COSC 121: Computer Programming II

Dr. Bowen Hui

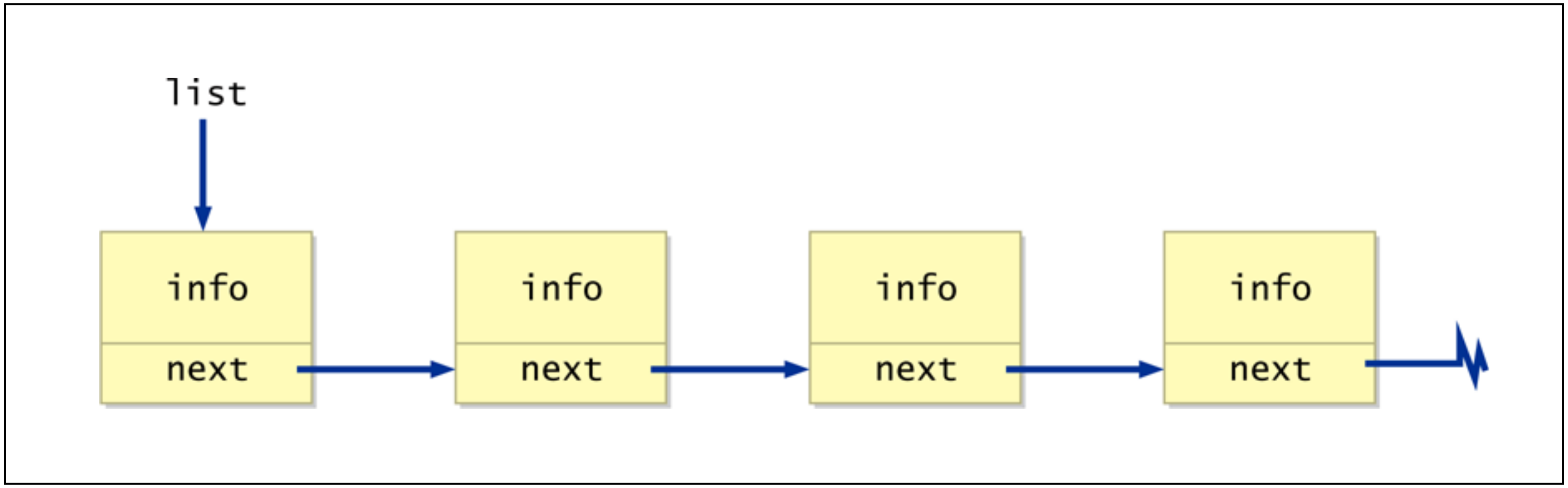
University of British Columbia Okanagan

Quick Review

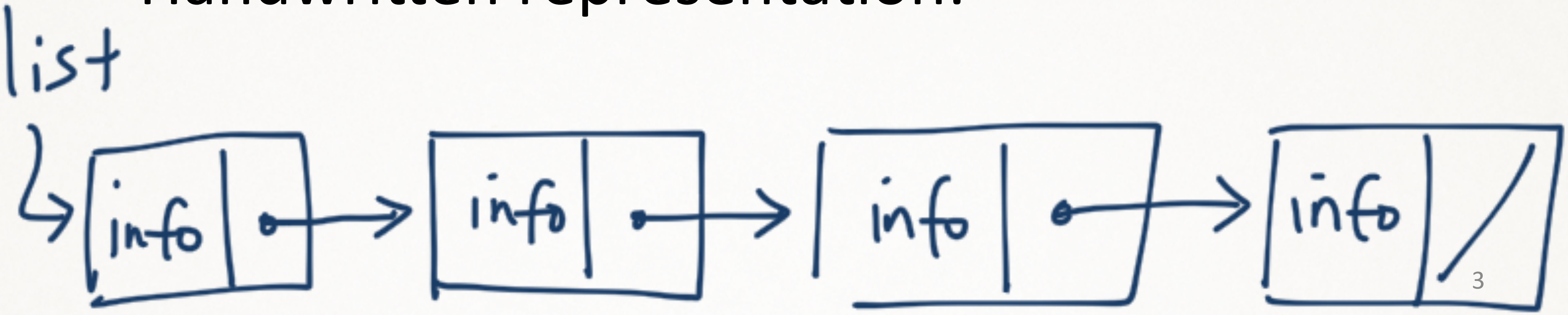
- Linked lists is a dynamic ADT that store information in variable sized structures
  - Efficient memory usage
  - Harder to implement (must manage everything manually)
  - Abstracts implementation detail from other classes
- Operations we saw: add, delete, insert

Quick Review (cont.)

