoid spam filters, recognizing common spam keywords.





## Phishing

***Phishing*** is the use of spam messages to trick users into supplying passwords and financial and personal data.

- Messages often report of security problems at your bank and direct to a bogus web site for data entry.

Notes:

- Never respond to requests for personal information over email. Legitimate businesses do not request information this way.
- Do not click on links or pre-typed addresses because they can be spoofed. Type the URL yourself.
- Check to make sure the web site is using encryption.
- Routinely review credit card and bank statements for unusual activity. Report suspected abuses to proper authorities.





## Malicious Threats Viruses and Worms

Viruses and worms are programs that are designed to negatively effect your computer. Often they are used to destroy software or steal personal data.

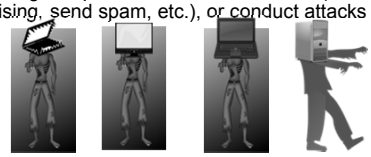
- A ***virus*** is a program that "infects" another program by embedding a copy of itself. When the infected program runs, the virus copies itself and infects other programs.
- A ***worm*** is an independent program that copies itself across network connections.
- A ***trojan*** is a program that hides inside another useful program, and performs secret operations.
  - ⇨May record keystrokes or other sensitive data or load malicious software.
- An ***exploit*** is a program that takes advantage of a security hole.
  - ⇨Backdoor access enters computer and reconfigures it for remote control.





## Zombies!!

Writing viruses and breaking into computer systems is a big business. Money can be made by stealing passwords and data, using computers to conduct activities (click on advertising, send spam, etc.), or conduct attacks on others.



**Zombies** – Infected, controlled computers

**Botnet** – Group of zombies working together

You are most likely to get bitten and infected by **social engineering** tricks rather than technical reasons!





## Preventing Viruses and Worms

Viruses must enter your computer through an entry point:

- infected email - with an attachment containing the program
- infected web site - that downloads malicious software
- infected program - loaded onto computer (from P2P, other users)

Although up-to-date anti-virus software offers protection, ***you*** are the ultimate line of defense.

Always evaluate what email you open, web sites you visit, and programs you install on your computer. Once you make the initial decision to allow access, it is impossible to know exactly what a program does on your computer.

## Intellectual Property

Intellectual property is any human creation like a photograph, music, textbooks, cartoons, etc.

Software is licensed in a form of leasing instead of buying. The license gives you the right to use the software personally, but not sell or give it away.

Shareware software allows you to download and try software for free, then pay the designer if you use it (honor system).

Ethics: It is very tempting to steal intellectual property (software, music, videos) on the Internet due to the availability of copying and distribution sites and tools.

- It is still ***STEALING***, even in a digital, virtual world.





## Copyright

A person automatically owns ***copyright*** of what he creates in the U.S., Canada, and most nations. Copyright applies to almost all artistic works (books, music, video, art, etc).

The copyright protects the owner's right to:

- Make a copy of the work
- Use a work as the basis for a new work (derivative work)
- Distribute or publish the work, including electronically
- Publicly perform and display the work

You are free to view or read anything on the Internet, but you need the copyright holder's permission to re-publish, modify, or re-distribute.





## Copyright and Free Use

The concept of ***Fair Use*** allows use of copyrighted material for educational or scholarly purposes, to allow limited quotation for review or criticism, and to permit parody.

Fair Use normally applies to distribution that is non-commercial.

There are large fines for violating copyright laws, especially for commercial purposes.

- Software is protected under the Software Copyright Act of 1980 in the United States.





## Software Development Ethics

Software developers should follow ethical standards in the development of their systems.

- Professional societies such as the Association of Computing Machinery (ACM) and the Institute of Electrical and Electronic Engineers (IEEE) have defined a code of ethics and professional practice.

Ethics are especially important for developers as many systems are ***safety-critical*** whose failure impacts society.

- Examples: control system in a nuclear reactor, communication networks (including the Internet), bank and financial systems

Such safety-critical systems use hardware redundancy, risk management techniques, and highly structured software engineering development methodologies.





## Survey
## Virus Writing

***Question:*** Would you write a destructive virus if you were absolutely sure you would not be caught?

**A)** yes

**B)** no

**C)** depends on the destructive effect





## Survey
## Intellectual Property - Software

***Question:*** It is acceptable to copy or use software obtained on the Internet without purchasing it …

**A)** never

**B)** sometimes depending on circumstances

**C)** always

## *Survey*
## *Intellectual Property - Music*

***Question:*** It is acceptable to copy or download music without paying …

**A)** never
**B)** sometimes depending on circumstances
**C)** always





## *Survey*
## *Intellectual Property - Frequency*

***Question:*** I have copied/downloaded music, movies, or software without paying …

**A)** never
**B)** in my lifetime
**C)** in the last year
**D)** in the last week
**E)** right now … during class





## *Survey*
## *Intellectual Property - Reasons*

***Question:*** My major reason for copying/downloading music/software/movies is:

**A)** I do not do it.
**B)** cost
**C)** rich media companies/entertainers
**D)** convenience
**E)** other





## *Open Discussion*

In small groups, discuss two questions:

1) What impacts (positive and negative) has technology had on your life?

2) What impacts (positive and negative) has technology had on society and our planet?





## *Conclusion*

IT benefits society in numerous ways, but requires ethical management and use similar to other technologies.

Malicious programs such as viruses and worms enter your computer through an email, web site, or infected program. Anti-virus software prevents some infection, but the computer user is the ultimate line of defense.

Ethics apply to the development of software to ensure that safety is considered when building software that may have negative effects if failures occur.

Copyright protects intellectual property, and applies on the Internet even with the existence of tools and sites that allows users convenient ways to steal digital data.





## *Objectives*

- List some issues with email.
- Define and give one example of netiquette.
- Define: phishing
- Define: virus, worm, trojan, exploit
- Explain the role of copyright for protecting intellectual property.
- Be able to discuss some benefits and issues with IT in your own personal life and society.

# COSC 122
# Computer Fluency

## How It Works

**Dr. Ramon Lawrence**
**University of British Columbia Okanagan**
**ramon.lawrence@ubc.ca**



## Key Points

1) Use our knowledge to understand how popular applications and systems work: Amazon, Facebook, Twitter, BitTorrent, iPhone.





## Amazon.com Overview

***Amazon.com*** is America's largest online retailer and sells books, DVDs, software, and other products.

- Headquartered in Seattle, Washington.
- Founded by Jeff Bezos in 1994.
- "Amazon" is named after the world's largest river. Since 2000, Amazon's logo has an arrow from A to Z, representing customer satisfaction (as it forms a smile).

