28.-02024

LAB-3(WEEK-4)

2022503003

1. Write a Java program to implement the below listed task

Insertion:

Insert at the beginning.
Insert at the end.
Insert at a specific position.
Insert after a specific node.
Insert before a specific node.

Deletion:

Delete from the beginning.
Delete from the end.
Delete a specific element by value.
Delete a specific element by position.

Traversal and Display:

Traverse and print the elements in the linked list.
Reverse and print the elements in the linked list.

Search and Access:

Search for an element by value.
Access an element by position.
Length and Counting:
Find the length (number of nodes) of the linked list.
Count the occurrences of a specific value in the list.

Sorting :

Sort the linked list (best sort).

Concatenation:

Concatenate (combine) two linked lists together.

Duplicate Removal:

Remove duplicate elements from a linked list.

CODE:

```
import java.util.*;
class Node{
    int data;
    Node next;
    Node(int data){
        this.data=data;
        next=null;
    }
}
class LinkedList{
    Node head;          //not static bruhhh!(then all ll will have common head!)
    void insertBeg(int data){
        Node newNode=new Node(data);
        if(head==null){
            head=newNode;
        }
        else{
            newNode.next=head;
            head=newNode;
        }
        display();
    }
    void insertEnd(int data){
        if(head==null){
            insertBeg(data);
            return;
        }
        Node newNode=new Node(data);
        Node current=head;
        Node prev=current;
        while(current!=null){ //stopping at previous!
            prev=current;
            current=current.next;
        }
        prev.next=newNode;
        display();
    }
    //1->2->3->4->5->6
    void insertPos(int pos,int data){
        if (head == null && pos != 1) {
            System.out.println("List is empty! Insert at position 1.");
            return;
        }
        if(pos==1){
            insertBeg(data);
        }
        else{
            int count=pos-1;
            if (pos==1){
                insertBeg(data);
            }
            else{
                Node newNode=new Node(data);
                Node current=head;
                while(count!=1){
                    current=current.next;
                    count--;
                }
                Node temp=current.next;
                current.next=newNode;
                newNode.next=temp;
            }
            display();
        }
    }
    void insertAfterSpecific(int target,int data){
        if(head==null){
            System.out.println("List is empty!Try inserting first!");
            return;
        }
        Node current=head;
        Node newNode=new Node(data);
        Boolean flag=false;
        while(current!=null){
            if(target==current.data){
```

```
            flag=true;
            break;
        }
        current=current.next;
    }
    if(flag){
        Node temp=current.next;
        current.next=newNode;
        newNode.next=temp;
        display();
    }
    else{
        System.out.println(target+" data is NOT found!");
    }
}
void insertBeforeSpecific(int target,int data){
    if(head==null){
        System.out.println("List is empty!Try inserting first!");
        return;
    }
    Node current=head;
    Node newNode=new Node(data);
    if(current.data==target){
        insertBeg(data);
        return;
    }
    Boolean flag=false;
    while(current!=null){
        if(target==current.next.data){
            flag=true;
            break;
        }
        current=current.next;
    }
    if(flag){
        Node temp=current.next;
        current.next=newNode;
        newNode.next=temp;
        display();
    }
    else{
        System.out.println(target+" data is NOT found!");
    }
}
/*-----------------------------------------INSERTION-END--------------------------------------------------*/
void delBeg(){
    if(head==null){
        System.out.println("List is empty!Try inserting first!");
        return;
    }
    if(head.next==null){
        head=null;
        System.out.println("List is empty!");
        return;
    }
```

```
        head=head.next;
        display();
    }
    void delEnd(){
        if(head==null){
            System.out.println("List is empty!Try inserting first!");
