Experiment:11

Aim: To implement Data Structures as Linked List
Write a Java Program to implement all operations of Singly Linked List.

Program:

```
import java.util.Scanner;
class Node//  Class Node
{
   protected int data;
   protected Node link;
   public Node()//  Constructor
   {   link = null;
       data = 0;}
   public Node(int d,Node n)//  Constructor
   {data = d;
    link = n;}
   //  Function to set link to next Node
   public void setLink(Node n)
   {link = n;}
   //  Function to set data to current Node
   public void setData(int d)
   {data = d;}
   //  Function to get link to next node
   public Node getLink()
   {return link;}
   //  Function to get data from current Node
   public int getData()
   {return data;}
}
// Class linkedList
class linkedList
{
   protected Node start;
   protected Node end ;
   public int size ;
```

```
    // Constructor
    public linkedList()
    {
        start = null;
        end = null;
        size = 0;
    }
//Function to check if list is empty
   public boolean isEmpty()
    {
        return start == null;
    }
    //  Function to get size of list
    public int getSize()
    {
        return size;
    }
   //  Function to insert an element at begining
    public void insertAtStart(int val)
    {
        Node nptr = new Node(val, null);
        size++ ;
       if(start == null)
        {   start = nptr;
            end = start;   }
        else
        {   nptr.setLink(start);
            start = nptr;      }
    }
    //  Function to insert an element at end
    public void insertAtEnd(int val)
    {  Node nptr = new Node(val,null);
        size++ ;
        if(start == null)
        {
            start = nptr;
            end = start;        }
        else
        {
            end.setLink(nptr);
            end = nptr;
        }
    }
    //  Function to insert an element at position
    public void insertAtPos(int val , int pos)
    {
        Node nptr = new Node(val, null);
        Node ptr = start;
       pos = pos - 1 ;
        for (int i = 1; i < size; i++)
        {
            if (i == pos)
            {
                Node tmp = ptr.getLink() ;
                ptr.setLink(nptr);
                nptr.setLink(tmp);
                break;
            }
           ptr = ptr.getLink();
        }
        size++ ;
    }
    //  Function to delete an element at position
    public void deleteAtPos(int pos)
    {
        if (pos == 1)
        {
            start = start.getLink();
            size--;
            return ;    }
        if (pos == size)
        {
            Node s = start;
            Node t = start;
            while (s != end)
            {
                t = s;
                s = s.getLink();
            }
            end = t;
            end.setLink(null);
            size --;
            return;
        }
        Node ptr = start;
       pos = pos - 1 ;
        for (int i = 1; i < size - 1; i++)
        {
            if (i == pos)
            {
                Node tmp = ptr.getLink();
                tmp = tmp.getLink();
                ptr.setLink(tmp);
                break;
            }
           ptr = ptr.getLink();
       }
       size-- ;
    }
    //  Function to display elements
    public void display()
    {
            System.out.print("\nSingly Linked List = ");
            if (size == 0)
            {System.out.print("empty\n");
               return;  }
            if (start.getLink() == null)
            {  System.out.println(start.getData() );
               return;  }
            Node ptr = start;
             System.out.print(start.getData()+ "->");
             ptr = start.getLink();
             while (ptr.getLink() != null)
             {
                System.out.print(ptr.getData()+ "->");
                ptr = ptr.getLink();
             }
             System.out.print(ptr.getData()+ "\n");
         }
    }
    //  Class SinglyLinkedList
    public class SinglyLinkedList
    {
        public static void main(String[] args)
        {
            Scanner scan = new Scanner(System.in);
            // Creating object of class linkedList
            linkedList list = new linkedList();
            System.out.println("Singly Linked List Test\n");

            char ch;
            //  Perform list operations
            do
            {
                System.out.println("\nSingly Linked List Operations\n");
                System.out.println("1. insert at begining");
    System.out.println("2. insert at end");
    System.out.println("3. insert at position");
    System.out.println("4. delete at position");
    System.out.println("5. check empty");
    System.out.println("6. get size");
    int choice = scan.nextInt();
    switch (choice)
    {
    case 1 :
        System.out.println("Enter integer element to insert");
        list.insertAtStart( scan.nextInt() );
        break;
    case 2 :
        System.out.println("Enter integer element to insert");
        list.insertAtEnd( scan.nextInt() );
        break;
    case 3 :
        System.out.println("Enter integer element to insert");
        int num = scan.nextInt() ;
        System.out.println("Enter position");
        int pos = scan.nextInt() ;
        if (pos <= 1 || pos > list.getSize() )
            System.out.println("Invalid position\n");
        else
            list.insertAtPos(num, pos);
        break;
    case 4 :
        System.out.println("Enter position");
        int p = scan.nextInt() ;
        if (p < 1 || p > list.getSize() )
            System.out.println("Invalid position\n");
        else
            list.deleteAtPos(p);
        break;
```

```
            case 5 :
                System.out.println("Empty status = "+ list.isEmpty());
                break;
            case 6 :
                System.out.println("Size = "+ list.getSize() +" \n");
                break;
            default :
                System.out.println("Wrong Entry \n ");
                break;
            }
           list.display();//  Display List
            System.out.println("\nDo you want to continue (Type y or n) \n");
            ch = scan.next().charAt(0);
        } while (ch == 'Y'|| ch == 'y');
    }
}
```