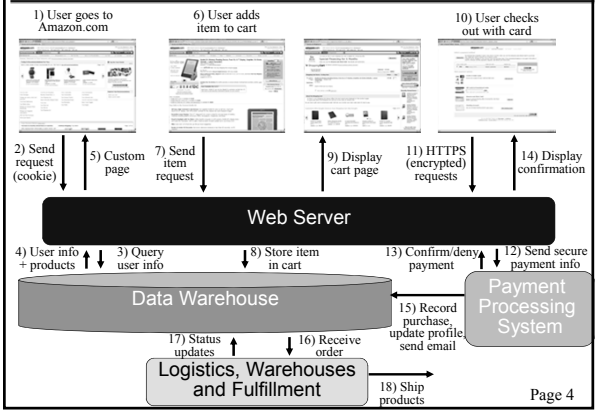
Amazon's Canadian site comes from the US, as it was legally prevented until March 2010 of operating any fulfillment centers in Canada. Products ship from Canada Post's Mississauga, ON.

Amazon provides technology and online hosting and services for many other retailers. Affiliates can sell through Amazon's system and link to Amazon's product database.





## Amazon.com – How it Works








## Facebook Overview

***Facebook*** is a social networking site with over 1.1 billion active users as of June 2013.

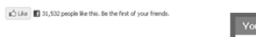
- Allows users to create personal profiles, add people as friends and send messages and updates to them.
- Founded by Mark Zuckerberg in 2004.
- Revenue ($5B+) from advertising (banner ads, news feed).
- Accessible directly or through applications on smartphones.
- 250+ billion user photos taking up more than 8 petabytes
  - ⇨ 350 million photos added each day (50 terabytes)
- 4.5 billion likes per day
- Facebook Platform is an API (application programming interface) allowing developers to write own applications.
  - ⇨ Currently more that 10 million applications with games being extremely popular.





## Facebook.com – Like Button

User adds Like button to page

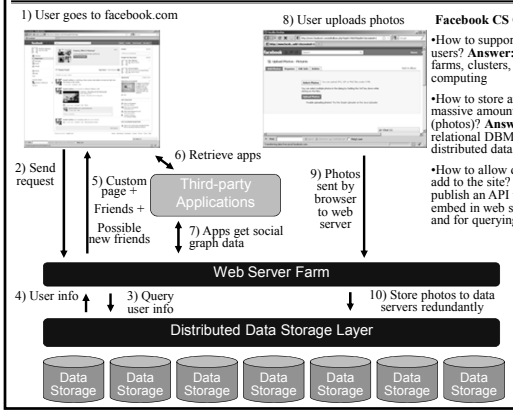



Added Code


When user clicks on Like request is sent to Facebook servers.

Build your own at:
http://developers.facebook.com/docs/reference/plugins/like

## Facebook.com – Applications




**Facebook CS Challenges:**

- How to support a billion users? **Answer:** server farms, clusters, cloud computing
- How to store and retrieve massive amounts of data (photos)? **Answer:** beyond relational DBMSs, distributed data stores
- How to allow developers to add to the site? **Answer:** publish an API with code to embed in web sites (FBML) and for querying (FQL)





## Facebook and Google Advertising

Facebook and Google make billions of dollars of revenue from advertising.

Facebook advertising is primarily banner advertising (display ads) and advertising in news feed. A company gets paid for banner advertising based on the number of displays ("impressions") and the number of user clicks ("click throughs").

Google advertising is primarily as sponsored results. Google gets paid each time a user clicks on a sponsored link.

Click through rates may be as low as 0.05% (Facebook) and the costs per click are on a bid system. Each click may only represent $0.10 to $0.50 of revenue.

Companies make money due to the billions of page views and clicks.





## Twitter Overview

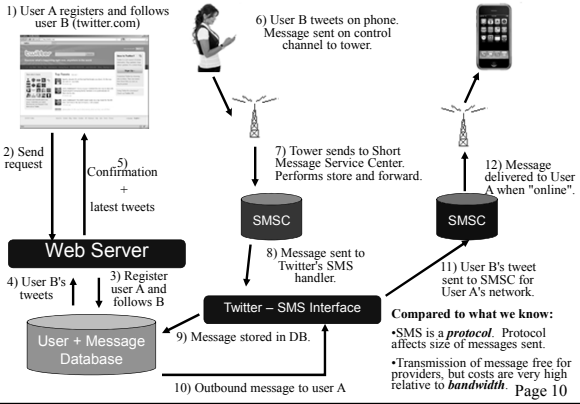
***Twitter*** is a social networking and blogging service that allows users to send and read user messages called tweets.

- ***Tweets*** are displayed on an user's page and can be up to 140 characters long (due to SMS compatibility).
- Users may subscribe (*followers*) to other user tweets.
- Tweets can be sent via the website, external applications (for smartphones/PCs), and the Short Message Service (SMS).
- Service is free but may be charged to use SMS or phone fees.
- Created in 2006 by Jack Dorsey.
- Currently has more than 500 million users and over 350 million tweets per day.



## Twitter – How it Works






**Compared to what we know:**

- SMS is a ***protocol***. Protocol affects size of messages sent.
- Transmission of message free for providers, but costs are very high relative to ***bandwidth***.





## BitTorrent Overview

***BitTorrent*** is a peer-to-peer file sharing protocol for data distribution. It is estimated to be the majority of Internet traffic.

**Basic idea:** Instead of downloading a large file from one source, the file is downloaded in pieces from many sources and re-assembled. This improves performance and reliability.

**How it works:**

- 1) A user creates a torrent descriptor file of the file to be shared. The file itself is put on a BitTorrent "seed" node and divided into pieces.
- 2) Another user downloads the torrent descriptor file and begins to download the file pieces. It may acquire pieces from other peers that had previously downloaded the file. Once a peer has the complete file, it can function as a seed.





## iPhone Overview

The ***iPhone*** is a ***smartphone*** manufactured by Apple that supports voice, text, browsing, email, and Wi-Fi. Distinctive features include its multi-touch screen, virtual keyboard, and thousands of third-party applications ("apps").

Smartphones are mini-computers that have an operating system capable of running programs both within and outside of a web browser.

- A major battle for market share between operating systems: Android, iPhone, Microsoft, Blackberry.

These devices are chosen more for their program capabilities and user interface features than phone service provider plans.

## *iPhone How it Works – Apps*

1) An iPhone application is built by a developer in the Objective-C programming language and compiled into a binary.

- Each smartphone platform supports a different language: RIM/Android – Java

2) The application is verified by Apple, and if it passes, is loaded onto the App store.

3) Users search the store for applications and download and run the binary file on their device. An App runs on the device directly rather than in the browser.

**What we have learned:**

- Basic programming skills (can be extended to develop apps)
  - ⇨ By 3rd year CS (or time on your own), you could do it.
- Hardware components and how computer works/run programs
- Components of applications and user interfaces





## *Conclusion*

We have investigated how some of the most popular systems and applications work. Each system requires ***creativity*** and a significant software ***engineering effort*** to design and build it.

We saw how the concepts ***we have learned*** in programming, computer systems, and networking/Internet are used in these systems and the research/technical challenges being faced.

Operational systems are continually improved, fixed for errors, and must remain working all the time. It takes considerable resources and people to operate.

The popular systems typically started from basic ideas and were expanded over time. It has never been easier to create a system and scale it up to millions of users.





## *Objectives*

- Understand some of the ideas behind common applications and systems and how it relates to the concepts discussed in the course.