            return;
        }
        if(head.next==null){
            head=null;
            System.out.println("List is empty!");
            return;
        }
        Node current=head;
        Node prev=current;
        while(current.next!=null){
            prev=current;
            current=current.next;
        }
        prev.next=null;
        display();
    }
    void delPos(int pos){
        if(head==null){
            System.out.println("List is empty!Try inserting first!");
            return;
        }
        int count=pos-1;
        if (pos==1){
            delBeg();
        }
        else{
            Node current=head;
            while(count!=1){
                current=current.next;
                count--;
            }
            current.next=current.next.next;
        }
        display();
    }
    void delVal(int target){
        if(head==null){
            System.out.println("List is empty!Try inserting first!");
            return;
        }
        Node current=head;
        if(current.data==target){
            delBeg();
            return;
        }
        Boolean flag=false;
        while(current!=null){
            if(target==current.next.data){
                flag=true;
                break;
            }
            current=current.next;
        }
        if(flag){
            current.next=current.next.next;
            display();
        }
        else{
            System.out.println(target+" data is NOT found!");
        }
    }

    /* -------------------------DELETION-END-----------------------------------------------------------------
*/
    void display(){
        if(head==null){
            System.out.println("List is empty!Try inserting first!");
            return;
        }
        Node current=head;
        while(current!=null){
            System.out.print(current.data+"->");
            current=current.next;
        }
        System.out.printf("Null\n");
    }
    /*-----------------------------DISPLAY------------------------------------------------------------------------ */
    void reverse(){
        if(head==null){
            System.out.println("List is empty!Try inserting first!");
            return;
        }
        if(head.next==null){
            return;
        }
        else{
            Node current=head;
            Node prev=null;
            Node nexti=null;
            while(current!=null){
                nexti=current.next;
                current.next=prev;
                prev=current;
                current=nexti;
            }
            head=prev;
        }
        display();
    }
    /*----------------------------REVERSE-------------------------------------------------------------- */
    void count(){
        if (head == null) {
            System.out.println("NO OF NODES: 0");
            return;
    }
    Node current=head;
    int count=0;
    if(head.next==null){
        System.out.println("NO OF NODES:1");
        return;
    }
    while(current!=null){
        count++;
        current=current.next;
    }
    System.out.println("NO OF NODES:"+count);
}
/*---------------------------------COUNT------------------------------------------------------------- */
void frequency(){
    if(head==null){
        System.out.println("List is empty!Try inserting first!");
        return;
    }
    HashMap<Integer,Integer> mapi=new HashMap<>();
    Node current=head;
    while(current!=null){
        if(mapi.containsKey(current.data)){
            mapi.put(current.data,mapi.get(current.data)+1);
        }
        else{
            mapi.put(current.data,1);
        }
        current=current.next;
    }
    System.out.println(mapi);
}
/*---------------------------------------------------------FREQUENCY------------------------ */
void concat(Node peak){
    Node current=head;
    while(current.next!=null){
        current=current.next;
    }
    current.next=peak;
    display();
}
/*-----------------------------------------------------CONCATENATION------------------------------ */
void sort(){
    if(head==null){
        System.out.println("List is empty!Try inserting first!");
        return;
    }
    display();
    ArrayList<Integer> arr=new ArrayList<>();
    Node current=head;
    while(current!=null){
        arr.add(current.data);
        current=current.next;
    }
    System.out.println(arr);
```

