Name: **ZOHAIB HASSAN SOOMRO**

RollNo#: **19SW42**

Subject: **DSA**

**Task 1: Write and test the following methods:**

**1. int search(int x)** //returns the index of element in the linked list

2. **int size()** //returns the size of the linked list

**3. int sum()** //returns the sum of all numbers in the linked list

**4. void deleteLast()** //deletes the last node in the list

**5. LinkedList copy()** //returns a new linked list that is the duplicate of the list this method is called up on.

**6. LinkedList subList(int p, int q)** //returns a new linked list that contains element from node p to node q of the list this method is called up on.

**7. void append(LinkedList l)** //l is appended to the list this method is called on.

**8. LinkedList merged(Linked l)** // returns a new list that merges l with the list the method is called on; maintaining the ascending order.

**Code:**

**class LinkedList{**

**Node start;**

**private class Node{**

**private int data;**

**private Node next;**

**public Node(int data){**

**this.data=data;**

**}**

**public Node(int data,Node next){**

**this.data=data;**

**this.next=next;**

**}**

**}**

**/////////////Method to print elements of LinkedList**

**public void printList(){**

**System.out.print("[");**

**for (Node p=start;p!=null;p=p.next){**

**System.out.print(p.data+",");**

**}**

**System.out.println("\b]");**

**}**

**/////////////Method to insert an element in LinkedList**

**public Node insert(int value){**

**if(start==null || start.data>value){**

**start= new Node(value,start);**

**return start;**

**}**

**Node p= start;**

**while (p.next!=null) {**

**if(p.next.data>value)**

**break;**

**p=p.next;**

**}**

**p.next= new Node(value,p.next);**

**return start;**

**}**

**/////////////Method to delete an element from LinkedList**

**public Node delete(int value){**

**if(start==null || start.data>value)**

**return start;**

**if(start.data==value){**

**System.out.println("Deleting element: "+start.data);**

**start= start.next;**

**return start;**

**}**

**Node p=start;**

**while((p=p.next)!=null){**

**if(p.next.data>value) break;**

**if(p.next.data==value){**

**System.out.println("Deleting element: "+p.next.data);**

**p.next=p.next.next;**

**break;**

**}**

**}**

**return start;**

**}**

**/////////////Method to search for an element in LinkedList**

**public int search(int element){**

**if(start==null)**

**throw new IllegalStateException("List is empty!");**

**if(start.data>element)**

**return -1;**

**Node p=start;**

**int index=1; //we will do indexing from 1 unlike arrays**

**while((p=p.next)!