\\Lab Assignment_4

CSE 2001: Data Structure & Algorithms
Programming Assignment-IV
(Singly Linked List)

P.S-This Assignment can be done in 3-4 ways..(Haid-tail approach ,Next appraoch,size
In Google u may found a different way ,in ChatGpt it may differ, So please don't confuse urself....

Only understand what is "next" in node, just make it crystal clear.. beacuse in question they have asked to go by the Node next way...

I have tried my best to make as much as simple..
Also Most of the methods are Efficient time complexity..

\\By Ansuman Swain

```
package Assignment_4;
import java.util.*;
class Node
{
protected int regd_no;
protected float mark;
protected Node next;
Node(int regd_no, float mark)
    {
        this.regd_no=regd_no;
        this.mark=mark;
        this.next=null;
    }
    public float getMark()
    {
        return mark;
    }
    public int getRegd_no()
    {
        return regd_no;
    }
}
public class LinkedList
{
    static Scanner obj = new Scanner(System.in);
    public  static Node create(Node start)
    {
                System.out.print("Enter Regd_no :");
                int rgno=obj.nextInt();
                System.out.print("Enter Marks :");
                float mark=obj.nextFloat();
                start = new Node(rgno,mark);
                System.out.println("Linkedlist created...");
                return start;
    }
```

```
public static Node InsBeg(Node start)
   {
       if(start==null)
       {
           start=create(start);
           return start;
       }
       else
       {
           System.out.print("Enter Regd_no : ");
           int r=obj.nextInt();
           System.out.print("Enter Marks : ");
           float m=obj.nextFloat();
           Node newNode = new Node(r,m);
           newNode.next=start;
           System.out.println("Node inserted at the beginning...");
           return newNode;
       }
   }
   public static Node InsEnd(Node start)
   {
       if(start==null)
       {
           start=create(start);
           return start;
       }
       else
       {
           System.out.print("Enter Regd_no : ");
           int r=obj.nextInt();
           System.out.print("Enter Marks : ");
           float m=obj.nextFloat();
           Node newNode = new Node(r,m);
           Node temp=start;
           while(temp.next!=null)
               temp=temp.next;
           temp.next=newNode;
           System.out.println("Node inserted at the end...");
           return start;
       }
   }
```

\\By Ansuman Swain

```
public static Node InsAny(Node start)
    {
        System.out.print("Enter the position to insert at: ");
        int position = obj.nextInt();
        Node temp = start;
        if(position==1)
        {
           return InsBeg(start);
        }
        else
        {
            for(int i=2;i<position;i++)
            {
                try{
                    temp = temp.next;
                    if(temp==null)
                        throw new NullPointerException();
                    }
                    catch(NullPointerException e)
                    {
                        System.out.println("Invalid position...");
                        return start;
                    }
            }
            System.out.print("Enter Regd_no : ");
            int r = obj.nextInt();
            System.out.print("Enter mark: ");
            float m = obj.nextFloat();
            Node newNode = new Node(r,m);
            newNode.next = temp.next;
            temp.next = newNode;
            System.out.println("Node inserted at "+position+" position");
            return start;
        }
    }
        public static Node DelBeg(Node start)
    {
        if(start==null)
            System.out.println("Nothing in the list to delete...");
        else
        {
            start=start.next;
             System.out.println("Node deleted at the beginning...");
        }
            return start;

    }
```

\\By Ansuman Swain

```
public static Node DelEnd(Node start)
    {
        if(start==null)
            System.out.println("Nothing in the list to delete...");
        else if (start.next == null)
        {
        start = null;
        System.out.println("End Node deleted....");
        }
        else
        {
        Node temp = start;
        while (temp.next.next!= null)
            temp = temp.next;
        temp.next=null;
        System.out.println("End Node deleted....");
        }
    return start;
    }
    public static Node DelAny(Node start)
    {
        System.out.print("Enter the position to delete at: ");
        int position = obj.nextInt();
        if (position == 1)
        {
            start=start.next;
             System.out.println("Node deleted at first position...");
            return start;
        }
        else
        {
            Node temp = start;
            for(int i=2;i<position;i++)
            {
                try{
                    temp = temp.next;
                    if(temp==null)
                        throw new NullPointerException();
                    }
                    catch(NullPointerException e)
                    {
                        System.out.println("Invalid position,Nothing to
                                            delete...");
                        return start;
                    }
            }
                Node jumper =temp;
                jumper=jumper.next;
                temp.next = jumper.next;
                System.out.println("Node deleted Successfully...");
                return start;
        }
    }
```