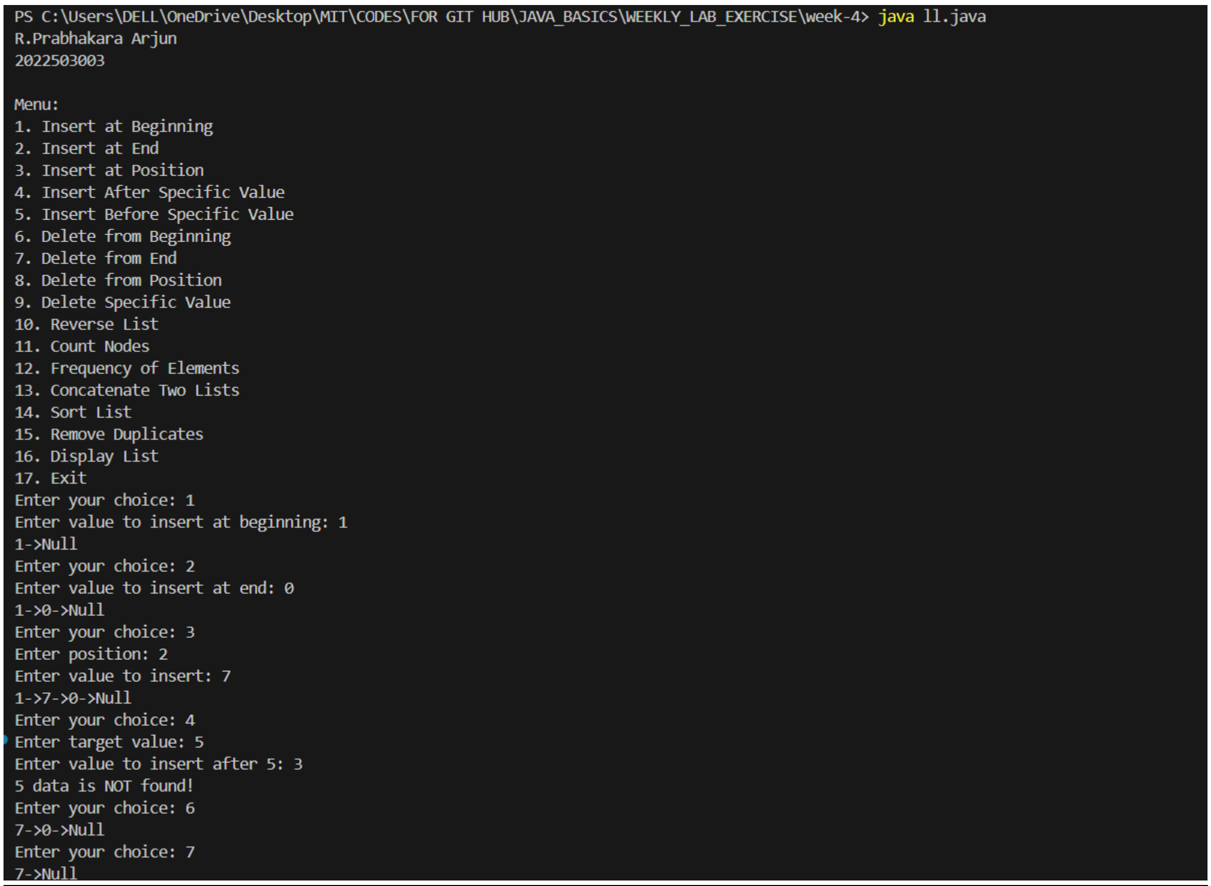
```
        Collections.sort(arr);
        System.out.println(arr);
        current=head;
        int i=0;
        while(current!=null){
            current.data=arr.get(i++);
            current=current.next;
        }
        display();
    }
    /*---------------------------------------------SORT--------------------------------------------------------*/
    void remDuplicate(){
        display();
        HashMap<Integer,Integer> mapi=new HashMap<>();
        Node current=head;
        while(current!=null){
            if(mapi.containsKey(current.data)){
                mapi.put(current.data,mapi.get(current.data)+1);
            }
            else{
                mapi.put(current.data,1);
            }
            current=current.next;
        }
        System.out.println(mapi);
        Boolean isDupi=false;
        for(int value:mapi.values()){
            if(value>=2){
                isDupi=true;
            }
        }
        if(!isDupi){
            System.out.println("No duplicates FOUND!!");
        }
        else{
           for(int key:mapi.keySet()){
                if(mapi.get(key)>1){
                    int temp=mapi.get(key);
                    while(temp!=0){
                        delVal(key);
                        temp--;
                    }
                }
        }
        }
        display();
    }
    /*------------------------------------------REMOVE DUPLICATES------------------------------- */
}
class ll{
    public static void main(String[] args) {
        System.out.println("R.Prabhakara Arjun\n2022503003");
        LinkedList link = new LinkedList();
        Scanner scanner = new Scanner(System.in);
        int choice, value, pos, target;
```

```
System.out.println("\nMenu:");
System.out.println("1. Insert at Beginning");
System.out.println("2. Insert at End");
