

# CSE 12 – Basic Data Structures and Object-Oriented Design

## Lecture 5

Greg Miranda & Paul Cao, Winter 2021

# Announcements

- Quiz 5 due Friday @ 8am
- Survey 2 due Friday @ 11:59pm
- PA1 due tonight @11:59pm
- PA2 released tomorrow (closed PA)

# Topics

- Linked List Implementations

# So 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.

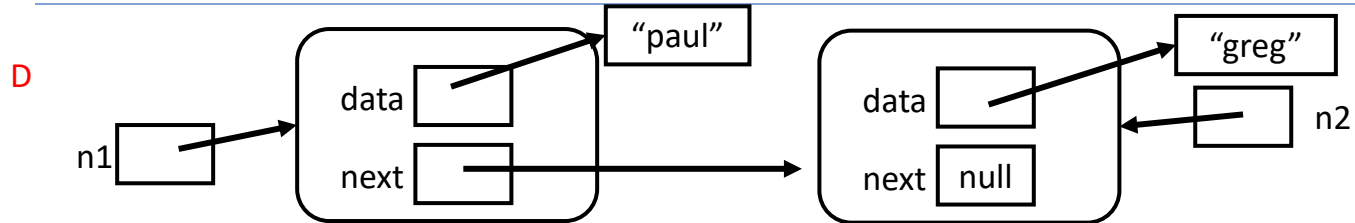
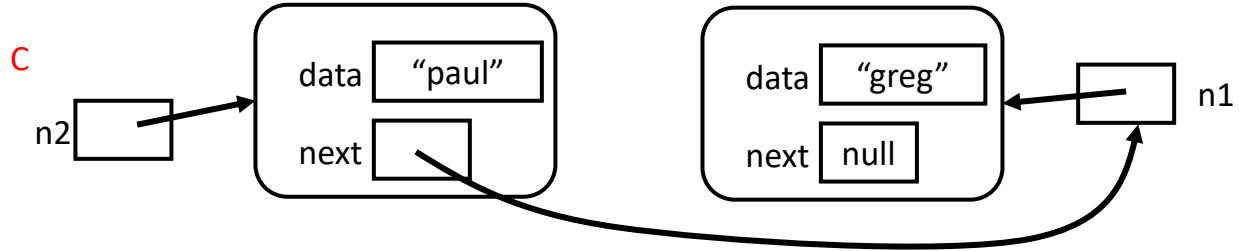
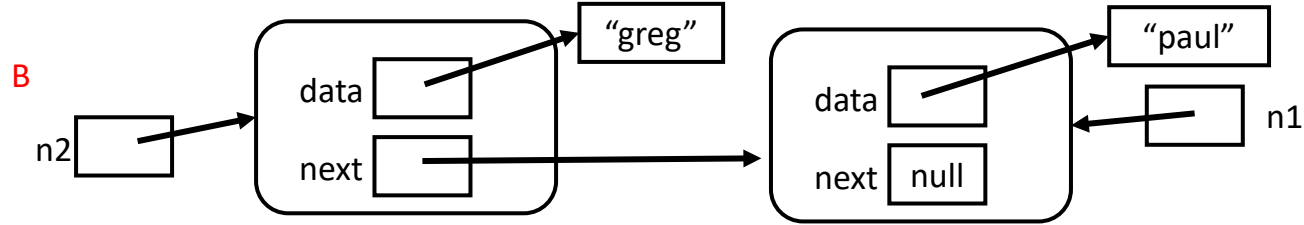
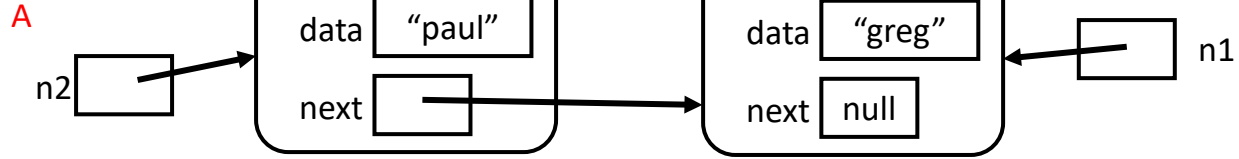
# Memory Model Diagrams and LinkedLists

```
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("paul", null);  
Node n2 = new Node("greg", null);  
n2.next = n1;
```