# COSC 122
# Computer Fluency

## Representing Images and Sound

**Dr. Ramon Lawrence**
**University of British Columbia Okanagan**
**ramon.lawrence@ubc.ca**



## Key Points

1) It is possible to digitize the naturally analog information of sound, images, and video.

2) Due to the large size of digitized images/video, compression is needed to make it more efficient to use and store the information.





## Overview

Most of the information in the real world is not digital by nature.

Although we saw some reasonable encodings for numbers and characters, it is a little more complex to store images and sounds on a computer.

Images and sound are analog by nature. To convert to digital, we must sample the original, encode it, and then compress it to make it usable.

The increasing power of computers has made the virtual reality that can be produced more and more realistic.





## Review: Digitizing Color

Recall that computers represent different colors by giving the intensities of red, green, and blue (RGB).

- Each red, green, and blue value was a number from 0 to 255 that we can represent in decimal or hexadecimal.
- Black is no color (all values are 0).
- White is full intensity of RGB (all values are 255).





## Color Question

***Question:*** What is the best description of color code: #B3009F?

**A)** a shade of purple
**B)** a shade of yellow
**C)** a shade of blue
**D)** a shade of green





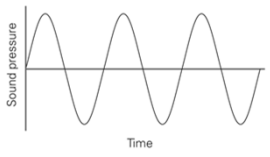
## Digitizing Sound

An object creates sound by vibrating in a medium such as air.

Vibrations push the air and pressure waves emanate from the object and vibrate our eardrums.

The force, or ***intensity*** of the push determines the volume.

The ***frequency*** (number of waves per second) is the pitch.



## Converting Analog Sound to Digital

Sound is analog (continuous) by nature.

To digitize sound into bits, we need to record with a binary number the amount by which the wave is above or below the zero line. (positive or negative sound pressure)

However, we cannot possibly record a value at every point in time, so we use ***sampling*** to collect information at certain intervals (points in time).

The ***sampling rate*** is the number of measurements per second. The higher the rate, the more accurate the digitization (but more space is required).





## How Fast a Sampling Rate?

Sampling rate should be related to the wave's frequency.

- Too slow a rate could allow waves to fit between the samples causing us to miss segments of sound.

Guideline is **Nyquist Rule**: Sampling rate must be at least twice as fast as the fastest frequency.

Human perception can hear sound up to 20,000 Hz, so 40,000 Hz sampling rate is enough.

- The standard for digital audio is 44,100 Hz.





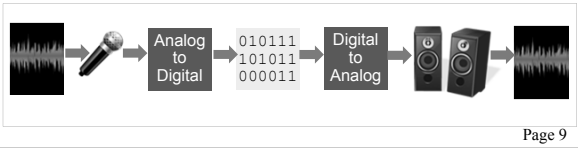
## Analog-to-Digital Converters

An ***analog-to-digital converter*** (ADC) samples a signal at regular intervals and outputs binary numbers into memory.

- Microphone is an example of an ADC.

To play sound, ***digital-to-analog converters*** (DAC) are used that receive binary numbers as input and output an electrical wave by filling in between the digital values.

- The electrical signal is output to a speaker which converts it to a sound wave.








## How Many Bits per Sample?

How accurate must the samples be?

- Bits must represent both positive and negative values.
- The more bits, the more accurate the measurement.
- The digital representation of audio CDs uses 16 bits (records 65,536 levels, half above and half below the zero line).





## Advantages of Digital Sound Representation

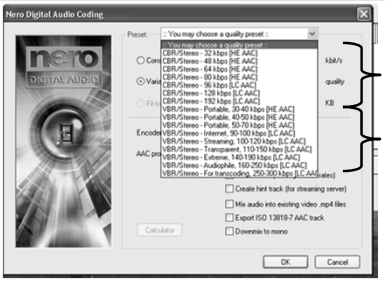
The advantages of digital representation:

- 1) All digital representations can be computed on (manipulated digitally). This makes it easier to edit and change them.
- 2) Reproducing the data can be done exactly.
  - ⇨ Bit file can be copied without losing any information.
  - ⇨ Original and copy are exactly the same.
- 3) Compression - Compression techniques such as (MP3 compression) allow for more compact representation.
  - ⇨ Remove waves that are outside range of human hearing.
  - ⇨ MP3 usually gets a compression rate of 10:1.
  - ⇨ MP3 stands for MPEG level 3 ("sound track" of MPEG digital video).





## Example: Digitizing CDs (CD ripping)




Quality is expressed in kbps. (kilo bits per second).
- If sampling rate is 44.1 kHz, what is the kbps for CD audio (16-bit samples)?
- For 192 kbps, what is the sample size? Does that sound right?

## Digital Compression

***Question:*** A music digitization program provides the two encodings below. Which encoding has the largest size (assuming no compression)?

**A)** Sample at 50 kHz and encode 16 bits per sample
**B)** Sample at 10 kHz and encode 32 bits per sample





## Digital Quality

***Question:*** A music digitization program provides the two encodings below. Which encoding has the best sound quality?

**A)** Sample at 50 kHz and encode 16 bits per sample
**B)** Sample at 10 kHz and encode 128 bits per sample





## Digitizing Images and Video

Just like sound, images and video is encoded by sampling. For images, a sample consists of how many measurements are taken over an area.

- For instance, when scanning, you can determine how many pixels (samples) per inch you will take.
- The more samples (higher pixels per inch) the finer detail the image will be encoded.

Movies are sequences of individual images.





## Two Ways to Encode Digital Images

**Bitmap representation** stores a 2D matrix (width x height) with a color intensity at each pixel.

- Examples: PNG, JPEG, GIF, TIFF

**Vector representation** describes an image as a sequence of lines or shapes each with a color.

- Examples: fonts (as scale better), Postscript (eps), PDF, Scalable Vector Graphics (svg)

Bitmap format is good for complex images and can be compressed. However, it does not scale well.

Vector representation is good for line art and text and will have smaller sizes for those types of images.





## ☆Digitizing Images and Video Compression

Without compression, storing images would be impractical.

Compression may be ***lossless*** (no information is lost) or ***lossy*** (information may be lost during compression).

**JPEG** compression can compress images.

- JPEG is a lossy compression scheme that makes images much smaller and the picture quality is controllable.
- Since our eyes are not very sensitive to small changes in hue, (but are sensitive to small changes in brightness), stores a less accurate description of hue (fewer pixels).
- Gets a 20:1 compression ratio without eyes being able to perceive the difference.
- The actual compression algorithm is beyond our scope.

***PNG*** is a lossless compression method.

- Best for text and line art.





## JPEG Example

Full quality – 83,261 bytes (2.6:1)

Average quality – 15,138 bytes (15:1)

Medium quality – 9,553 bytes (23:1)

Low quality – 4,787 bytes (46:1)

Source: Wikipedia - http://en.wikipedia.org/wiki/JPEG

## Aside: JPEG in Digital Cameras

Most digital cameras use some form of JPEG compression and often provide you with a setting that indicates image quality.

Smaller images (and thus more images on the camera) come at the cost of lower quality. Probably better to select high quality!