System.out.println("3. Insert at Position");
System.out.println("4. Insert After Specific Value");
System.out.println("5. Insert Before Specific Value");
System.out.println("6. Delete from Beginning");
System.out.println("7. Delete from End");
System.out.println("8. Delete from Position");
System.out.println("9. Delete Specific Value");
System.out.println("10. Reverse List");
System.out.println("11. Count Nodes");
System.out.println("12. Frequency of Elements");
System.out.println("13. Concatenate Two Lists");
System.out.println("14. Sort List");
System.out.println("15. Remove Duplicates");
System.out.println("16. Display List");
System.out.println("17. Exit");
do {
   System.out.print("Enter your choice: ");
   choice = scanner.nextInt();

   switch (choice) {
      case 1:
         System.out.print("Enter value to insert at beginning: ");
         value = scanner.nextInt();
         link.insertBeg(value);
         break;
      case 2:
         System.out.print("Enter value to insert at end: ");
         value = scanner.nextInt();
         link.insertEnd(value);
         break;
      case 3:
         System.out.print("Enter position: ");
         pos = scanner.nextInt();
         System.out.print("Enter value to insert: ");
         value = scanner.nextInt();
         link.insertPos(pos, value);
         break;
      case 4:
         System.out.print("Enter target value: ");
         target = scanner.nextInt();
         System.out.print("Enter value to insert after " + target + ": ");
         value = scanner.nextInt();
         link.insertAfterSpecific(target, value);
         break;
      case 5:
         System.out.print("Enter target value: ");
         target = scanner.nextInt();
         System.out.print("Enter value to insert before " + target + ": ");
         value = scanner.nextInt();
         link.insertBeforeSpecific(target, value);
         break;
      case 6:
```