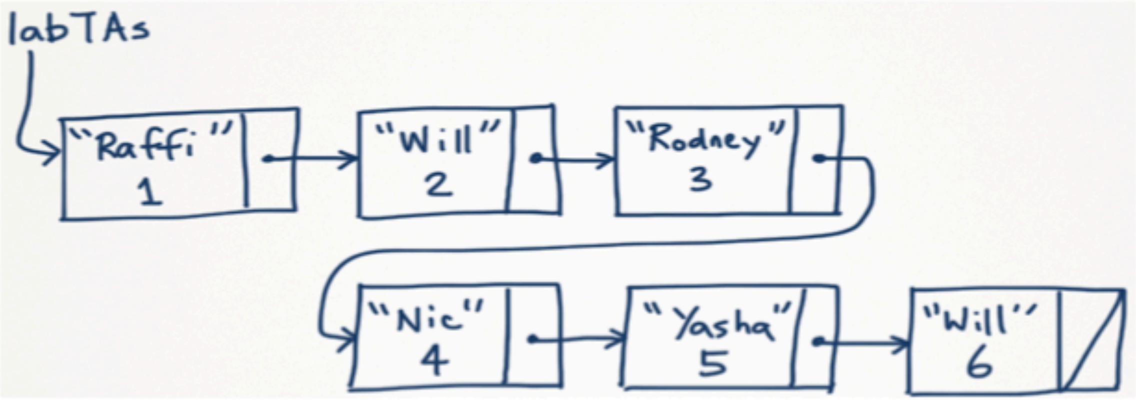
- Textbook representation:



- Handwritten representation:

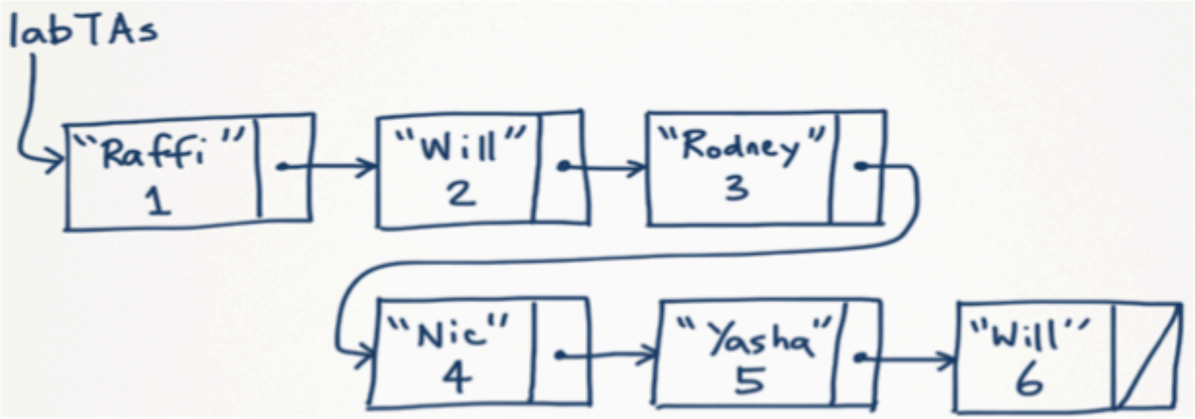


Short Conceptual Exercises



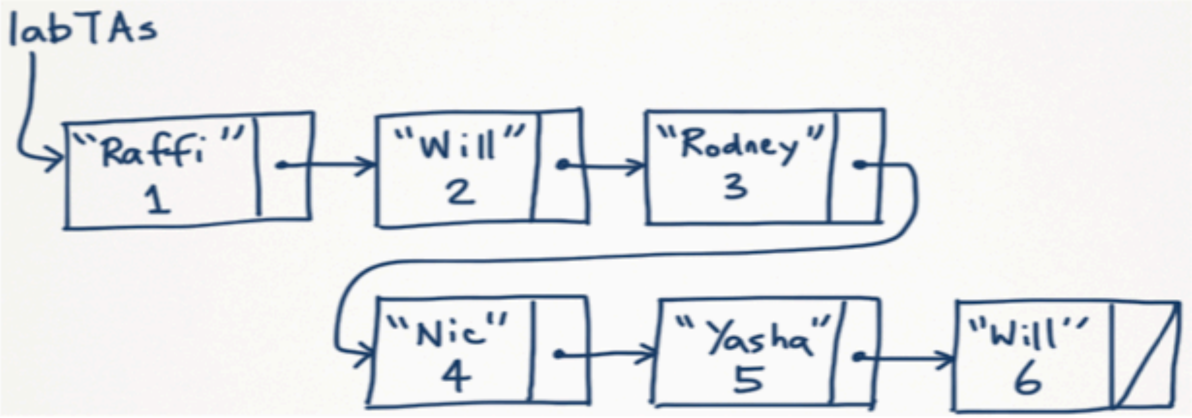
1. Using the above list, draw the resulting linked list that changes the node with "Rodney" to "Yasha", while leaving the section number the same.