Example:

- Nikon D5000 – 12.3 megapixel sensor (4288 x 2848)
- RAW format is 12-bits per pixel: 18.45 MB (uncompressed) and 10.6 MB (compressed)
- JPEG: high quality: 5.9 MB, medium: 3.3 MB, low: 1.5 MB
- Source: Nikon





## MPEG Compression Scheme

The **MPEG** compression scheme follows the same idea as JPEG, but is applied to motion pictures.

Two "levels" of compression:

- 1) JPEG-like compression is applied to each frame.
- 2) Then "**interframe coherency**" is used so that only record and transmit the differences between one frame and the next. This results in huge amounts of compression.





## Digital Compression

***Question:*** True or false: An MP3 performs lossy compression.

**A)** true

**B)** false





## Optical Character Recognition

***Optical Character Recognition*** (OCR) is the process of analyzing captured images to determine its contents.

- Example: Scan in a document and have the computer convert the document into a text document rather than an image.

OCR is used in other areas:

- auto-mail sorters
- photo radar cameras (license plate numbers)
- handwriting recognition
- fingerprinting technology (biometrics - eye, fingerprint, etc.)

***Captchas*** used on web sites to stop automated programs are based on the idea that humans are better at recognizing image patterns that computer algorithms.








## Virtual Reality: Fooling the Senses

Input and output devices can use all senses to engage the user in the virtual reality experience.

- Sound and sight we have seen already.
- Smells - it has been done, but not well.
- Taste - not really..

Touch has been increasingly used to communicate realism. Examples including vibrating controllers and interactive devices that provide motion and vibration that mimics real world cues.

- These ***haptic devices*** engage our sense of touch.





## Advanced: The Challenges of Bandwidth and Latency

Although images, sound, and video are represented digitally, two issues challenge the construction of a virtual reality.

- ***Latency*** is the time it takes for information to be delivered.
  - ⇨ Too long a latency period ruins the illusion as we can sense the delay.
  - ⇨ Absolute limit to how fast information can be transmitted—speed of light.
- ***Bandwidth*** is the rate at which information can be delivered.
  - ⇨ Bandwidth is important as digital encodings, even with compression, consume a lot of space.

## Conclusion

Sounds, images, and video are digitally encoding by ***sampling*** the analog input and encoding each sample in bits.

The raw samples consume significant amounts of space, so they are ***compressed*** to make them faster to process and smaller to store.

Although increasing computer power has made virtual reality more realistic, continuing work is performed on compression and techniques to improve ***bandwidth*** and reduce ***latency***.





## Objectives

- Define: intensity, frequency
- Define: sampling, sampling rate
- Define: Nyquist Rule
- Explain the purpose of analog-to-digital and digital-to-analog converters.
- List two advantages of digital sound.
- Compare and contrast: lossy and lossless compression
- Define: JPEG, MPEG, haptic device, OCR
- Compare the difference between representing images using bitmaps or vectors.
- Define: bandwidth, latency

# COSC 122
# Computer Fluency

## Security

**Dr. Ramon Lawrence**
**University of British Columbia Okanagan**
**ramon.lawrence@ubc.ca**



## Key Points

1) Privacy involves ensuring personal information is used and distributed according to a person's wishes.

2) Security encompasses the various ways for ensuring privacy and protecting digital data.

3) Security includes user identification, access privileges, and protocols and encryption.

4) Encryption encodes text so that only the intended receiver can understand it.





## Privacy

***Privacy*** is the right of people to choose freely under what circumstances and to what extent they will reveal themselves, their attitude, and their behavior to others.

Information technology threatens privacy due to the ease of storing, copying, and exchanging digital information that is collected from a variety of sources (government, business, etc.).

As users of services, we are often forced or must "voluntary disclose", private information that we trust the organizations will keep secure and not distribute.

Although there are numerous rules and regulations for privacy, they are not consistent across all countries and cannot always be rigorously enforced.





## Privacy: Whose Information Is It?

An interesting question about privacy relates to who "owns" information in a transaction or exchange.

For instance, when you buy groceries, does the grocery store have the right to the information about what you purchased?

- This is valuable information to the merchant as they can spot trends that help in marketing and inventory management.

Beware: If you have any sort of membership card, your purchase information can be maintained across visits to get a profile of your purchases.

- Merchants can also use your credit or debit card information.
- Most organizations now must disclose how they will use the information and give you the right to "opt out".





## A Privacy Success Story
## So Long Tele-marketers!

**Before:** The telemarketing industry's "self-policing" mechanism required individuals to write a letter or make an on-line payment to stop telemarketing calls. Individuals received numerous, unwanted calls.

**Solution:** The United States government set up the Do-Not-Call List. Anyone on the list cannot be called by a tele-marketer without incurring a fine.

**Result:** There are over 80,000,000 households on the list and the telemarketing industry has largely collapsed.

**In Canada:** The government has passed legislation creating a Do-Not-Call list similar to the United States.





## Privacy on the Internet

The Internet is ***not*** an anonymous communication system.

- User ids, cookies, and IP addresses can be used to track communications and interactions.
- Any interaction with a web site can be logged and recorded.
- Email travels (and may be logged) by numerous servers.

As with real-world communications, privacy can only be guarded with adequate security and knowledge.

The complexity of computer applications and systems makes it much harder to understand risks to privacy.

You must assume that anything you do on the web will become public even "trending".

## Privacy Breaker: The Cookie

A ***cookie*** is a small file stored on your computer by your browser by a web site that you visit.

A cookie file allows a site to identify you between visits by storing information such as your user id.

Cookies can be abused by advertisers who store them on your computer whenever you visit a site they have ads on. They can then use the user id in your cookie to detect when you visit other sites that they provide advertising for.

Browsers now give you the option of disabling cookies on a per site, individual request, or overall basis.





## Your Digital Footprints

Your Internet activities are recorded in a variety of places which results in a large digital footprint:

- Browsers store: Browsing history and cache, form data, cookies, etc. Learn how to delete them or use Google Incognito mode or Anonymizer.
- ISP stores: Some traffic information, bandwidth usage, potentially logs of sites visited
- Cellphone companies store: History of calls, cell phone towers used, call detail records, text message content/detail, and IP information. Some of this information is stored for over a year and is available without a warrant.

Learn how you create digital footprints and avoid being caught in the act, ending your relationship, etc.!





## Identity Theft

***Identity theft*** is the crime of posing as someone else for fraudulent purposes.

It is too easy to get personal information for others:

- from spam email or bogus web sites
- from security breaches in registered databases
- from accidental release on the Internet
- from paper records including discarded documents

Identity theft is a growing problem because most financial transactions are entirely automatic. Once you have the key identifying fields for a person, a system assumes you are that person and no manual verification is performed.





## Protecting Your Identity

It may sound paranoid, but in today's digital society, your identity is your most important asset and must be protected:

- Ensure your computer security including anti-virus and software is up-to-date.
- Only use trusted software, email, and web services.
- Be wary of scams that are "Too good to be true!"
- Chose strong passwords and keep them safe.
- Shred documents that contain personal and financial data.
- Do not trust an organization or person unless you have evidence that you should do so.