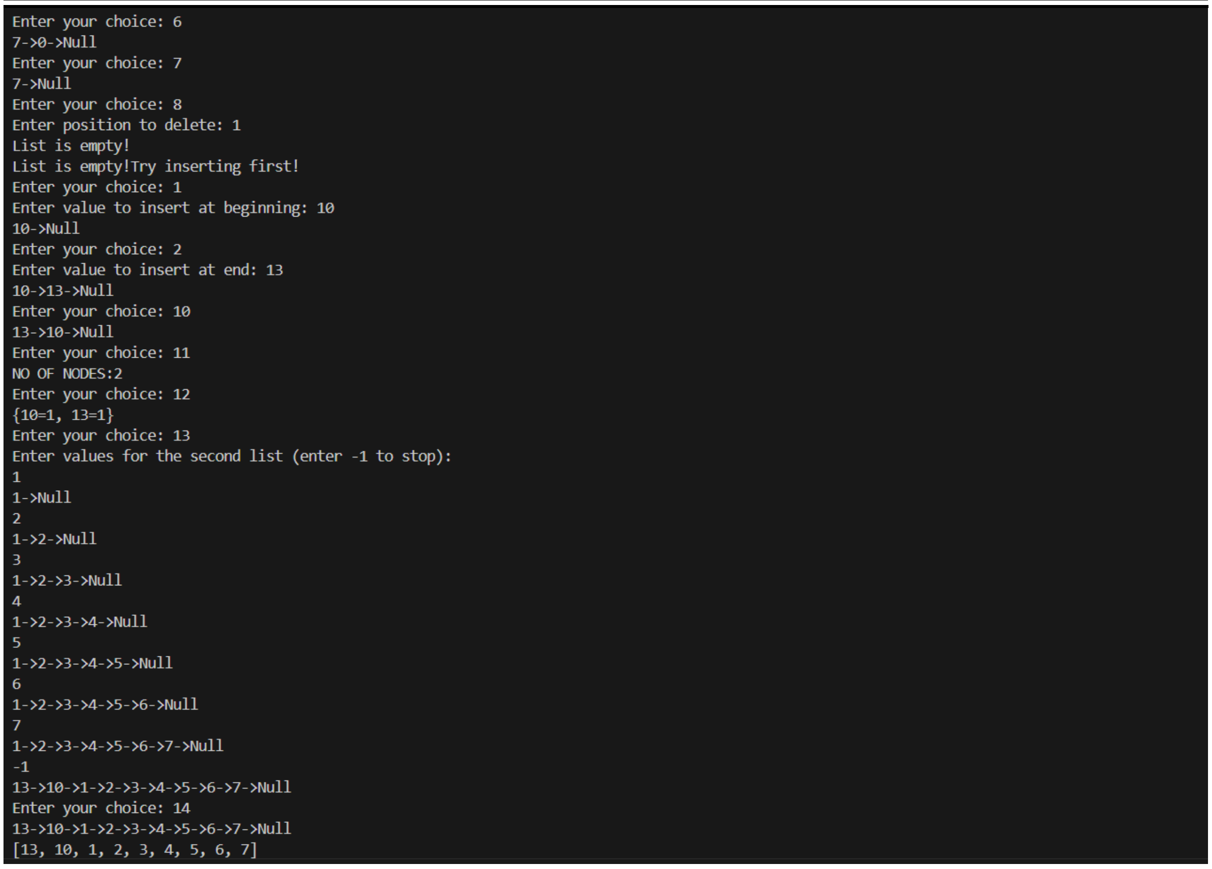
```
                    link.delBeg();
                    break;
                case 7:
                    link.delEnd();
                    break;
                case 8:
                    System.out.print("Enter position to delete: ");
                    pos = scanner.nextInt();
                    link.delPos(pos);
                    break;
                case 9:
                    System.out.print("Enter value to delete: ");
                    target = scanner.nextInt();
                    link.delVal(target);
                    break;
                case 10:
                    link.reverse();
                    break;
                case 11:
                    link.count();
                    break;
                case 12:
                    link.frequency();
                    break;
                case 13:
                    LinkedList link1 = new LinkedList();
                    System.out.println("Enter values for the second list (enter -1 to stop):");
                    while ((value = scanner.nextInt()) != -1) {
                        link1.insertEnd(value);
                    }
                    link.concat(link1.head);
                    break;
                case 14:
                    link.sort();
                    break;
                case 15:
                    link.remDuplicate();
                    break;
                case 16:
                    link.display();
                    break;
                case 17:
                    System.out.println("Exiting...");
                    break;
                default:
                    System.out.println("Invalid choice. Please try again.");
            }
        } while (choice != 17);

        scanner.close();
    }
}
```