Draw the memory model diagram for this code. Answer choices next slide.

```
Node n1 = new Node("paul", null);  
Node n2 = new Node("greg", null);  
n2.next = n1;
```

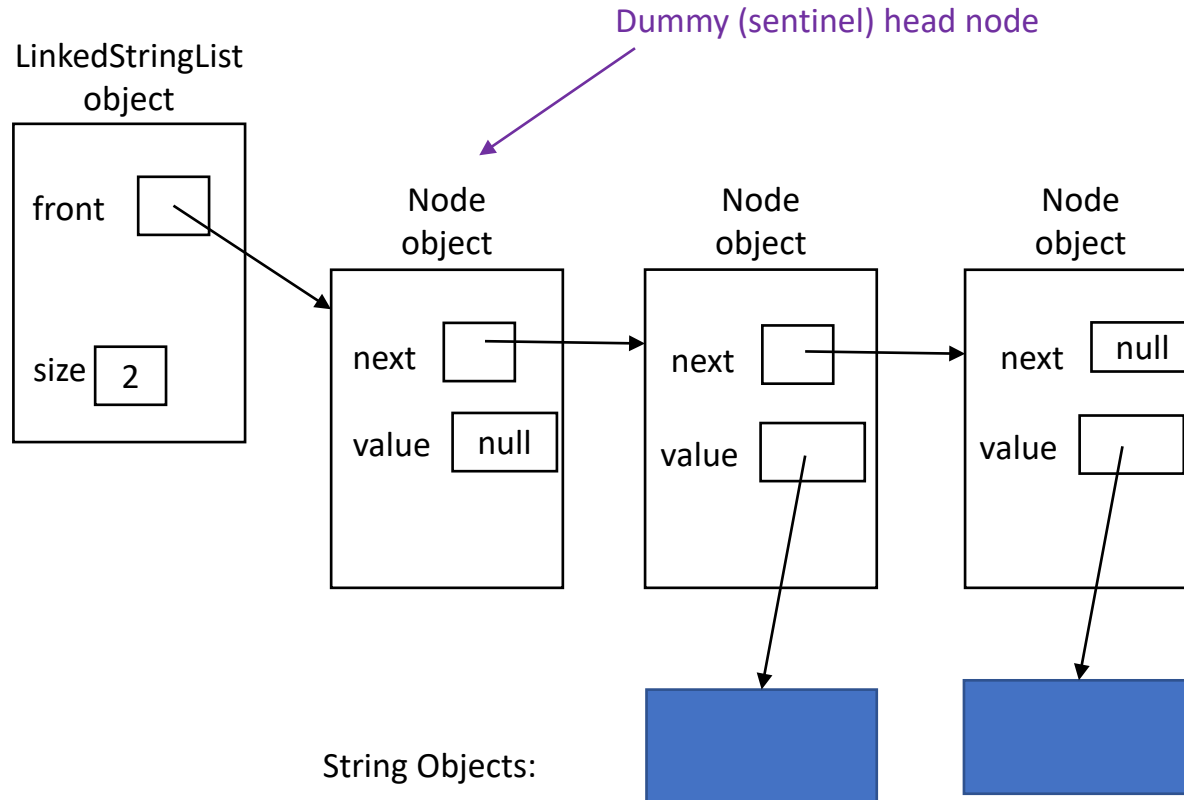


**E. None of these**

# Toward Linked List Implementation

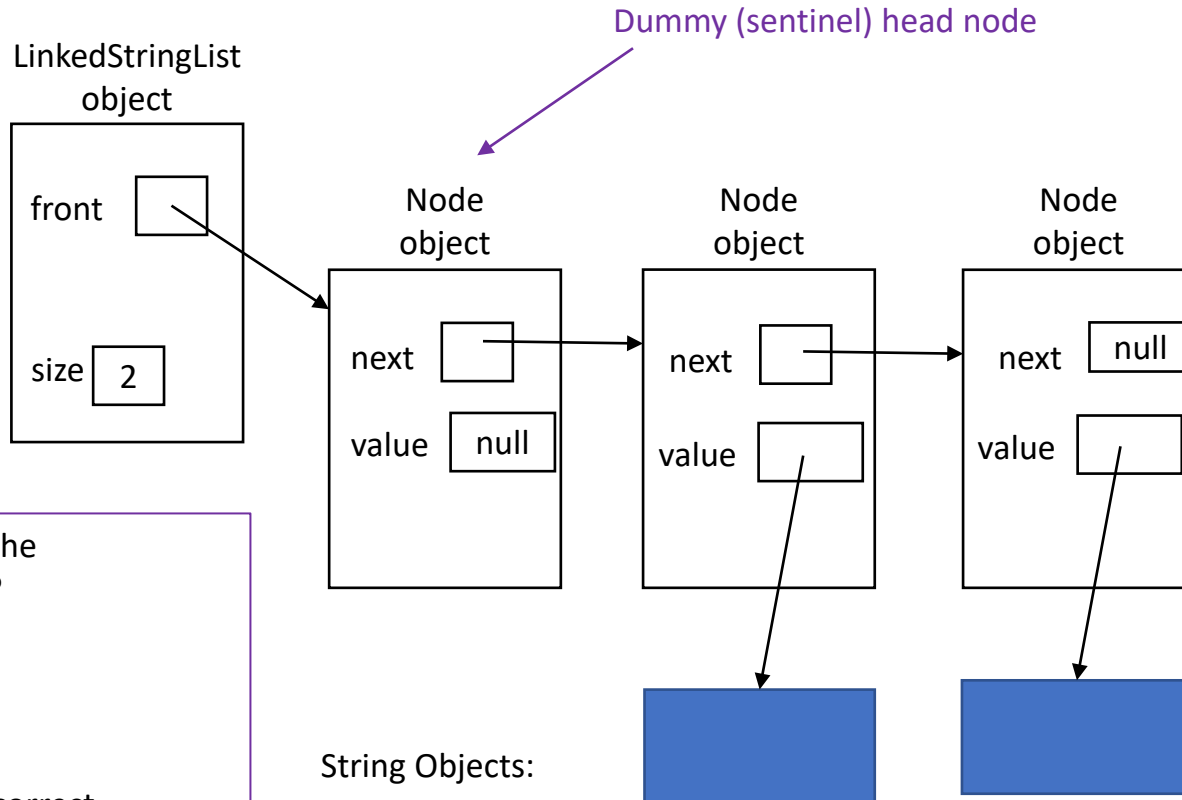
- Linked Lists are implemented with a Node class.
- The Node forms the structure of the list. It contains:
  - A reference to the data stored at that position in the list
  - A reference to the next node in the list
  - Optionally (for a doubly linked list) a reference to the previous node in the list.
- 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.

# Singly Linked List with sentinel Node: Picture





# Singly Linked List with sentinel Node: Picture

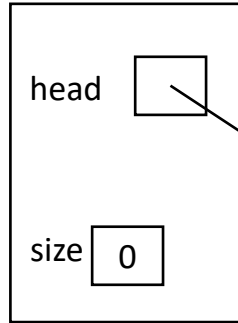


What type is head in the LinkedList class?

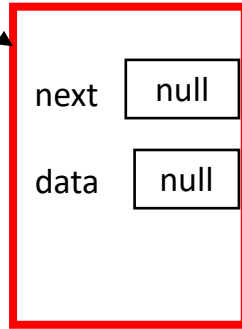
- A. Node
- B. String
- C. LinkedList
- D. StringList
- E. More than one is correct