## Identify Theft

***Question:*** Do you know any one who has been a victim of identify theft?

**A)** I have been a victim

**B)** A member of my family has been a victim.

**C)** A friend has been a victim.

**D)** Someone I know has been a victim.

**E)** I do not know someone who has been a victim of identify theft.





## Security

***Security*** is the act of keeping precious data safe and only accessible to the correct people.

- Security is a way of enforcing privacy in digital systems.

There are many different security technologies. In general, security involves several things:

- ***User identification*** - verify system user is who they say they are
- ***Access privileges*** - only allow user to access data they have the privilege (or right) to access or update.
- ***Security or encryption protocol*** - stores or transmits data in such a way that only users with the correct access privileges can use it.

## User Identification

A system performs ***user identification*** to determine if the user is who they claim to be.

The most common form of user identification is a user id and a password. The user id may be user chosen or system assigned. The password is chosen by the user and is private.

Other technologies for user identification:

- biometrics - finger printing, voice recognition, eye scans
- digital access cards and keys

The ***authentication system*** is used to verify the user id and password is correct.





## Creating Good Passwords

Your password is your only form of defense against other users accessing your data and private information.

- It is crucial to select a good one because there are techniques to "crack" passwords, especially weak ones.

Cracking passwords:

- Directed guessing - use common words, names, birth dates, and other information known about the user.
- Brute force - try all possible character sequences to find the password (usually limited by denying access after a while)

Good passwords have at least 6 characters with a mixture of upper and lower case letters, numbers, and punctuation.

- It should not contain components of dictionary words or personal information.





## Changing Passwords

Passwords should be changed periodically.

Although managing passwords for many different systems is cumbersome, using a single password for everything is risky.

A good idea is to recycle passwords by rotating through a few or making slight changes to existing ones.

Question: Why can the administrator not tell me my forgotten password?

- Answer: Passwords are encrypted when stored on the computer to prevent the administrator (and others) from knowing it. Administrators are only allowed to reset a password.





## Password

***Question:*** I have at least one bad password (a name, a birth date) for an important computer system that I use.

**A)** Yes

**B)** No





## Access Privileges

***Access privileges*** limit access to data and software functions based on the rights assigned to the user.

The ***access control system*** verifies a user has access to the given resource before allowing them to use it.

On shared machines, your user id provides you access to some files and programs. However, you cannot typically access the files and directories of other users unless they allow you to.

Three common access privileges:

- ***read*** - can read file contents
- ***write*** - can update file contents or delete entire file
- ***execute*** - can run a program or enter a directory

These access privileges may be specified on a per user basis, to ***groups*** of users, or to all users (public).





## Encryption And Decryption Terminology

An ***encryption system (protocol)*** converts data into a form that cannot be understood by anyone but the intended user.

- ***Encryption*** transforms a data representation so it is no longer understandable to users without the decryption key.
- ***Decryption*** converts an encrypted data representation into its original form, usually using a key or private information.

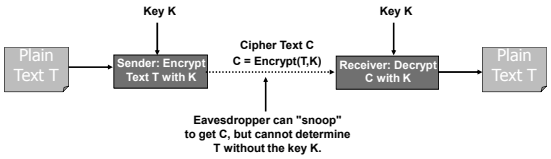
***Cleartext*** or ***plaintext*** is the information before encryption.

***Cipher text*** is the information in encrypted form.

A ***cryptosystem*** is a combination of encryption and decryption methods.

A ***one-way cipher*** is an encryption system that cannot be easily reversed (used for passwords).

## Cryptosystem Diagram Sender to Receiver








## Caesar Cipher

The ***Caesar cipher*** was used by Julius Caesar to encrypt messages sent to his generals.

- Encryption Algorithm: Shift each letter over 3 places, wrapping around to the start of the alphabet as necessary.
- Decryption Algorithm: Go back 3 letter places in the alphabet, wrapping as necessary.

Example:

- `Plain text  =` **`ABCDEFGHIJKLMNOPQRSTUVWXYZ`**
- `Cipher text =` **`DEFGHIJKLMNOPQRSTUVWXYZABC`**

Example:

- Plain text = `HELLO WORLD.`
- Cipher text = `KHOOR ZRUOG.`

Question: Pick a partner and exchange a short encrypted message (then decrypt).





## Security Caesar Cipher

***Question:*** Decrypt the following Caesar cipher message:

**SLFN D!**

**A)** PICK A!

**B)** VOIQ G!

**C)** PICK G!

**D)** PICK D!

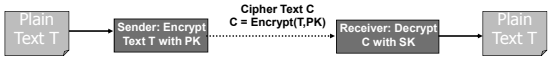




## Public Key Cryptosystems

Public key cryptosystems use two keys: one public (PK) and one private (SK). The keys are designed so that senders can send a message encrypted using the public key and only the receiver (who made the keys) can decrypt the message.

Diagram:








## RSA Public Key Cryptosystem Selecting a Key

The RSA public key cryptosystem relies on prime numbers.

Any number can be factored into primes in only one way.

A key is chosen with special properties:

- Must be the product of two different prime numbers p and q.
- p and q must be about 64 or 65 digits long to produce a 129-digit public key.
- p and q must also be 2 greater than a multiple of 3.

If *p* and *q* are kept secret, the code cannot be cracked.

- If the key is large enough, factoring to find *p* and *q* can't be done in any reasonable amount of time even by software.





## Strong Encryption Techniques

A communicating party can use the technology to protect their communication so no one else can read it.

Government agencies would like this technology kept out of the hands of "bad guys" and install backdoors (or trapdoors) to allow the government to crack encryption.

## System Backup

A ***system backup*** is a copy of valuable data and software that is used to restore a failed system.

Performing regular system backups is important, even for personal data, that may get lost due to system and natural disasters.

Mission-critical data is frequently backed up to multiple different sites to handle major natural disasters.

***System redundancy*** is a good thing to insure the system continues to operate properly. Redundancy can be in the form of software backups or hardware components (multiple drives).





## Backing Up a Personal Computer

What to backup:

- All personal data including documents, pictures, and music.
- Software settings such as Internet favorites.
- Do not backup operating system or programs as they can be re-installed from source CDs.

How to backup:

- Simple: Use a duplicate device such as a USB key or extra hard drive and copy files to it periodically.
- Offsite: Burn a CD or DVD with files and store in another place.
- Online: Use cloud services (DropBox, Google).
- Sophisticated: Install and configure backup software that regularly saves data to another drive or CD/DVD.





## Backup

***Question:*** The last time I backed up the important files on my computer or laptop was…

**A)** Last week
**B)** Last month
**C)** Last semester
**D)** Last year
**E)** Never … do you mean the computer can lose my files?





## Conclusion

Preserving our ***privacy*** is especially important in our digital world because of the amount of information collected and the simplicity that it can be exchanged.

Security protocols and systems are designed to restrict access to systems and data to the appropriate individuals.

- Security involves user identification (authentication system), access privileges (access control system), and encryption.
- We must use good passwords to protect our privacy.

Various encryption protocols provide data security. RSA public encryption is a strong encryption scheme.

We must backup our data and system in addition to securing it.