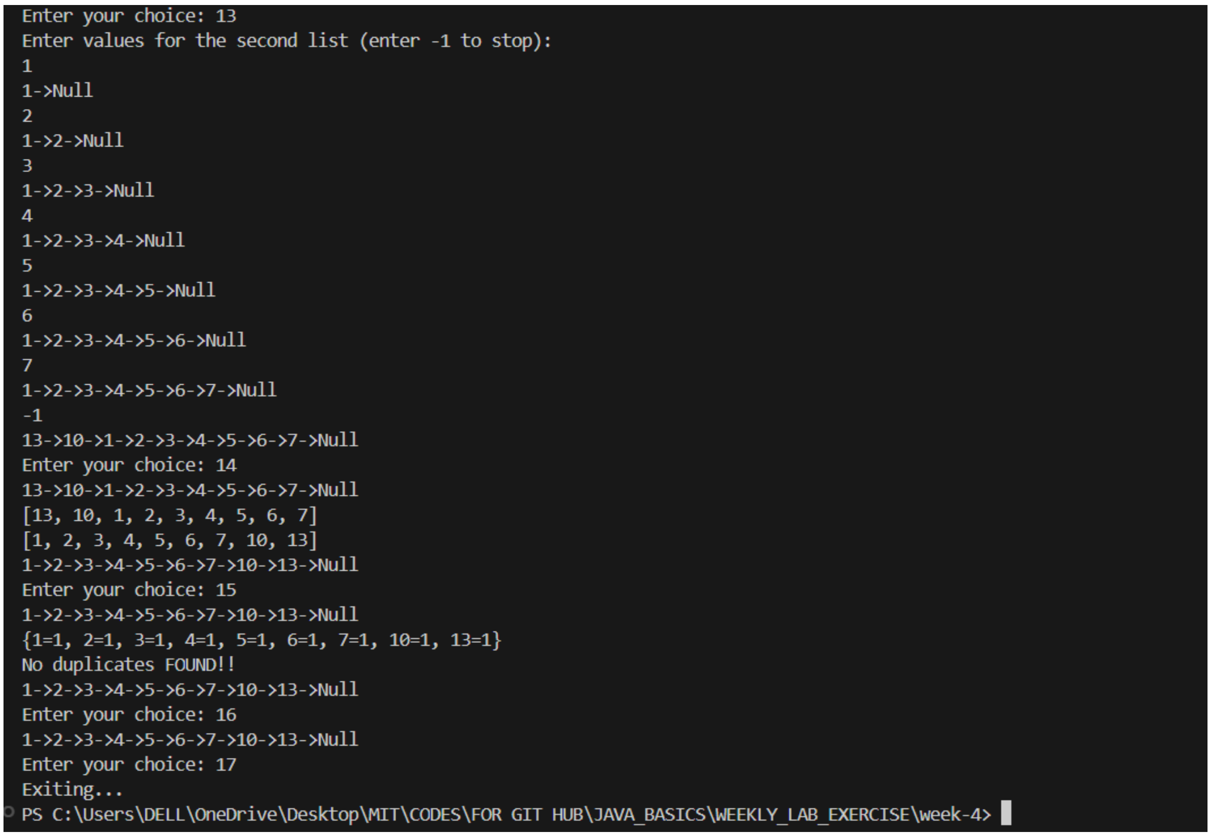


```
PS C:\Users\DELL\OneDrive\Desktop\MIT\CODES\FOR GIT HUB\JAVA_BASICS\WEEKLY_LAB_EXERCISE\week-4> java ll.java
R.Prabhakara Arjun
2022503003

Menu:
1. Insert at Beginning
2. Insert at End
3. Insert at Position
4. Insert After Specific Value
5. Insert Before Specific Value
6. Delete from Beginning
7. Delete from End
8. Delete from Position
9. Delete Specific Value
10. Reverse List
11. Count Nodes
12. Frequency of Elements
13. Concatenate Two Lists
14. Sort List
15. Remove Duplicates
16. Display List
17. Exit
Enter your choice: 1
Enter value to insert at beginning: 1
1->Null
Enter your choice: 2
Enter value to insert at end: 0
1->0->Null
Enter your choice: 3
Enter position: 2
Enter value to insert: 7
1->7->0->Null
Enter your choice: 4
Enter target value: 5
Enter value to insert after 5: 3
5 data is NOT found!
Enter your choice: 6
7->0->Null
Enter your choice: 7
7->Null
```



```
Enter your choice: 6
7->0->Null
Enter your choice: 7
7->Null
Enter your choice: 8
Enter position to delete: 1
List is empty!
List is empty!Try inserting first!
Enter your choice: 1
Enter value to insert at beginning: 10
10->Null
Enter your choice: 2
Enter value to insert at end: 13
10->13->Null
Enter your choice: 10
13->10->Null
Enter your choice: 11
NO OF NODES:2
Enter your choice: 12
{10=1, 13=1}
Enter your choice: 13
Enter values for the second list (enter -1 to stop):
1
1->Null
2
1->2->Null
3
1->2->3->Null
4
1->2->3->4->Null
5
1->2->3->4->5->Null
6
1->2->3->4->5->6->Null
7
1->2->3->4->5->6->7->Null
-1
13->10->1->2->3->4->5->6->7->Null
Enter your choice: 14
13->10->1->2->3->4->5->6->7->Null
[13, 10, 1, 2, 3, 4, 5, 6, 7]
```