# Empty Singly Linked List with sentinel node

MySinglyLinkedList  
object

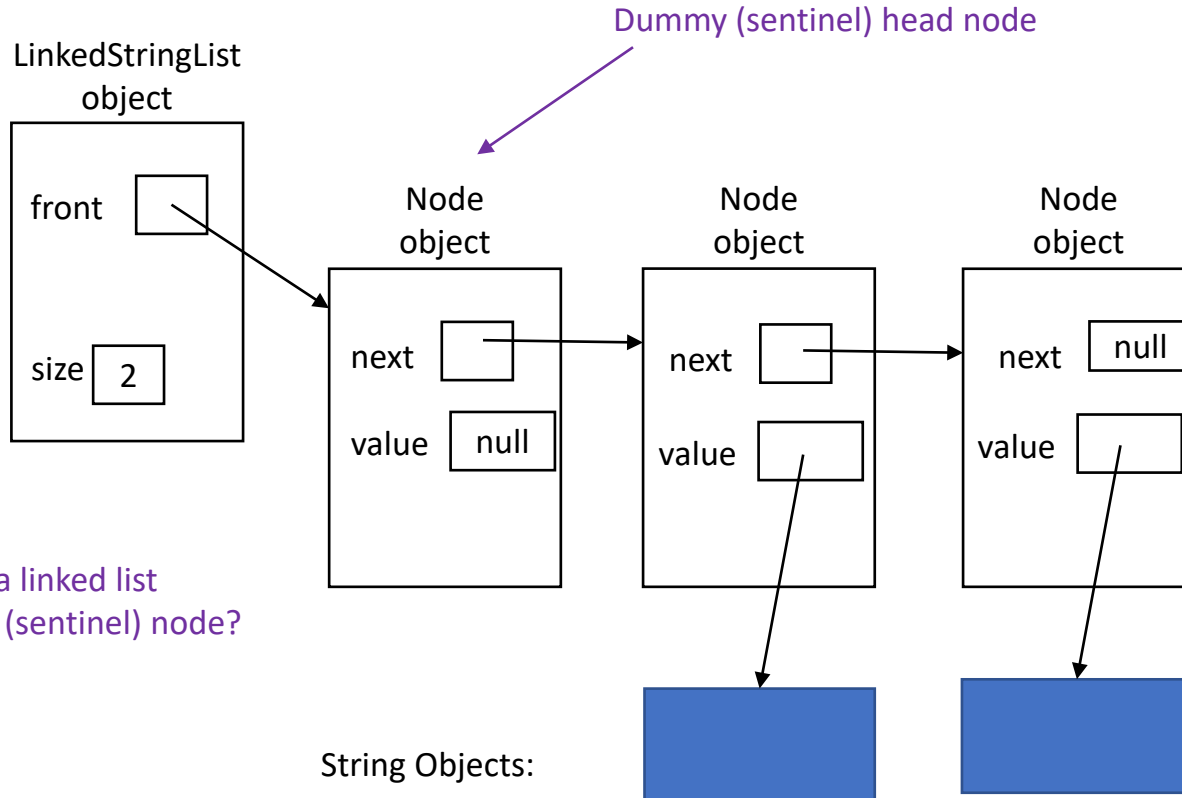


Node  
object



*This node is always there!!*

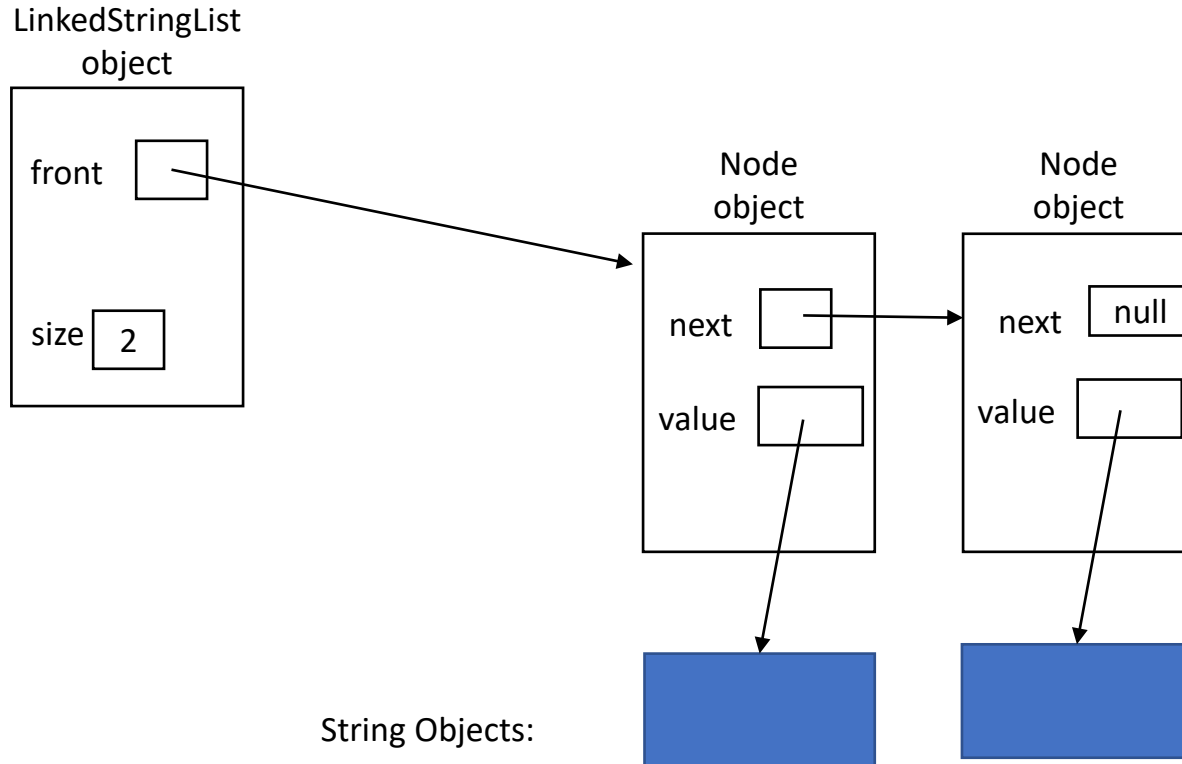
# Singly Linked List with sentinel Node: Picture