## Objectives

- Discuss some issues with maintaining privacy in a digital world.
- Define cookie and explain how it can invade your privacy.
- Define identity theft and list some precautions to avoid it.
- Define security and list three components of security.
- Define: user identification, access privilege, authentication system, access control system
- Define: encryption system, encrypt, decrypt, plain text, cipher text, cryptosystem, one-way cipher
- Draw a diagram and explain how encryption/decryption works.
- Be able to encode and decode a Caesar cipher.
- Explain the key idea between public (RSA) key encryption.
- Define: system backup, redundancy

# COSC 122
# Computer Fluency

## Limits of Computation

**Dr. Ramon Lawrence**
**University of British Columbia Okanagan**
**ramon.lawrence@ubc.ca**



## Key Points

1) Computers can demonstrate "artificial intelligence" but cannot yet mimic human creativity.

2) Game trees and search strategies are used to create the intelligence in games.

3) Scientists use big-Oh notation to analyze and compare the performance of algorithms.

4) There exists some problems where there is no efficient solution or no solution at all.





## Do Computers have Limits?

We have seen that the computer gets it power by being able to perform simple operations very fast.

By combining these simple operations into larger software programs, a computer can perform complex tasks.

Two interesting questions:

- 1) Can computers do anything (everything)?
- 2) Can computers behave like humans?





## Can Computers Think?

Alan Turing posed the ***Turing Test*** to evaluate if a computer can mimic a human. The Turing Test:

- To identical rooms labeled A and B are connected electronically to a judge who can type questions directed to the occupant of either room. A human being occupies one room, and the other contains a computer. The judge's goal is to decide, based on the questions asked and the answers received, which room contains the computer. If after a reasonable period of time the judge cannot decide for certain, the computer can be said to be intelligent.
- The computer is intelligent if it acts enough like a human to deceive the judge.

The test does not define thinking, intelligence, awareness or focus on any specific ability.





## Can Computers Think? (2)

Computers have come closer to passing the test but not quite.

Computer advances:

- Better understanding and parsing of natural language (ELIZA)
- Recognize semantics in language and communication
- Translate to and from natural language realistically
- Improved computational power

Work to do:

- Computational power will go beyond that of the human brain in 50 years.
- Outstanding challenge is modeling complexity and intelligence in software.





## Acting Intelligently?

Eliza is a system developed by MIT researcher Joseph Weizenbaum to carry on a conversation as though she were a psychotherapist.

- The software used word clues for answers but did not understand the meaning so it was not intelligent.

Example:

```
User: I'm depressed.
Doctor: Why are you depressed?
User: My mother is not speaking to me.
Doctor: Tell me about your mother.
User: She doesn't want me to major in engineering.
Doctor: No?
User: No, she wants me to go into medicine.
```

## *Survey*
## *Computer Intelligence*

***Question:*** I believe a computer will behave like a human …

**A)** never
**B)** in my lifetime
**C)** within 50 years
**D)** within 20 years
**E)** within 10 years





## *Survey*
## *Computers and Humanity*

***Question:*** It is a good thing if computers/robots become as intelligent as humans and develop/display emotions.

**A)** Strongly Agree
**B)** Agree
**C)** Neutral
**D)** Disagree
**E)** Strongly Disagree





## *Artificial Intelligence*

***Artificial intelligence*** (***AI***) refers to the ability of a computer to mimic human intelligence in certain situations.

To exhibit intelligence, the computer has to "understand" a complex situation and reason well enough to act on its understanding.

One example of AI is computer intelligence in playing games such as chess and checkers.





## *Game Intelligence*

For a computer to play a game against a human opponent, it must make intelligent decisions on its moves.

Strategy games such as checkers and chess have been targeted games for computing "artificial intelligence".

Even video games require the computer to determine strategies, even though the decision making is less complex.

- This includes games such as role-playing games and strategy/conquest games.

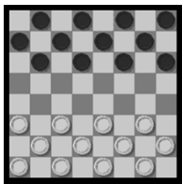




## *Checkers*

The world champion is a program called Chinook created by researchers at the University of Alberta, Canada.

- http://www.cs.ualberta.ca/~chinook
  - ⇨ Get crushed by Chinook if you choose



- They also have a research group called GAMES (Game- playing, Analytical methods, Minimax search, and Empirical Studies)
  - ⇨ http://www.cs.ualberta.ca/~games
- Checkers was solved in 2005. Perfect play by both sides leads to a draw. There are 500 billion billion ($5 \times 10^{20}$) positions.
- Group is working on poker players as well.





## *Chess*

A computer program does not have world champion status, but defeated world champion Gary Kasparov in a regulation match in 1997.

Deep Blue

- Developed by IBM at a cost of millions of dollars.
  - ⇨ Powered by a RS/6000 massively parallel mainframe.
  - ⇨ Can evaluate 200 million board positions a second.
  - ⇨ http://www.research.ibm.com/deepblue/home/html/b.html

## Jeopardy

IBM's Watson AI program competed on Jeopardy! in February 2011 against champions Ken Jennings and Brad Rutter.

- Watson: $77,147  Jennings: $24,000  Rutter: $21,000

Watson is a specialized program with a huge, self-contained database that parses English, formulates queries to database, and filters and selects correct answer.

- Database has 200 million unstructured pages.
- Watson consisted of 2,800 computers and terabytes of memory.
- Applied to medicine, banking, and research.

"Final Jeopardy!" question it got wrong in category U.S. Cities: *Its largest airport is named for a World War II hero, its second largest for a World War II battle.*





## A Simple Game Tic-Tac-Toe

A good way to look at structures and algorithms that are capable of playing games of pure skill is by examining a simple game like Tic-Tac-Toe (also called x's and o's).

Tic-Tac-Toe

- A game of pure skill
  - ⇨ No element of chance
- Can program Tic-Tac-Toe by "looking" for forced moves, traps, and patterns.
  - ⇨ Careful case by case analysis
    - Can be done because of the relatively few cases possible





## Game Playing Mini-max Strategy

The majority of game playing systems employ something called a ***mini-max strategy***.

- The basic idea in a mini-max strategy is that you determine a move which ***maximizes*** your potential to win the game and ***minimizes*** your opponent's potential to win the game.

The mini-max strategy consists of three components:

- **Move generator** - determine your possible moves
- **Board evaluator** - evaluate the desirability of each move
- **Mini-max procedure** - determine an efficient way to search through all the possible moves that you can perform

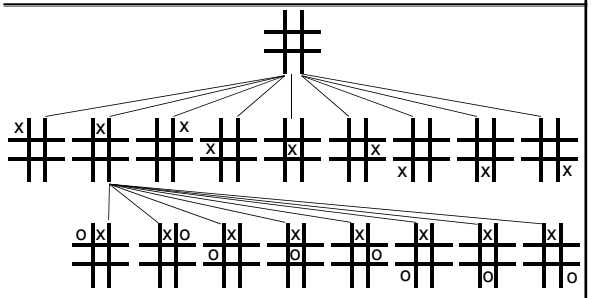
All of these components use or work upon a ***game tree***.