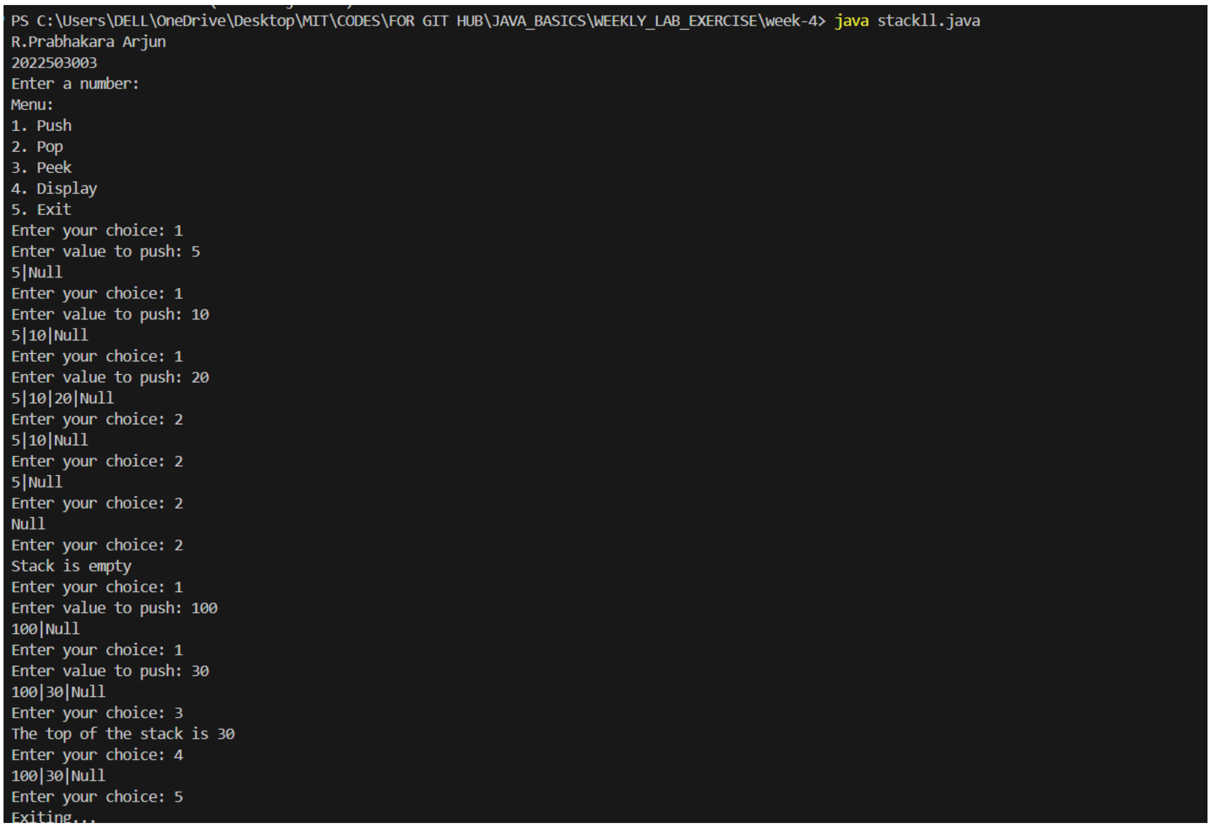
```
Enter your choice: 13
Enter values for the second list (enter -1 to stop):
1
1->Null
2
1->2->Null
3
1->2->3->Null
4
1->2->3->4->Null
5
1->2->3->4->5->Null
6
1->2->3->4->5->6->Null
7
1->2->3->4->5->6->7->Null
-1
13->10->1->2->3->4->5->6->7->Null
Enter your choice: 14
13->10->1->2->3->4->5->6->7->Null
[13, 10, 1, 2, 3, 4, 5, 6, 7]
[1, 2, 3, 4, 5, 6, 7, 10, 13]
1->2->3->4->5->6->7->10->13->Null
Enter your choice: 15
1->2->3->4->5->6->7->10->13->Null
{1=1, 2=1, 3=1, 4=1, 5=1, 6=1, 7=1, 10=1, 13=1}
No duplicates FOUND!!
1->2->3->4->5->6->7->10->13->Null
Enter your choice: 16
1->2->3->4->5->6->7->10->13->Null
Enter your choice: 17
Exiting...
PS C:\Users\DELL\OneDrive\Desktop\MIT\CODES\FOR GIT HUB\JAVA_BASICS\WEEKLY_LAB_EXERCISE\week-4>
```

