AZHAR ALI
023-23-0314
SEC A

LAB 2

FALL 2024

TASK 1 AND 2

Methods

```
boolean SearchNode(int value){

    if(head==null){
        System.out.println("List is Empty");
        return false;
    }
    else  {
        Node temp = head;
        while(temp!=null){
            if(temp.data==value){
            System.out.println("Value Found!" + value);
            return true;
        }
        temp = temp.next;
        }}

        System.out.println("Value Not Found!");
        return false;

    }
int FindLength(){
    int count = 1;
    if(head==null){
        System.out.println("List is Empty");
        return 1;
    }
    else {
        Node temp = head;
        while(temp.next!=null){
           count++;
           temp=temp.next;
        }
        return count;
    }

}
```

Output

```
3 10 5 5 7
Value Not Found!
Length : 5
3 10 900 5 5 7

20 3 10 900 5 5 7

20 200 3 10 900 5 5 7

200 3 10 900 5 5 7
```

Main Method

```
Run | Debug
public static void main(String[] args) {

        Linked_List ll = new Linked_List();
        ll.addToFront(data:10);
        ll.addToBack(data:5);
        ll.addToBack(data:7);
        ll.addToFront(data:3);
        ll.printList();

        ll.SearchNode(value:4);
        System.out.println("Length : "+ll.FindLength());

        ll.addMiddle(index:3,data:900);
        ll.printList();
        System.out.println("");
        ll.addToFront(data:20);
        ll.printList();
        System.out.println("");
        ll.addMiddle(index:2,data:200);
        ll.printList();
        System.out.println("");
        ll.removeFromFront();
        ll.printList();

}
```

TASK 3

```
class Linked_ListOf3 implements List {
    class Node {
        String data;
        Node next;

        Node(String data) {
            this.data = data;
            this.next = null;
        }
    }
    Node head;
    int size;

    public Linked_ListOf3() {
        head = null;
        size = 0;
        }

    @Override
    public boolean isEmpty() {
        return head == null;
    }

    @Override
    public int size() {
        return size;
    }
```

TASK 3

```java
@Override
public void add(int index, String item) {
    if (index < 1 || index > size + 1) {
        System.out.println("Index should be less than or equal to the list size and greater than 0.");
        return;
    }

    Node newNode = new Node(item);
    if (index == 1) {
        newNode.next = head;
        head = newNode;
    } else {
        Node temp = head;
        for (int i = 1; i < index - 1; i++) {
            temp = temp.next;
        }
        newNode.next = temp.next;
        temp.next = newNode;
    }
    size++;
}

@Override
public void remove(int index) {
    if (index < 1 || index > size) {
        System.out.println("Index should be less than or equal to the list size and greater than 0.");
        return;
            }

    if (index == 1) {
        head = head.next;
    } else {
        Node temp = head;
        for (int i = 1; i < index - 1; i++) {
            temp = temp.next;
        }
        temp.next = temp.next.next;
    }
    size--;
}
```

TASK 3

```java
@Override
public void remove(String item) {

    if(head==null){
        System.out.println("List is empty!");
    }

    if(head.data.equals(item)){
        head = head.next;
        return;
    }

    Node temp = head;
    if( !temp.next.data.equals(item) && temp.next!=null){
        temp=temp.next;
    }
    if(temp==null){
        System.out.println("item not found!");
    }
    if (temp.next == null) {
        System.out.println("Item not found!");
    } else {
        temp.next = temp.next.next;
        size--;
    }

}

void printList() {
    if (head == null) {
        System.out.println("List is Empty!");
        return;
    } else {
        Node temp = head;
        System.out.print("[Size:"+size()+"-");
        while (temp != null) {
            System.out.print(temp.data+",");
            temp = temp.next;
        }
        System.out.print("]");
    }
}
```

```java
public Linked_ListOf3 duplicate() {
Linked_ListOf3 duplicateList = new Linked_ListOf3();

if (head == null) {
    return duplicateList;
}

Node current = head;
while (current != null) {
    duplicateList.add(current.data);
    current = current.next;
}

return duplicateList;
}
@Override
public Linked_ListOf3 duplicateReversed() {
    Linked_ListOf3 reversedList = new Linked_ListOf3();

    if (head == null) {
        return reversedList;
    }

    Node current = head;
    while (current != null) {

        Node newNode = new Node(current.data);
        newNode.next = reversedList.head;
        reversedList.head = newNode;
        current = current.next;
    }

    return reversedList;
}
```

TASK 3

Main Method and Output

```
Run | Debug
public static void main(String[] args) {
    Linked_ListOf3 LL = new Linked_ListOf3();
    LL.add(item:"This");
    LL.add(item:"Is");
    LL.add(item:"list");
    System.out.println("Original list:");
    LL.printList();
    LL.add(index:3, item:"Java");
    System.out.println("\nList after adding 'Java' at index 2:");
    LL.printList();
    LL.remove(item:"Is");
    System.out.println("\nList after removing 'Is':");
    LL.printList();
    Linked_ListOf3 duplicateList = LL.duplicate();
    System.out.println("\nDuplicate list:");
    duplicateList.printList();
    Linked_ListOf3 reversedList = LL.duplicateReversed();
    System.out.println("\nReversed duplicate list:");
    reversedList.printList();
    LL.remove(item:"is");
    System.out.println("\nList after to remove 'is':");
    LL.printList();
    LL.add(index:3, item:"Error");
    System.out.println("\nList after attempting to add 'Error' at index 3:");
    LL.printList();
}
```

```
Original list:
[Size:3-This,Is,list,]
List after adding 'Java' at index 2:
[Size:4-This,Is,Java,list,]
List after removing 'Is':
[Size:3-This,Java,list,]
Duplicate list:
[Size:3-This,Java,list,]
Reversed duplicate list:
[Size:0-list,Java,This,]
List after to remove 'is':
[Size:2-This,Java,]
List after attempting to add 'Error' at index 3:
[Size:3-This,Java,Error,]
```