=null){**

**index++; //increment before if condition bcz one element(i.e start.data) is already checked above**

**if(p.data==element)**

**return index;**

**}**

**return -1;**

**}**

**/////////////Method to return total number of elements in LinkedList**

**public int size(){**

**if(start==null)**

**return 0;**

**int size=0;**

**for (Node p=start;p!=null;p=p.next) {**

**size++;**

**}**

**return size;**

**}**

**/////////////Method to return sum of all elements in LinkedList**

**public int sum(){**

**if(start==null)**

**return 0;**

**int sum=0;**

**for (Node p=start;p!=null;p=p.next) {**

**sum+=p.data;**

**}**

**return sum;**

**}**

**/////////////Method to delete last element of LinkedList**

**public void deleteLast(){**

**if(start==null)**

**throw new IllegalStateException("List is Empty!");**

**Node p=start;**

**while(p.next.next!=null){**

**p=p.next;**

**}**

**System.out.println("Deleting Last element: "+p.next.data);**

**p.next=null;**

**}**

**/////////////Method to return copy of current LinkedList Object**

**public LinkedList copy(){**

**if(start==null)**

**throw new IllegalStateException("Can not copy because current List is Empty!");**

**LinkedList list= new LinkedList();**

**Node p=start;**

**while(p!=null){**

**list.insert(p.data);**

**p=p.next;**

**}**

**return list;**

**}**

**/////////////Method to return sublist of current LinkedList from Node p to Node q**

**public LinkedList subList(int p,int q){**

**if(start==null)**

**throw new IllegalStateException("Can not copy because current List is Empty!");**

**if(p<1 || q<1 || p>this.size() || q>this.size())**

**throw new IllegalArgumentException("Invalid arguments!");**

**LinkedList list=new LinkedList();**

**int fromTill=1; //As indexing is from 1**

**Node node=start;**

**while(node!=null){**

**if(fromTill>=p && fromTill<=q)**

**list.insert(node.data);**

**if(fromTill==q) break;**

**node=node.next;**

**fromTill++;**

**}**

**return list;**

**}**

**/////////////Method to append argument LinkedList to current LinkedList**

**public void append(LinkedList list){**

**if(list.size()==0) return;**

**Node p= list.start;**

**while(p!