2. Implement Stack using LinkedList

CODE:

```
import java.util.*;
class Node{
    int data;
    Node next;
    Node(int data){
        this.data=data;
        next=null;
    }
}
class LinkedList{
    Node head;
    Node top;            //not static bruhhh!(then all ll will have common head!)
    void push(int data){
        Node newNode=new Node(data);
        if(top==null){
            top=newNode;
            head=top;
        }
        else{
            top.next=newNode;
            top=newNode;
        }
        display();
    }
    /*---------------------------------------------INSERTION-END------------------------------*/
    void pop(){
        if(head==null) {
            System.out.println("Stack is empty");
            return;
        }
        if(head.next==null){
            head=null;
            top=null;
        }
        else{
            Node current=head;
            Node prev=current;
            while(current.next!=null){
                prev=current;
                current=current.next;
            }
            prev.next=null;
        }
        display();
    }

    /* --------------------------------------------DELETION-END-------------------------------------*/
    void display(){
        Node current=head;
        while(current!=null){
            System.out.print(current.data+"|");
            current=current.next;
        }
        System.out.printf("Null\n");
    }
    /*--------------------------------------------------DISPLAY---------------------------------- */
    void peek(){
        if(top==null){
            System.out.println("Stack is empty");
        }
        else{
            System.out.println("The top of the stack is "+top.data);
        }
    }
}
class stackll {
    public static void main(String[] args) {
        System.out.println("R.Prabhakara Arjun\n2022503003");
        System.out.print("Enter a number:");
        LinkedList stack = new LinkedList();
        Scanner scanner = new Scanner(System.in);
        int choice, value;
        System.out.println("\nMenu:");
        System.out.println("1. Push");
        System.out.println("2. Pop");
        System.out.println("3. Peek");
        System.out.println("4. Display");
        System.out.println("5. Exit");
        do {
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();
            switch (choice) {
                case 1:
                    System.out.print("Enter value to push: ");
                    value = scanner.nextInt();
                    stack.push(value);
                    break;
                case 2:
                    stack.pop();
                    break;
                case 3:
                    stack.peek();
                    break;
                case 4:
                    stack.display();
                    break;
                case 5:
                    System.out.println("Exiting...");
                    break;
                default:
                    System.out.println("Invalid choice. Please try again.");
            }
        } while (choice != 5);
        scanner.close();
    }
}
```



```
PS C:\Users\DELL\OneDrive\Desktop\MIT\CODES\FOR GIT HUB\JAVA_BASICS\WEEKLY_LAB_EXERCISE\week-4> java stackll.java
R.Prabhakara Arjun
2022503003
Enter a number:
Menu:
1. Push
2. Pop
3. Peek
4. Display
5. Exit
Enter your choice: 1
Enter value to push: 5
5|Null
Enter your choice: 1
Enter value to push: 10
5|10|Null
Enter your choice: 1
Enter value to push: 20
5|10|20|Null
Enter your choice: 2
5|10|Null
Enter your choice: 2
5|Null
Enter your choice: 2
Null
Enter your choice: 2
Stack is empty
Enter your choice: 1
Enter value to push: 100
100|Null
Enter your choice: 1
Enter value to push: 30
100|30|Null
Enter your choice: 3
The top of the stack is 30
Enter your choice: 4
100|30|Null
Enter your choice: 5
Exiting...
```