Short Conceptual Exercises



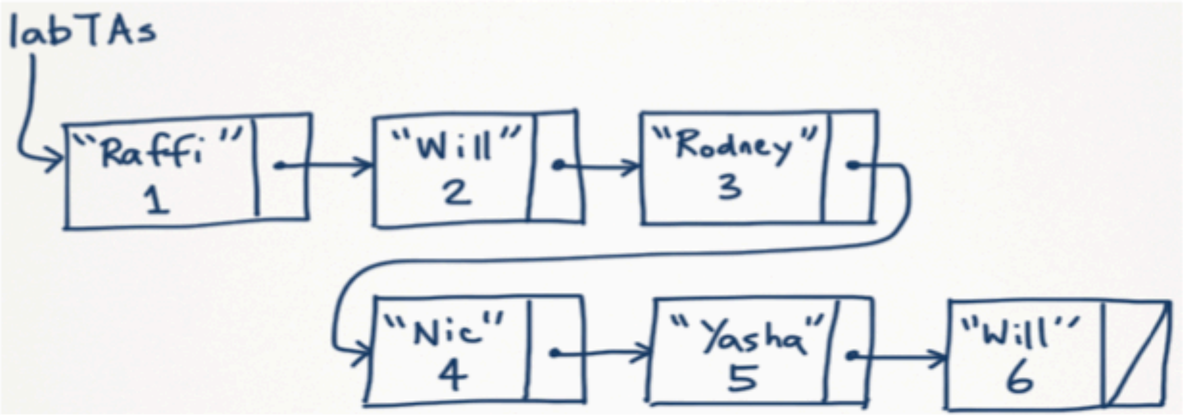
2. Draw the resulting linked list that deletes the node with "Will" and section 6 from the above list.

Short Conceptual Exercises



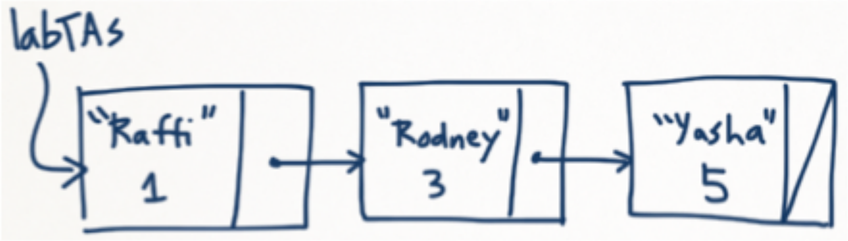
3. Draw the resulting linked list that deletes the node with "Will" and section 2 from the above list.

Short Conceptual Exercises



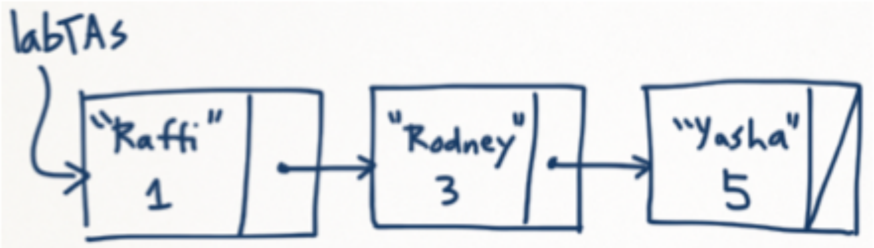
4. Draw the resulting linked list that deletes the node with "Raffi" and section 1 from the above list.

Short Conceptual Exercises



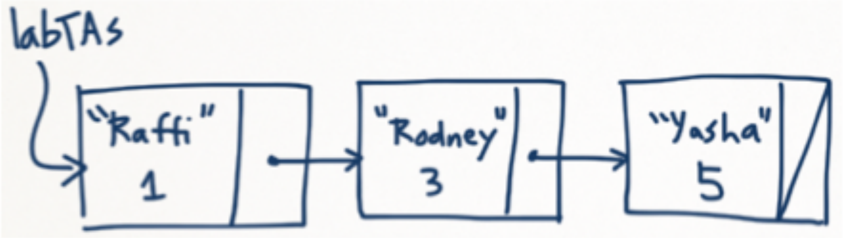
5. Draw the resulting linked list that adds the node with "Will" and section 6 to the end of the above list.

