

PA1 due Wed @ 8am

Lecture 5

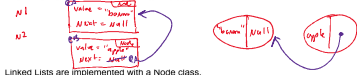
What is a Linked List?

A Linked List is a data structure that implements a List ADT, where elements in the list may appear anywhere in memory, but are "linked" together in a particular order using references or pointers.

```
class Node {
    String value;
    Node next;
    public Node(String value, Node next) {
        this.value = value;
        this.next = next;
    }
}

// Somewhere else in the code... still inside Node class (can access next)
Node n1 = new Node("banana", null);
Node n2 = new Node("apple", null);
n2.next = n1;
```

Draw the memory model diagram for this code.



Linked Lists are implemented with a Node class.

The Node forms the structure of the list.

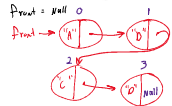
It contains:

- A reference to the data stored at that position in the list
- A reference to the next node in the list
- Optionally (for a doubly linked list) a reference to the previous node in the list.

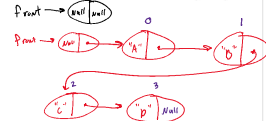
The Linked List itself usually contains only a reference to the first node in the list (head), and sometimes a reference to the last node (tail). It also might store the list's size.

Assume that a linked list contains the following elements in this order: "A", "B", "C", "D". Draw the memory model of the linked list. Label the index of each element.

Without a sentinel node (root starts with null).



With a sentinel node (root starts with a dummy Node):



Name: _____ PID: _____ Code: 5268

```
public class LinkedListStringList implements StringList {
    Node front;
    int size;

    public LinkedListStringList() {
        this.front = new Node(null, null);
    }

    /* Add an element at the beginning of the list */
    public void prepend(String s) {
        Node newFront = new Node(s, this.front.next);
        this.front.next = newFront;
        this.size++;
    }

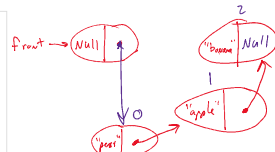
    /* Add an element at the end of the list */
    public void add(String s) {
        Node curNode = front;
        while (curNode.next != null) {
            curNode = curNode.next;
        }
        curNode.next = new Node(s, null);
        size++;
    }

    /* Add an element at the specified index */
    public void insert(int index, String s) {
        Node curNode = front;
        int count = 0;
        while (count < index) {
            curNode = curNode.next;
            count++;
        }
        Node node = new Node(s, curNode.next);
        curNode.next = node;
        size++;
    }

    /* Remove the element at the specified index */
    public void remove(int index) {
        Node curNode = front;
        for (int i = 0; i < index; i++) {
            curNode = curNode.next;
        }
        curNode.next = curNode.next.next;
        size--;
    }
}
```

Draw the linked list for the following method calls:

```
prepend("banana");
prepend("apple");
prepend("pear");
```



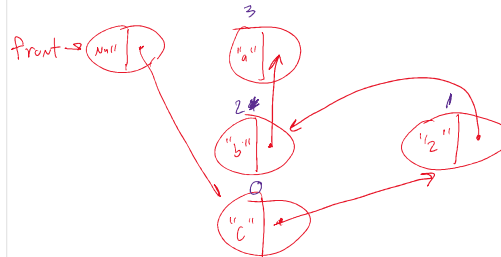
Draw the linked list for the following method calls:

```
add("banana");
add("apple");
add("pear");
```



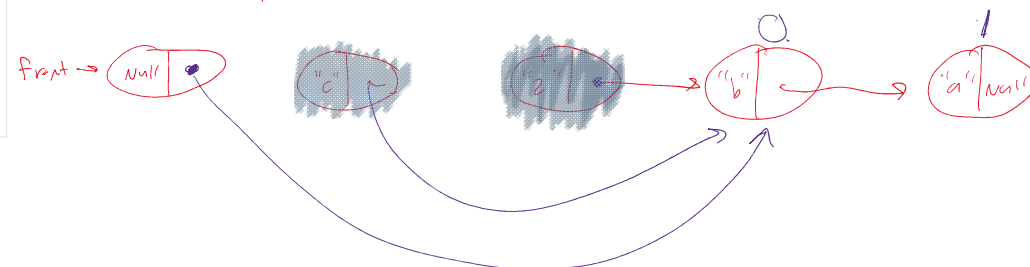
Draw the linked list for the following method calls:

```
insert(0, "a");
insert(0, "b");
insert(0, "c");
insert(1, "z");
```



Draw the linked list for the following method calls:

```
add("c");
add("z");
add("b");
add("a");
remove(1);
remove(0);
```



no dummy (empty list)

front = null

add()

```
if (front == null) {
    front = new Node(...);
}
```

else {

```
// find null
    .next = new Node(...);
}
```