LAB ASSIGNMENT-5

Q Write a menu-driven program on DLL that have method functions: Create, display, Insertion at Beginning, end, any position, Delete from Beginning, end any position, count the no. of nodes. search the node, Sorting of LL.

Ans.

```
import java.util.*;
class node
{
    node prev;
    int info;
    node next;
}
public class DoubleLinkedList
{
    static node start = null;
    static node end = 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;
        p.prev = null;
        start = p;
        end = p;
        System.out.println("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;
            end.next = p;
            p.prev = end;
            end = p;
            start = p;
            System.out.println("Do you want
            to add more:");
            ch = sc.next().charAt(0);
        }
    }
    public static void insbeg()
    {
        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)
        {
            p.prev = null;
            start = p;
            end = p;
        }
        else
        {
            p.prev = end;
            end.next = p;
            end = p;
        }
    }
    public static void insend()
    {
        Scanner sc = new Scanner(System.in);
        node p = new node();
        System.out.println("Enter info:");
        p.info = sc.nextInt();
        p.prev = null;
        p.next = start;
        start.prev = p;
        start = p;
    }
```

```
public static void insany()
{
  Scanner sc=new Scanner (System.in);
  node p= new node();
  System.out.println(" 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 q= new node();
      System.out.println("Enter info:");
      q.info=sc.nextInt();
      p.next=null;
      if (start==null)
```

```
            {
              q.prev = null;
              start = q;
              end = q;
            }
            else
            {
              q.prev = end;
              end.next = q;
              end = q;
            }
        }
    }
    p.prev = null;
    p.next = start;
    start.prev = p;
    start = p;
}
public static void display()
{
    if (end == null)
    {
        System.out.println("Linked List is
        empty");
    }
    else
    {
```

```
        node p = end;
        while (p != null)
        {
          System.out.print(p.info + " → ");
          p = p.prev;
        }
    }
}
public static int countnodes()
{
    int count = 0;
    node p = start;
    while (p != null)
    {
      count += 1;
      p = p.next;
    }
    return count;
}
public static void delbeg()
{
  if (start == null)
  {
    System.out.println("Linked List is empty");
  }
  else if (start.next == null)
  {
    System.out.println("Deleted node is " +
    start.info);
```

```
            start = null;
            end = null;
        }
        else
        {
            node q = start;
            q.next.prev = null;
            start = start.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 = end;
            q.prev.next = null;
            end = q.prev;
            q.next = null;
            System.out.println("Deleted node is " +
            q.info);
            q.prev = null;
            System.out.println("Deleted node is " +
            q.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 the position:");
            int pos = countnodes();
            if (pos > count)
            {
                System.out.println("Deletion not possible");
            }
```

```
else
{
  if (pos==1)
  {
    delbegi();
  }
  else if (pos == count)
  {
    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.prev=null;
    t.next=null;
    System.out.println("Deleted node
    is"+t.info);
  }
 }
}
```

```
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 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 from beg");
    System.out.println("4 → Insert node from end");
    System.out.println("5 → Insert node from any pos");
    System.out.println("6 → Delete node from beginning");
```

```
System.out.println("7 -> Delete node from end");
System.out.println("8 -> Delete node from any pos");
System.out.println("9 -> Count no. of nodes");
System.out.println("10 -> Sorting the nodes");
System.out.println("11 -> Searching the nodes");
System.out.println("Enter the choice");
int choice = sc.nextInt();
switch(choice)
{
case 0:
    System.out.exit(0);
    break;
case 0:
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;
}
}
}
}
```