Can you implement a linked list without this dummy (sentinel) node?

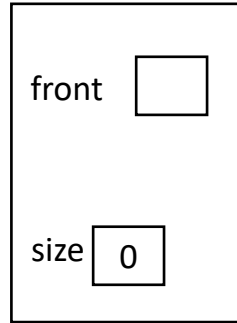
- A. Yes
- B. No

# Singly Linked List without sentinel Node: Picture

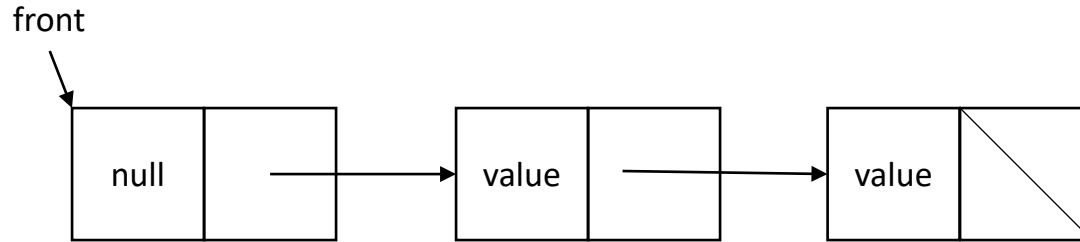


# Empty Single Linked List without sentinel node

LinkedList  
object

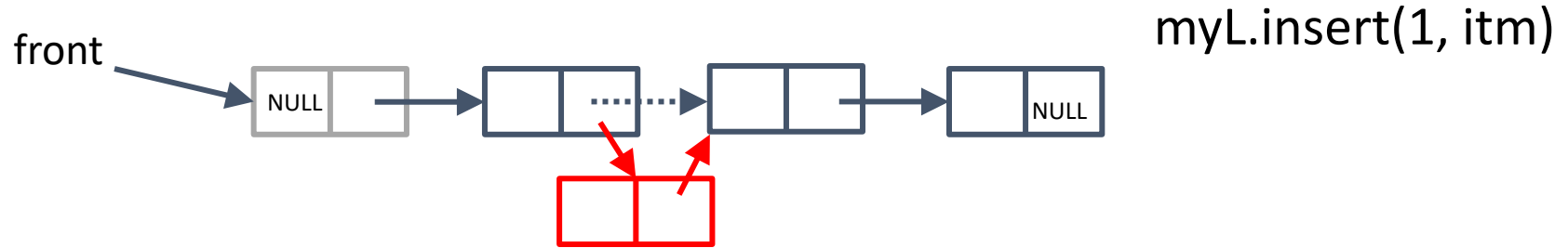


# Singly Linked List: Abstracted Picture



Does this list use a sentinel node?

- A. Yes
- B. No
- C. Not sure

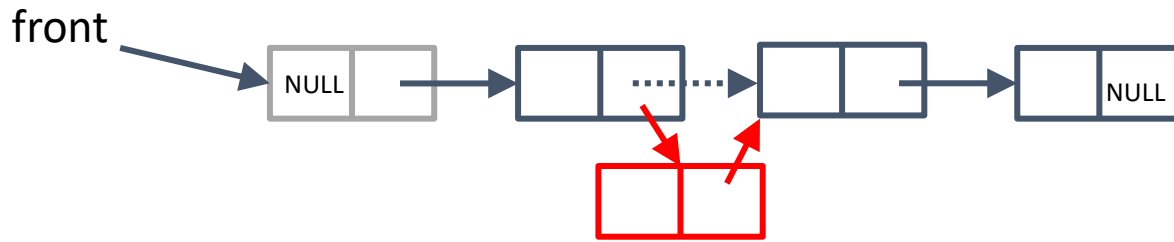


```
// In LinkedList class (NOT Node class)
```

```
public void insert(int index, String s) {
```

```
}
```

