LAB ASSIGNMENT-4

Q1. Menu-driven program on SLL that have the methods functions: Create, Display, Insert at Beginning, End, any position, Delete from Beginning, End and from any position., Search, Count, Sort, Reverse the LL.

Ans.

```
import java.util.*;
class Node
{
    int info;
    Node next;
}
public class SingleLinkedList
{
    static node start = null;
    public static void create()
    {
        Scanner sc = new Scanner(System.in);
        node p = new node();
        System.out.print("Enter info:");
        p.info = sc.nextInt();
        p.next = null;
        start = p;
        node q = p;
        System.out.print("Do you want to add more:");
```

```
        char ch = sc.next().charAt(0);
        while (ch != 'n' || ch != 'N')
        {
            p = new node();
            System.out.print("enter info:");
            p.info = sc.nextInt();
            p.next = null;
            start = p;
            q = p;
            System.out.print("Do you want to add
            more:");
            ch = sc.next().charAt(0);
        }
    }
    public static void display()
    {
        if (start == null)
        {
            System.out.println("Linked list is empty");
        }
        else
        {
            node p = start;
            while (p != null)
            {
                System.out.print(p.info + "→");
                p = p.next;
            }
        }
    }
```

```
public static void insend()
{
    Scanner sc=new Scanner(System.in);
    node p=new node();
    System.out.println("Enter info:");
    p.info=sc.nextInt();
    p.next=null;
    if (start==null)
    {
        start=p;
    }
    else
    {
        node t=start;
        while (t.next!=null)
        {
            t=t.next;
        }
        t.next=p;
    }
}
public static void insbegi()
{
    Scanner sc=new Scanner(System.in);
    node p=new node();
    System.out.println("Enter info:");
    p.info=sc.nextInt();
    p.next=start;
}
```

```
public static void insany()
{
    Scanner sc = new Scanner (System.in);
    System.out.printf ("Enter position: ");
    int pos = sc.nextInt();
    int count = countnodes();
    if (pos > count+1)
    {
        System.out.println ("Insertion not
        possible ");
    }
    else
    {
        if (pos == 1)
        {
            insbeg();
        }
        else if (pos == count+1)
        {
            insend();
        }
        else
        {
            node p = new node();
            System.out.println ("Enter info:");
            p.info = sc.nextInt();
            node t = start;
            for (int i=0 ; i<=pos-2; i++)
            {
                t = t.next;
            }
```

```
            p.next = t.next;
            t.next = p;
        }
    }
}
public static void delbeg()
{
    if (start==null)
    {
        System.out.println("Linked List is empty");
    }
    else
    {
        node q = start;
        start = q.next;
        q.next = null;
        System.out.println("Deleted node is "+
        q.info);
    }
}
public static void delend()
{
    if (start==null)
    {
        System.out.println("Linked List is empty");
    }
    else if (start.next== null)
    {
```

```
        node q = start;
        start = q.next;
        q.next = null;
        System.out.println("Deleted node is " +
        q.info);
    }
    else
    {
        node q = start;
        while ((q.next).next != null)
        {
            q = q.next;
        }
        System.out.println("Deleted node is " +
        q.next.info);
    }
}
public static void delany()
{
    if (start == null)
    {
        System.out.println("Linked List is empty");
    }
    else
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter position:");
        int pos = sc.nextInt();
        int count = countnodes();
```

```
if (pos > count)
{
    System.out.println("Deletion not
    possible");
}
else
{
    if (pos==1)
    {
        delbeg();
    }
    else if (pos == count)
    {
        ~~insert~~delend();
    }
    else
    {
        node q = start;
        for (int i=1; i<=pos-2; i++)
        {
            q=q.next;
        }
        node t = q.next;
        q.next= t.next;
        t.next = null;
        System.out.println("Deleted node is "
        + t.info);
}
```

```
    }
}
public static int countnodes()
{
    int count=0;
    node p=start;
    while(p!=null)
    {
        count+=1;
        p=p.next;
    }
    return count;
}
public static void sort()
{
    for(node p=start;p.next!=null;p=p.next)
    {
        for(node q=p.next;q!=null;q=q.next)
        {
            if(p.info>q.info)
            {
                int t=p.info;
                p.info=q.info;
                q.info=t;
            }
        }
    }
}
```

```
public static void search()
{
  Scanner sc = new Scanner (System.in);
  System.out.println(" Enter the element
  you want to search:");
  int ele = sc.nextInt();
  int pos = -1;
  int i = 0;
  node q = start;
  while (q != null)
  {
    i++;
    if (q.info == ele)
    {
      pos = i;
      System.out.println(" Element found
      in position: " + pos);
    }
    q = q.next;
  }
  if (pos == -1)
  {
    System.out.println("Search unsuccessful");
  }
}
public static void main (String[] args)
{
```

```
Scanner sc = new Scanner (System.in);
while (true)
{
  System.out.println (" MENU DRIVEN PROGRAM");
  System.out.println (" 0 -> Exit");
  System.out.println (" 1 -> Creation");
  System.out.println (" 2 -> Display");
  System.out.println (" 3 -> Insert node at beginning");
  System.out.println (" 4 -> Insert node at end");
  System.out.println (" 5 -> Insert node at any
position :");
  System.out.println (" 6 -> Delete node at beginning");
  System.out.println (" 7 -> Delete node at end");
  System.out.println (" 8 -> Delete node at any position :");
  System.out.println (" 9 -> Count the no. of nodes");
  System.out.println (" 10 -> Sorting the nodes");
  System.out.println (" 11 -> Searching the nodes :");
  System.out.println (" 12 -> Reverse the node");
  System.out.println (" Enter the choice :");
  int choice = sc.nextInt();
  switch (choice)
  {
  case 0 :
       System.exit(0);
       break;
  case 1 :
       create();
       break;
```

```
case 2:
    display();
    break;
case 3:
    insbeg();
    break;
case 4:
    insend();
    break;
case 5:
    insany();
    break;
case 6:
    delbeg();
    break;
case 7:
    delend();
    break;
case 8:
    delany();
    break;
case 9:
    countnodes();
    break;
case 10:
```

```
                sort();
                break;
            case 11:
                search();
                break;
            default:
                System.out.println("Wrong choice");
                break;
            }
        }
    }
}
```