- A game tree stores, and allows the mini-max procedure to manipulate, the possible moves that can be made.





## Move Generator Example



**Note:** Many branches are omitted.
The second level would actually contain 9 x 8 = 72 nodes.





## Move Generator Tic-Tac-Toe

Creating a complete game tree starting from the empty state board for Tic-Tac-Toe turns out to be more complex than you might first expect:

- The game tree contains approximately **550,000** nodes.
  - ⇨ Easy for a computer to handle, but not insignificant.

For more complex games, the complete game tree is effectively unmanageable because the number of possible nodes in the game tree is unbelievably large.

- Therefore, we will not want to construct the entire game tree when making a decision, but rather only construct and search the most "promising" parts of the game tree.

***Heuristics*** and ***pruning*** are used to only evaluate the most likely beneficial moves.





## Board Evaluator

The second component of a game system is the ***board evaluator*** which is responsible for determining if a given board position or state is advantageous for the player.

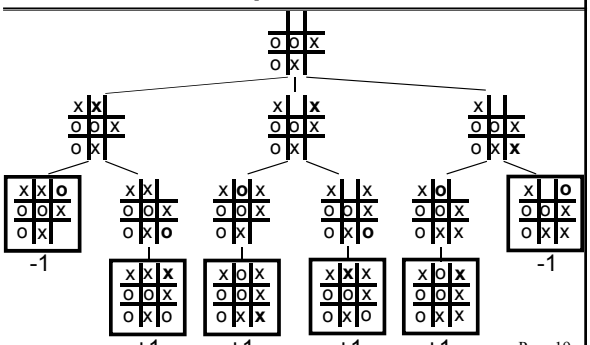
- The board evaluator determines the good and bad moves.

The move generator builds a game tree to get some insight as to what might happen in future moves:

- Future board scenarios are thus known by playing out moves, counter-moves, counter-counter-moves, etc.
- Future board scenarios are of no use if you have no mechanism to evaluate them.

The board evaluator determines when a sequence of moves (a path in the game tree) is advantageous for the player.

## Board Evaluator
## Tic-Tac-Toe Example








## Why did Deep Blue Win?

Deep Blue ended up winning due to increasing computation power.

- This extra power allowed the computer to examine more possible moves in the game tree.

The use of ***parallel computers*** that have multiple processors and memory allow for complex problems to be solved.

- The top 500 most powerful computers in the world have thousands of processors and are used for simulations of weather, military tests, and earthquakes.

Is Deep Blue intelligent?

- The search algorithm was "intelligent", but does it qualify as what we consider intelligence?





## Survey
## Computer Games

***Question:*** I have noticed an improvement in the intelligence/interactivity of the computer or computer characters in the games I play.

**A)** Yes
**B)** No





## Survey
## Computer Games and Your Time

***Question:*** I have spent more time this semester playing games than working on this course.

**A)** Yes
**B)** No





## Survey
## Social Computer Games

***Question:*** I confess to playing social games Zynga (Farmville, MafiaWars, etc.) or other Facebook or online games:

**A)** Never – What a waste of time!
**B)** Never – I love games but those are **NOT** games!
**C)** Once a month or less
**D)** Once a week
**E)** Daily or many times per day.

- Help me! I am addicted. I play all the time (even during class)!





## Computer Creativity

Computers can run programs that automatically generate music, art, and pictures.

- The intelligence is still with the software - not the computer.
- The software is encoding human intelligence.

The underlying question is: Is creativity algorithmic? If it is, computers may one day be creative.

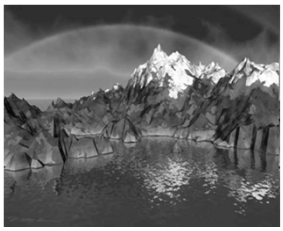
- Many things that are creative are algorithmic.
  - ⇨Mathematics was once considered creative or inspired.

Creativity is sometimes inspiration but is also a lot of revision.

- Inspiration to create something totally new. Revision is modifying existing to produce something new. Algorithmic?
- Many "new" advertising, research, etc. are based on revisions.

## Computer Generated Art

These pictures are generated from algorithms.



Source: Ken Musgrave - http://www.kenmusgrave.com/art_gallery.html





## The Universality Principle

In theory, all computers have the same ability to compute as they use the same basic functions.

- This is called the ***Universality Principle***.

In practice, differences in computer hardware, software, and operating systems make it impossible to run all software on all computers and to run it efficiently.

Examples:

- processors encode instructions differently in hardware
- operating systems support different features
- programs require processing speed that hardware cannot achieve

Six basic instructions: Add, Subtract, Set_to_One, Load, Store, and Branch_On_Zero.





## Why are some programs faster than others?

Recall that an ***algorithm*** is a sequence of steps to solve a problem.

The performance of an algorithm when implemented on a computer depends on the approach used to solve the problem and the actual steps taken.

Although faster hardware makes all algorithms faster, algorithms that solve the same problem can be compared in a hardware-independent way using ***big-Oh*** notation.





## Algorithms Best and Worst Case

Very few algorithms have the exact same performance every time because the performance of an algorithm typically depends on the size of the inputs it processes.

The ***best case*** performance of the algorithm is the most efficient execution of the algorithm on the "best" data inputs.

The ***worst case*** performance of the algorithm is the least efficient execution of the algorithm on the "worst" data inputs.

The ***average case*** performance of the algorithm is the average efficiency of the algorithm on the set of all data inputs.

Best, worst, and average-case analysis typically express efficiency in terms of the input size of the data.

- The input size is often a function of ***n***.





## Algorithms Big-Oh Notation

***Big-Oh notation*** is a mechanism for quickly communicating the efficiency of an algorithm.

- Big-Oh notation measures the worst case performance of the algorithm by ***bounding*** the formula expressing the efficiency.

In big-Oh notation:

- The performance is specified as a function of ***n*** which is the size of the problem.
  - e.g. *n* may be the size of an array, or the number of values to compute
- Only the most significant expression of ***n*** is chosen:
  - e.g. If the method performs $n^3 + n^2 + n$ steps, it is $O(n^3)$.
  - Significance ordering: $2^n$, $n^5$, $n^4$, $n^3$, $n^2$, $n*log(n)$, $n$, $log(n)$
- Constants are ignored for big-Oh:
  - e.g. If the method performs $5*n^3 + 4*n^2$ steps, it is $O(n^3)$.





## Algorithms Common Big-Oh Notation Values

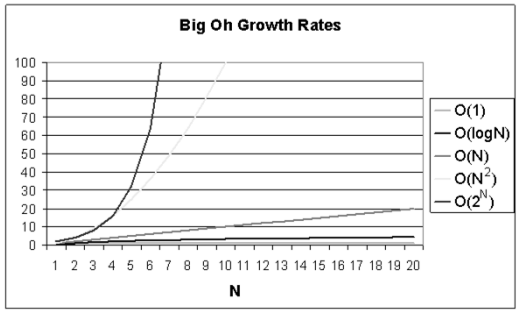
There are certain classes of functions with common names:

- constant = O(1)
- logarithmic = O(*log n*)
- linear = O(*n*)
- quadratic = O($n^2$)
- exponential = O($2^n$)

These functions are listed in order of fastest to slowest.