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



myL.insert(1, itm)

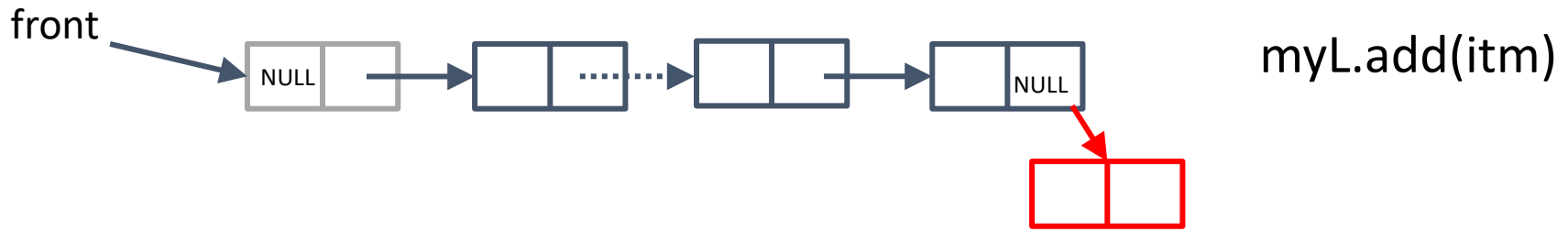
```
public void insert(int index, String s) {  
    Node newNode = new Node(s, null);  
    if (index < 0 || index > size) throw new IndexOutOfBoundsException();  
    Node curr = this.front;  
    for(int i = 0; i < index; i++) {  
        curr = curr.next;  
    }  
  
    this.size += 1;  
}
```

What line of code will complete this method correctly (in the blank)?

- A) No line is needed. The code is correct as written.
- B) `curr.next = newNode;`
- C) `curr = newNode;`
- D) `newNode.next = curr.next;`
- E) None of them is correct

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



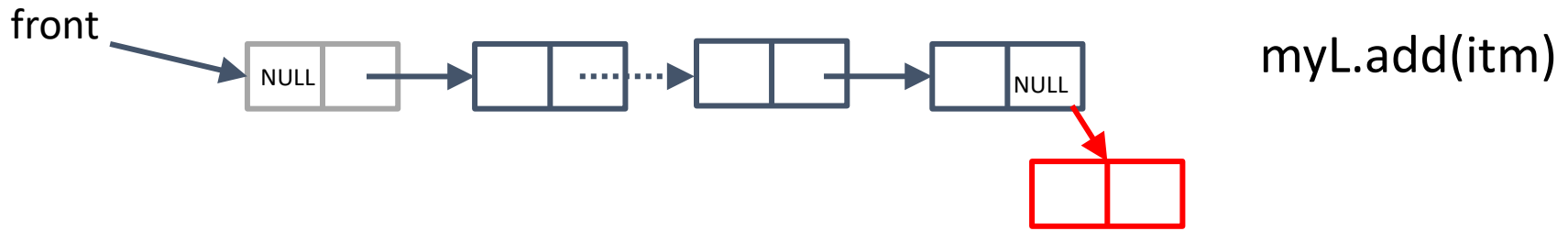


```
public void add(String s) {  
    Node curr = this.front;  
    while(_____A_____ != null) {  
        curr = curr.next;  
    }  
    _____B_____  
    this.size += 1;  
}
```

What line of code will complete this method correctly for blank A?

- A) curr.next
- B) curr
- C) front
- D) front.next
- E) None of them is correct

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



```
public void add(String s) {  
    Node curr = this.front;  
    while(curr.next != null) {  
        curr = curr.next;  
    }  
    _____B_____  
    this.size += 1;  
}
```

What line of code will complete this method correctly for blank B?

- A) No line is needed. The code is correct as written.
- B) `curr.next = new Node(s, curr.next);`
- C) `curr = new Node(s, null);`
- D) `curr.next = new Node(s, curr);`
- E) None of them is correct

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

# LinkedList Remove

```
/* Remove the element at the specified index */  
void remove(int index);
```

- Write a test case for the LinkedList remove method
- Implement the LinkedList remove method