\\By Ansuman Swain

```
public static Node DelbyRegd(Node start)
    {
        if (start == null)
        {
            System.out.println("Linked list is empty..Nothing to delete.");
            return start;
        }
        System.out.print("Enter the regd_no of the node to delete: ");
        int regd_no = obj.nextInt();
        Node current = start;
        Node previous = null;
        if (start.regd_no == regd_no)
        {

            System.out.println("Node with regd_no " + regd_no + " deleted
successfully.");
             return start.next;
        }
        boolean key=true;
        while (current != null)
        {
            if (current.regd_no == regd_no)
            {
                previous.next = current.next;
                current.next = null;
                System.out.println("Node with regd_no " + regd_no + " deleted
successfully.");
                key=false;
                return start;
            }
            previous = current;
            current = current.next;
        }
        try{
            if(key)
            throw new NullPointerException();
        }
        catch(NullPointerException e)
        {
            System.out.println("Node with regd_no " + regd_no + " not found in the
list.");
        }
        return start;
    }
```

\\By Ansuman Swain

```
public static void search(Node start)
    {
        if (start == null)
            System.out.println("List is empty. Nothing to search.");
        else
        {
            System.out.print("Enter the regd_no for search : ");
            float regd_no = obj.nextFloat();
            Node temp = start;
            boolean found = false;
            while (temp != null)
            {
                if (temp.regd_no == regd_no)
                {
                    System.out.print("Enter the new mark for the student: ");
                    float newMark = obj.nextFloat();
                    temp.mark = newMark;
                    System.out.println("Mark updated.....");
                    found = true;
                }
                temp = temp.next;
            }
                try
                {
                    if(!found)
                        throw new Exception("Not found any node having this
regd_no...");
                }
                catch(Exception e)
                {
                    System.out.println(e.getMessage());
                }
            }
    }
    public static int count(Node start)
    {
        int count=0;
        Node temp=start;
        while(temp!=null)
        {
            count++;
            temp=temp.next;

        }
        return count;
    }
```

\\By Ansuman Swain

```
public static void sort(Node start)
   {
       if (start==null)
           System.out.println("Linked List is empty... ");
       else if(start.next==null)
           System.out.println("Linked list is sorted...");
       else
       {
           boolean swap;
           Node temp;
           Node lastNode=null;
           do
           {
               swap = false;
               temp = start;
               while (temp.next != lastNode)
               {
                   if (temp.mark < temp.next.mark)
                   {
                       int tempRegdNo = temp.regd_no;
                       float tempMark = temp.mark;
                       temp.regd_no = temp.next.regd_no;
                       temp.mark = temp.next.mark;
                       temp.next.regd_no = tempRegdNo;
                       temp.next.mark = tempMark;
                       swap = true;
                 }
                       temp = temp.next;
                 }
                 lastNode = temp;
           } while (swap);
           System.out.println("Linked list sorted according to marks in Ascending
                              order...");
       }

   }
   public static Node reverse(Node start)
   {
       Node reversed = null;
       Node present = start;
       Node hold;
       while (present != null)
       {
           hold = present.next ;
           present.next = reversed;
           reversed = present;
           present = hold;
       }
       System.out.println("List reversed successfully!");
       return reversed;
   }
       public static void display(Node start)
   {
       Node temp=start;
       if(temp==null)
       {
       System.out.println("Empty link list...");
       }
       else
       {
       System.out.println("Regd.no\t\tMarks");
       while(temp!=null)
       {
           System.out.println(temp.regd_no+"\t\t"+temp.mark);
           temp=temp.next;
       }
       System.out.println("All printed....");
       }
   }
```

\\By Ansuman Swain

```
public static void main(String[] args)
    {
        Node start=null;
            System.out.println("\n******** MENU ********");
            System.out.println("0: Exit");
            System.out.println("1: Creation");
            System.out.println("2: Display");
            System.out.println("3: Insert at the beginning");
            System.out.println("4: Insert at the end");
            System.out.println("5: Insert at any position");
            System.out.println("6: Delete from the beginning");
            System.out.println("7: Delete from the end");
            System.out.println("8: Delete from any position");
            System.out.println("9:Delete by student regd_no.");
            System.out.println("10: Search and update mark");
            System.out.println("11: Sort the list");
            System.out.println("12: Count the number of nodes ");
            System.out.println("13: Reverse the Linked list");
            //u can use these inside loop ..but here in the output it ll be big so.
        while (true)
        {
            System.out.print("\nEnter your choice: ");
            int choice = obj.nextInt();
            System.out.println();
             switch (choice)
             {
                case 0:
                System.out.println("Thank you...\n Have a great day..");
                    System.exit(0);
                case 1:
                    start=create(start);
                    break;
                case 2:
                    display(start);
                    break;
                case 3:
                    start = InsBeg(start);
                    break;
                case 4:
                    start = InsEnd(start);
                    break;
                case 5:
                    start = InsAny(start);
                    break;
                case 6:
                    start = DelBeg(start);
                    break;
                case 7:
                    start = DelEnd(start);
                    break;
                case 8:
                    start = DelAny(start);
                    break;
                case 9:
                    start=DelbyRegd(start);
                    break;
                case 10:
                    search(start);
                    break;
                case 11:
                    sort(start);
                    break;
                case 12:
                    System.out.println("Number of nodes in the list: " +
count(start));
                    break;
                case 13:
                    start = reverse(start);
                    break;
                default:
                    System.out.println("Wrong choice!...please enter a valid
choice...");
                    break;
            }
        }
    }
}
```