# Rajalakshmi Engineering College

Name: Gokulan V
Email: 240701151@rajalakshmi.edu.in
Roll no: 240701151
Phone: 9361185506
Branch: REC
Department: CSE - Section 9
Batch: 2028
Degree: B.E - CSE

Scan to verify results

## 2024_28_III_OOPS Using Java Lab

### REC_2028_OOPS using Java_Week 9_CY

Attempt : 1
Total Mark : 40
Marks Obtained : 40

### Section 1 : Coding

1. Problem Statement

Sanjay is working on a program to merge two sorted linked lists into a single sorted list using Java's LinkedList class from the Collections framework. Given two sorted linked lists, he wants to merge them while maintaining the sorted order.

Write a Java program that:

Reads two sorted linked lists.Merges them into a single sorted linked list.Prints the merged list in ascending order.

***Input Format***

The first line contains an integer m (the size of the first linked list).

The second line contains m space-separated integers (sorted).

The third line contains an integer n (the size of the second linked list).

The fourth line contains n space-separated integers (sorted).

***Output Format***

The output prints the merged linked list as space-separated integers.

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 2
5 10
3
1 3 8
Output: 1 3 5 8 10

***Answer***

```
import java.util.*;
class MergeSortedLinkedLists {

// You are using Java

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n=sc.nextInt();
        LinkedList <Integer> l1 = new LinkedList <>();
        for(int i=0;i<n;i++){
            l1.add(sc.nextInt());
        }
        int m=sc.nextInt();
        LinkedList <Integer> l2 = new LinkedList <>();
        for(int i=0;i<m;i++){
            l2.add(sc.nextInt());
        }
        LinkedList <Integer> l3 = new LinkedList <>();
        l3.addAll(l1);
        l3.addAll(l2);
        Collections.sort(l3);
```

```
        for(int a : l3){
            System.out.print(a+" ");
        }
    }

}
```

***Status :*** Correct ***Marks : 10/10***

2. Problem Statement

Rahul is working on a list manipulation problem where he needs to reverse a specific subarray using a stack. Given an array and two indices l and r, he wants to reverse only the portion of the array from index l to r (both inclusive) while keeping the rest of the array unchanged.

Since Rahul wants to solve this problem efficiently, he decides to use a stack to reverse the subarray in O(r - l) time.

Your task is to help Rahul by implementing this functionality.

***Input Format***

The first line contains an integer n, the size of the array.

The second line contains n space-separated integers arr[i].

The third line contains two integers l and r, denoting the start and end indices of the subarray to reverse.

Note: The array follows 0-based indexing.

***Output Format***

The output prints the modified array after reversing the subarray between indices l and r.

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 6
1 2 3 4 5 6
1 4

Output: 1 5 4 3 2 6

***Answer***

```
import java.util.*;

public class Main {

// You are using Java

   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int n=sc.nextInt();
      Stack <Integer> l = new Stack<>();
      for(int i=0;i<n;i++){
         l.push(sc.nextInt());
      }
      int a=sc.nextInt();
      int b=sc.nextInt();
      for(int i=0;i<a;i++){
         System.out.print(l.get(i)+" ");
      }
      for(int i=b;i>=a;i--){
         System.out.print(l.get(i)+" ");
      }
      for(int i=b+1;i<n;i++){
         System.out.print(l.get(i)+" ");
      }

   }

}
```

***Status :*** Correct ***Marks : 10/10***

3. Problem Statement

Sarah, a warehouse manager, is managing a list of product names in her

store's inventory system. She needs to perform basic operations like adding (inserting) new products, removing products that are sold out or discontinued, displaying all the products in stock, and searching for a specific product in the inventory list.

Sarah's goal is to manage the inventory using a list of product names (strings). The system allows her to perform the following operations using ArrayList:

Insert a Product: Sarah adds a new product to the inventory.Delete a Product: Sarah removes a product from the inventory when it's sold or discontinued.Display the Inventory: Sarah checks all the products currently available in the inventory.Search for a Product: Sarah searches for a specific product in the inventory to check if it's available.

***Input Format***

The input consists of multiple space-separated values representing different operations on a product list. Each operation follows a specific format:

1 <product_name> - Adds <product_name> to the product list.

2 <product_name> - Removes <product_name> from the product list if it exists.

3 - Print all products currently on the list.

4 <product_name> - Checks if <product_name> exists in the list.

***Output Format***

The output displays,

For (choice 1) prints, " <item> has been added to the list."

For (choice 2) prints, " <item> has been removed from the list."