Short Conceptual Exercises



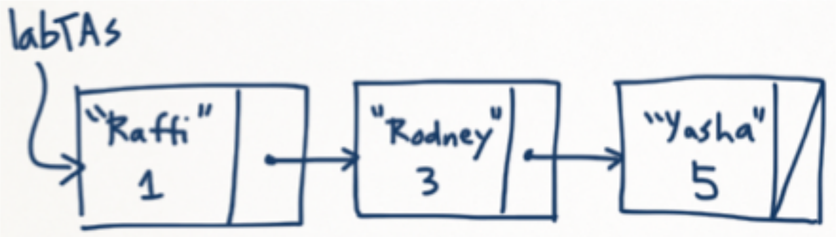
6. Draw the resulting linked list that adds the node with "Rodney" and section 3 to the end of the resulting list from the previous question.

Short Conceptual Exercises



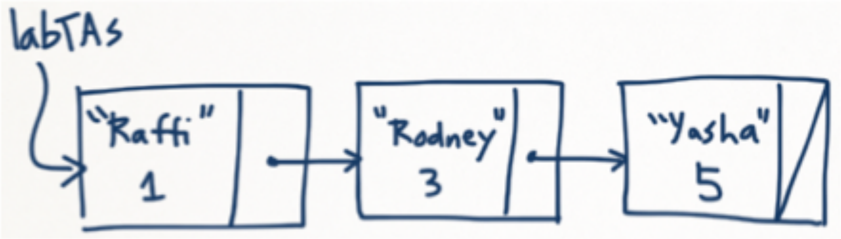
7. Using the above list, draw the resulting linked list that inserts the node with “Nic” and section 4 to the above list, directly after the node with “Rodney” and section 3.

Short Conceptual Exercises



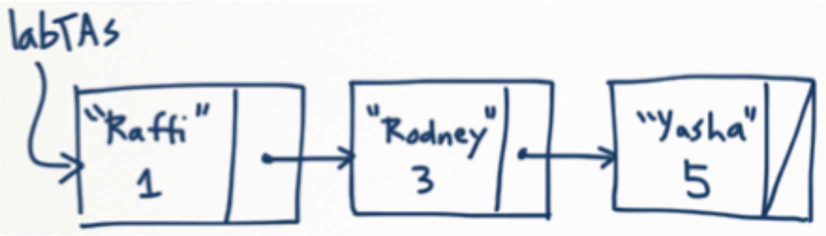
8. Using the above list, draw the resulting linked list that inserts the node with "Will" and section 2 to the above list, directly after the node with "Raffi" and section 1.

Short Conceptual Exercises



9. Using the above list, draw the resulting linked list that inserts the node with "Nic" and section 4 to the above list, directly before the node with "Raffi" and section 1.

Short Conceptual Exercises



10. Using the above list, draw the resulting linked list that inserts the node with "Will" and section 2 to the above list, directly after the node with "Yasha" and section 5.

A Complete Linked List Class

- Write a program that models a list of students so we can keep track of them, who is registered, and who is not.
- Which classes will you need?

A Complete Linked List Class

- Write a program that models a list of students so we can keep track of them, who is registered, and who is not.
- Which classes will you need?
  - A class to store the student as a node
  - A class to keep track of students as a linked list
    - Has methods for adding and deleting nodes
  - A class to test the entire program

StudentNode.java from Last Class

```
public class StudentNode {
  private String        name;
  private int           id;
  private StudentNode   next;
  public Node( String sn, int sid )
  {
    name = sn;
    id   = sid;
    next = null;
  }
  public String getName()                   { return name; }
  public String getId()                     { return id; }
  public void setNext( StudentNode other )  { next = other; }
  public String toString()
  {
    return “Name: ” + name + “(” + id + “)”;
  }
}
```

Writing the Registration Class

```
public class Registration {
  // keep track of the list of students

  // creates an empty list of students
  public Registration() { … }

  // creates a new node with given name and id
  // adds it to end of list
  public void add( String name, int id ) { … }

  // finds node with given ID and deletes it
  public void delete( int id ) { … }

  // inserts new node based on IDs
  // assume list is sorted from largest to smallest
  public void insert( String name, int id ) { … }

  // traverse each node in list
  // concat each node’s toString() info
  public String toString() { … }
}
```

Summary of Linked Lists

- Basic operations:
  - add
  - delete
  - insert
  - toString
  - traverse
- Use of test cases:
  - Consideration of multiple scenarios to ensure method details are implemented properly