- For example, for large values of *n*, an algorithm that is considered O(*n*) is faster than an algorithm that is O($2^n$).
- Big-Oh notation is useful for specifying the growth rate of the algorithm execution time.
  - How much longer does it take the algorithm to run if the input size is doubled?

## Big Oh Growth Rates








## Big-Oh Exercise

1) What is the Big-Oh for the following formulas?

- a) $4*n^3 + 3*n^2 + 6*n$
- b) $n + n*log(n)$





## Big Oh Notation

***Question:*** What is the big Oh for the following formula:

$$4*n^2 + 3*n^4 + 6*n$$

**A)** $O(n^2)$
**B)** $O(3n^4)$
**C)** $O(n^4)$
**D)** $O(n)$





## Big-Oh Notation

***Question:*** What is the big Oh notation for the formula:

$$4*n^2 + 3*n^4 + 6*n + 5*n*log(n)$$

**A)** $O(4*n^2)$
**B)** $O(n^4)$
**C)** $O(6*n^4)$
**D)** $O(5*n*log(n))$
**E)** $O(n*log(n))$





## Best/Worst/Average Case

***Question:*** Assuming your mark is given between 0 and 100. How many of the three values (best case, worst case, average case) do you know for sure regardless who is in the class?

**A)** 0
**B)** 1
**C)** 2
**D)** 3





## Best/Worst/Average Case Finding a Song

***Question:*** You have 1000 songs on your music player and want a particular song. You "search" for your song by pressing the random button (pick a random song) until your song comes up. How many times do you have to press the random button for each case? **Assume that the randomize feature can return the same song more than once.**

**A)** best case = 1, worst case = 1000, average case = 500
**B)** best case = 1, worst case = 500, average case = 500
**C)** best case = 1, worst case = forever, average case = 500

## How Hard Can a Problem Be?

There exists problems that no computer can solve efficiently.

These problems are called ***NP-complete problems*** and are considered ***intractable***.

- The only way to solve the problem is to try all possible solutions to find the best.
- Even the most powerful computers cannot solve large examples of these problems.
- Example problem: Travelling salesman problem - find best route between *n* cities.

**Holy Grail:** Solving a NP-complete problem is a Holy Grail in computer science and would be an amazing breakthrough.





## How Hard Can a Problem Be? (2)

Even worse, there exist problems that have been proven to be unsolvable regardless of the computer speed.

There exist no algorithms at all for such ***unsolvable problems***.

An example is the Halting Problem that has the simple task of asking if a given program will always stop (halt).





## Computers in the Future

The future of IT is bright. There are many technologies being developed that are migrating from the research labs into use.

- Software agents – Can software be your personal butler?
- Robots – When we build robots, what would you want it to do?
- Self-healing and adapting – Can our systems fix themselves?
- Wearable computers – Can we embed computers in clothing and glasses? In our eyes and brains?
- Language translation – Can we have the universal translator?
- Personal Life Databases – Can we record all of our life information and moments (text, images, sound, video)?
  - ⇨What would that look like? Would you want that?
- Automatic driving cars – Our cars will do the driving (probably better than us). They already know where they are going…
- Presence technology – I know you are here…





## Computers in the Future (2)

Some challenges:

- Information overload
  - ⇨If we can get data from everywhere at any time, do we get too much?
  - ⇨Can we trust the data we get?
  - ⇨How about our privacy and security?
- Always-on society
  - ⇨Our technology has trained us to be always available for communication.
  - ⇨Is that good? Are we actually more productive that way? More human?
- Pace of innovation
  - ⇨Technology has sped up society and business. Everything changes rapidly. Innovation may not always be good.
- Essence of Humanity
  - ⇨If everything is automated and computerized around us, do we lose the essence of being human?
  - ⇨Are we ready for the ability to alter human DNA and lifestyles?





## Conclusion

Computers do not yet mimic human creativity although they demonstrate "***artificial intelligence***" in many domains.

- One of these domains is game playing where intelligence is provided by game trees and search strategies.

Computer scientists compare algorithms independently of hardware using ***big-Oh notation***.

***NP-complete problems*** are problems where no efficient solution exists. ***Unsolvable problems*** are problems where it is proven no solution at all exists.





## Objectives

- Explain the Turing Test in your own words.
- Define: artificial intelligence
- List and briefly explain the three components of game playing using game trees and the mini-max strategy.
- Define: Universality Principle
- Be able to convert a formula in *n* into big-Oh notation.
- Compare and contrast: best case, worst case, average case
- Compare and contrast: NP-complete problem, unsolvable problem

# COSC 122
# Computer Fluency

## Computer Fluency Summary

**Dr. Ramon Lawrence**
**University of British Columbia Okanagan**
**ramon.lawrence@ubc.ca**



## Computer Fluency

***Fluency*** means that you are able to adapt to new applications and use computers efficiently.

We have studied the skills, concepts, and capabilities of IT.

- Although the detailed skills may be forgotten or change over time, the fundamental concepts and capabilities allow us to learn new skills as required.

Remember, the key to being an expert user is using your past knowledge to understand how to use new systems.

- No one remembers all details and skills.





## Computer Fluency
## Skills, Concepts, and Capabilities

***Skills*** are the ability to use computers today to solve your problems.

- You have learned new applications: Word, Excel, Access, HTML editors and the ability to learn new applications.

***Concepts*** are the fundamental principles that apply to many situations. They are the building blocks of future learning.

- Key concepts: how the Internet (TCP/IP) works, how a computer works (Fetch/Execute cycle), key components of programming (HTML/JavaScript), information representation, security

***Capabilities*** are ways to expand your thinking.

- Thinking algorithmically, reasoning, debugging, designing, creating, searching and representing information.





## The Big IT Ideas

The big IT ideas essentially boil down to two things:

- ***Information must be structured on the computer to be useful.*** All information is represented as bits, so knowing the context is essential for understanding the meaning.
- ***Programs encode algorithms to solve problems.*** Algorithms represent intelligence on how to solve problems and provide the computer with the context and capability to perform all its advanced functions.
  - ⇨ Computer programming is the art and science of solving problems on the computer.





## Computers in Society

By understanding the technology, we have a better perspective on the role and influence of computers in society.

Like all technologies, information technology can be used for positive change and negative actions.

As users, and even designers, we have a role to play in shaping the effect of technology on this world. Displaying good ethics and protecting privacy is as important as building complex computer systems.





## Lifelong IT Learning

This course has prepared you for lifelong IT learning.

Computer systems and technology ***will*** change (the skills), but it is the ***attitude*** that is most important.

How much information in the course will you remember?

How much do you *need* to remember to apply the concepts?

As an expert user, you are confident and ready to:

- Learn new systems with confidence by applying gained knowledge, experience, and fundamental concepts.
- Ask for help (when needed) by understanding key terminology and components of computers.
- Evaluate new systems that may improve your productivity.
- Protect yourself in the digital world by understanding the role, benefits, and issues of computer systems.

# *Want more?*
# *Where to go from here?*