For (choice 3) prints, "Items in the list:" followed by each item in the list on a new line, or "The list is empty." if the list is empty.

For (choice 4) prints," <item> is found in the list." or " <item> not found in the list."

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 1 apple 1 banana 2 apple 3 4 apple

Output: apple has been added to the list.
banana has been added to the list.
apple has been removed from the list.
Items in the list:
banana
apple not found in the list.

***Answer***

```
import java.util.ArrayList;
import java.util.Scanner;

// You are using Java
class StringListOperations {
    public static void deleteItem(ArrayList arr,String s){
        arr.remove(s);
        System.out.println(s+" has been removed from the list.");
    }
    public static void insertItem(ArrayList arr,String s){
        arr.add(s);
        System.out.println(s+" has been added to the list.");
    }
    public static void displayList(ArrayList arr){
        if(arr.isEmpty()){
            System.out.print("The list is empty.");
        }
        else{
        System.out.println("Items in the list:");
        for(int i=0;i<arr.size();i++){
            System.out.println(arr.get(i));
        }}
    }
    public static void searchItem(ArrayList arr,String s){
        int c=0;
        for(int i=0;i<arr.size();i++){
            if(s.equals(arr.get(i))){
                c=1;
            }
```

```
        }
        if(c==1){
            System.out.println(s+" is found in the list.");
        }
        else{
            System.out.println(s+" not found in the list.");
        }
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        ArrayList<String> list = new ArrayList<>();

        String input = sc.nextLine();
        String[] commands = input.split(" ");
        int i = 0;
        while (i < commands.length) {
            int choice = Integer.parseInt(commands[i]);
            switch (choice) {
                case 1:
                    if (i + 1 < commands.length) {
                        StringListOperations.insertItem(list, commands[i + 1]);
                        i += 2;
                    } else {
                        System.out.println("No string provided for insertion.");
                        i++;
                    }
                    break;
                case 2:
                    if (i + 1 < commands.length) {
                        StringListOperations.deleteItem(list, commands[i + 1]);
                        i += 2;
                    } else {
                        System.out.println("No string provided for deletion.");
                        i++;
                    }
                    break;
                case 3:
                    StringListOperations.displayList(list);
                    i += 1;
                    break;
```

```
            case 4:
                if (i + 1 < commands.length) {
                    StringListOperations.searchItem(list, commands[i + 1]);
                    i += 2;
                } else {
                    System.out.println("No string provided for searching.");
                    i++;
                }
                break;
        }
    }
  }
}
```

*Status :* Correct *Marks : 10/10*

4. Problem Statement

Rahul, a stock trader, wants to analyze the stock prices of a company over several days. For each day, he wants to determine the stock span, which is the number of consecutive days (including the current day) where the stock price is less than or equal to the price on that day.

The stock span helps him understand how long a stock has been continuously increasing or staying the same. You need to help Rahul by computing the stock span for each day using a Stack data structure efficiently.

Example:

Input:

7

100 80 60 70 60 75 85

Output:

1 1 1 2 1 4 6

Explanation:

For each day:

Day 1: Price = 100  Span = 1 (Only this day)Day 2: Price = 80  Span = 1 (Only this day)Day 3: Price = 60  Span = 1 (Only this day)Day 4: Price = 70  Span = 2 (Includes today and previous day)Day 5: Price = 60  Span = 1 (Only this day)Day 6: Price = 75  Span = 4 (Includes today and previous three days)Day 7: Price = 85  Span = 6 (Includes today and previous five days)

***Input Format***

The first line contains an integer n, the number of days.

The second line contains n space-separated integers prices[i], where prices[i] represents the stock price on the i-th day.

***Output Format***

The output prints n space-separated integers representing the stock span for each day.

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 7
100 80 60 70 60 75 85
Output: 1 1 1 2 1 4 6

***Answer***

```
import java.util.*;
class main{
   public static void main(String[] args){
      Scanner sc = new Scanner(System.in);
      int n=sc.nextInt();
      Stack <Integer> st = new Stack <>();
      int[] price = new int [n];
      for(int i=0;i<n;i++){
         price[i]=sc.nextInt();
      }
      int[] span= new int[n];
      for(int i=0;i<n;i++){
         while(!st.isEmpty() && price[st.peek()]<=price[i]){
```

```
            st.pop();
        }
        span[i]=(st.isEmpty())?(i+1):(i-st.peek());
        st.push(i);
    }
    for(int i=0;i<n;i++){
        System.out.print(span[i]+" ");
    }
  }
}
```

***Status :*** Correct **Marks : 10/10**