=null){**

**this.insert(p.data);**

**p=p.next;**

**}**

**}**

**/////////////Method to merge argument LinkedList & current LinkedList and return new LinkedList**

**public LinkedList merged(LinkedList list){**

**if(size()==0 && list.size()==0)**

**return new LinkedList(); //return new empty list both current & argument lists are empty**

**LinkedList newList= new LinkedList();**

**Node p= start;**

**while(p!=null){**

**newList.insert(p.data);**

**p=p.next;**

**}**

**p=list.start;**

**while(p!=null){**

**newList.insert(p.data);**

**p=p.next;**

**}**

**return newList;**

**}**

**////////////main method starts here**

**public static void main(String[] args) {**

**System.out.println("\t\*\*\*NOTE: For indexing i have used 1 as starting index of elements in LinkedList\*\*\*\*");**

**LinkedList list= new LinkedList();**

**list.insert(9);**

**list.insert(4);**

**list.insert(5);**

**list.insert(10);**

**list.insert(1);**

**System.out.print("List: ");**

**list.printList();**

**list.delete(5);**

**System.out.print("List: ");**

**list.printList();**

**System.out.println("\nIndex of 9 = "+list.search(9));**

**System.out.println("Size of LinkedList = "+list.size());**

**System.out.println("Sum of elements of LinkedList = "+list.sum()+"\n");**

**list.insert(12);**

**System.out.print("List: ");**

**list.printList();**

**list.deleteLast();**

**System.out.print("After Deleting last element of LinkedList the List= ");**

**list.printList();**

**LinkedList copiedList= list.copy();**

**System.out.print("\nList= ");**

**list.printList();**

**System.out.print("Copied List= ");**

**copiedList.printList();**

**list.insert(13);**

**list.insert(6);**

**list.insert(8);**

**System.out.print("\nList: ");**

**list.printList();**

**LinkedList sublist= list.subList(2,4); //copies from 2nd element till 4th element**

**System.out.print("from 2nd element till 4th element...\nSublist= ");**

**sublist.printList();**

**list.append(sublist);**

**System.out.print("\nAfter appending sublist in Original List= ");**

**list.printList();**

**LinkedList mergedList= copiedList.merged(sublist);**

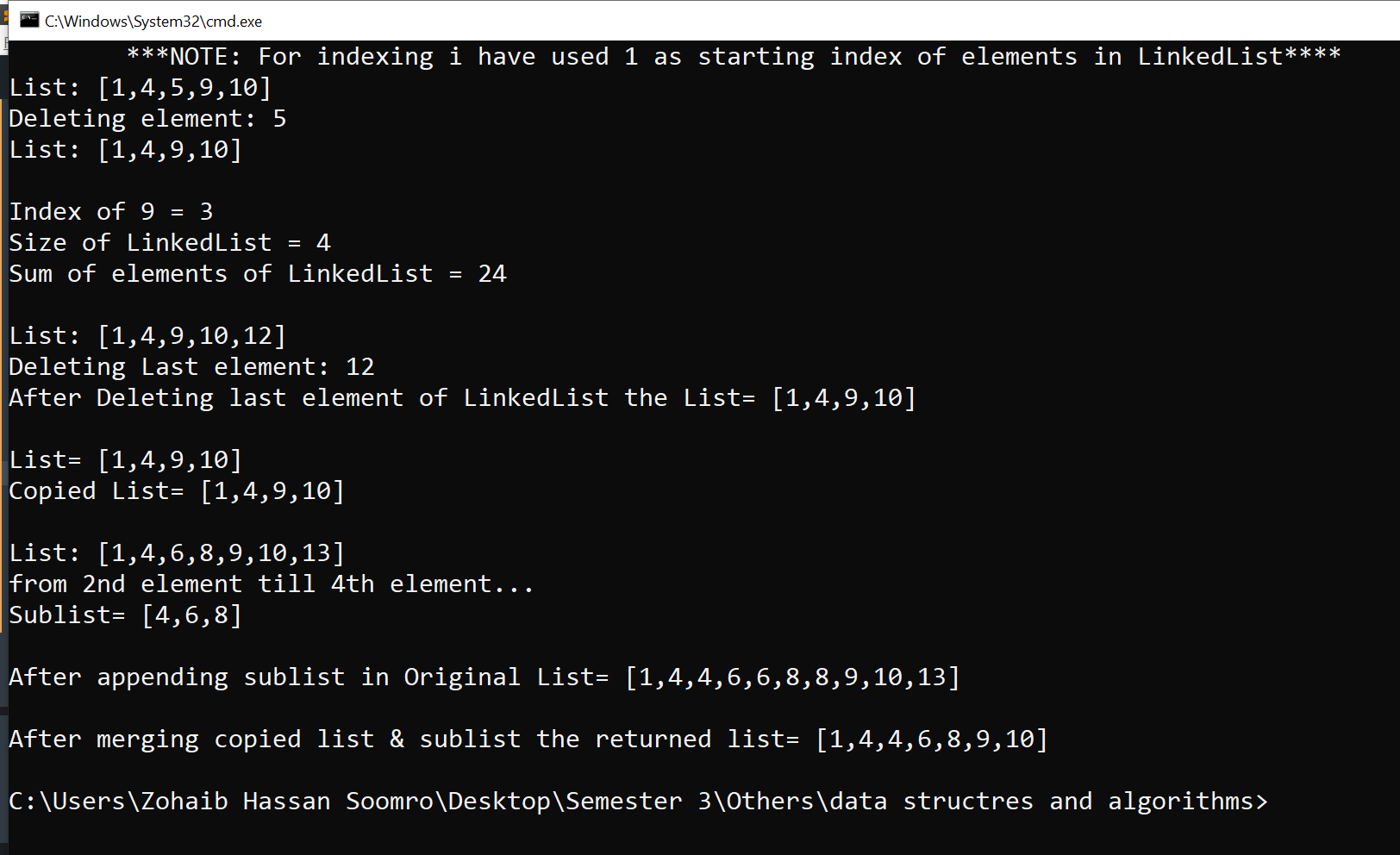
**System.out.print("\nAfter merging copied list & sublist the returned list= ");**

**mergedList.printList();**

**}**

**}**

**Task#1 Output:**



Task 2: Implement a linked list for Student class. Student class should have roll\_num and name as instance variables. The linked list should have the following operations:

◦ **insert(Student s)**

**◦ delete(Student s)**

**◦ printList()** // print the roll numbers and names of every student in the list.

**Code:**

**class Student{**

**String rollNumber,name;**

**public Student(String rollNumber, String name){**

**this.rollNumber=rollNumber;**

**this.name=name;**

**}**

**}**

**class LinkedListTask2{**

**Node start;**

**private class Node{**

**private Student std;**

**private Node next;**

**public Node(Student std){**

**this.std=std;**

**}**

**public Node(Student std,Node next){**

**this.std=std;**

**this.next=next;**

**}**

**}**

**/////////////Method to print all students' records in LinkedList**

**public void printList(){**

**if(start==null)**

**throw new IllegalStateException("Student LinkedList is empty!");**

**Node p=start;**

**System.out.print("[");**

**while(p!=null){**

**System.out.print("("+p.std.rollNumber+","+p.std.name+"),");**

**p=p.next;**

**}**

**System.out.println("\b]");**

**}**

**/////////////Method to insert a student record in LinkedList**

**public Node insert(Student std){**

**if(start==null){**

**start= new Node(std,start);**

**return start;**

**}**

**Node p= start;**

**while (p.next!=null) {**

**p=p.next;**

**}**

**p.next= new Node(std,p.next);**

**return start;**

**}**

**/////////////Method to delete an Student record from LinkedList**

**public Node delete(Student s){**

**if(start==null)**

**return start;**

**if(start.std.rollNumber.equals(s.rollNumber) && start.std.name.equals(s.name)){**

**start= start.next;**

**return start;**

**}**

**Node p=start;**

**while(p.next!=null){**

**if(p.next.std.rollNumber.equals(s.rollNumber) && p.next.std.name.equals(s.name)){**

**p.next=p.next.next;**

**break;**

**}**

**p=p.next;**

**}**

**return start;**

**}**

**public static void main(String[] args) {**

**LinkedListTask2 studentRecords= new LinkedListTask2();**

**Student zohaib=new Student("19sw42","Zohaib");**

**Student amrat=new Student("19sw43","Amrat");**

**Student ahmad=new Student("19sw44","Syed Ahmad Shah");**

**Student uzair=new Student("19sw45","M.Uzair");**

**Student arsam=new Student("19sw46","Arsam");**

**studentRecords.insert(zohaib);**

**studentRecords.insert(amrat);**

**studentRecords.insert(ahmad);**

**studentRecords.insert(uzair);**

**studentRecords.insert(arsam);**

**System.out.println("Students' Records: ");**

**studentRecords.printList();**

**studentRecords.delete(amrat);**

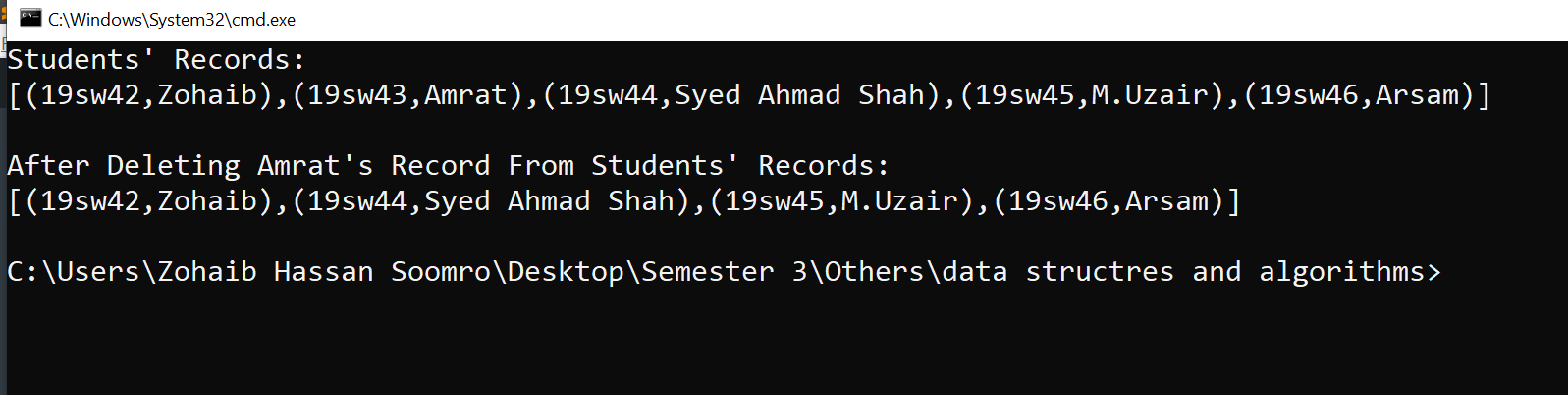
**System.out.println("\nAfter Deleting Amrat's Record From Students' Records: ");**

**studentRecords.printList();**

**}**

**}**

**Task#2 Output:**



**Task 3: Explore java.util.LinkedList class; Create a linked list of type String using this class and apply any 5 of its methods.**

**Code:**

**import java.util.LinkedList;**

**class ApplyMethodsTask3{**

**public static void main(String[] args) {**

**LinkedList<String> countries= new LinkedList<String>();**

**countries.add("Pakistan"); //#1**

**countries.add("India");**

**countries.addFirst("Bangladesh"); //#2**

**countries.addLast("Russia"); //#3**

**countries.add("USA");**

**String allCountries= countries.toString(); //#4**

**System.out.println("LinkedList<String>= "+allCountries);**

**System.out.println("Element at index 2= "+countries.get(2)); //#5**

**}**

**}**

**Task#3 Output:**

