Rajalakshmi Engineering College

Name: Neha R N
Email: 240701354@rajalakshmi.edu.in
Roll no: 240701354
Phone: 9080137196
Branch: REC
Department: CSE - Section 7
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

A teacher is filtering a list of words provided by students. Some words contain too many vowels, making them difficult for a spelling competition. The teacher decides to remove all words that contain more than two vowels.

Help the teacher to implement it using ArrayList.

Input Format

The first line contains an integer N, representing the number of words in the list.

The next N lines contain a string representing the words (one per line).

Output Format

The output consists of words that contain two or less than two vowels, printed in the same order they appeared in the input. Each word is printed on a new line.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 1
sri

Output: sri

Answer

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

// You are using Java
class VowelFilter {

  public static int countVowels(String word) {
      int count = 0;
      for(char c : word.toCharArray()) {
         if("aeiou". indexOf(c) != -1) {
             count ++;
         }
      }
      return count;
  }
   public static void filterWords(int n , Scanner sc) {
       ArrayList<String> words = new ArrayList<>();
       for (int i = 0; i < n; i++) {
           words.add(sc.nextLine());
       }
         for (String word : words) {
             if (countVowels(word) <= 2) {
                 System.out.println(word);
             }
         }
   }

}
```

```
public class Main {
  public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int n = sc.nextInt();
      sc.nextLine();
      VowelFilter.filterWords(n, sc);
      sc.close();
  }
}
```

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

2. 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) {
        //type your code here
         Scanner sc = new Scanner(System.in);

        int m = sc.nextInt();
        LinkedList<Integer> list1 = new LinkedList<>();
        for (int i = 0; i < m; i++) {
            list1.add(sc.nextInt());
        }

        int n = sc.nextInt();
        LinkedList<Integer> list2 = new LinkedList<>();
        for (int i = 0; i < n; i++) {
            list2.add(sc.nextInt());
        }

        LinkedList<Integer> merged = new LinkedList<>();

        int i = 0, j = 0;
        while (i < list1.size() && j < list2.size()) {
            if (list1.get(i) <= list2.get(j)) {
                merged.add(list1.get(i));
                i++;
            } else {
                merged.add(list2.get(j));
                j++;
            }
        }
        while (i < list1.size()) {
            merged.add(list1.get(i));
            i++;
        }

        while (j < list2.size()) {
            merged.add(list2.get(j));
            j++;
        }

        for (int num : merged) {
            System.out.print(num + " ");
        }

        sc.close();
    }

}
```

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

3. 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

```
// You are using Java
import java.util.Scanner;
import java.util.Stack;

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

```
        int n = sc.nextInt();
        int[] prices = new int[n];
        for(int i = 0; i < n; i++) {
            prices[i] = sc.nextInt();
        }

        Stack<Integer> stack = new Stack<>();
        int[] span = new int[n];

        for (int i = 0; i < n; i++) {

            while (!stack.isEmpty() && prices[stack.peek()] <= prices[i]) {
                stack.pop();
            }

            if (stack.isEmpty()) {

                span[i] = i + 1;
            } else {
                span[i] = i - stack.peek();
            }

            stack.push(i);
        }

        for (int s : span) {
            System.out.print(s + " ");
        }

        sc.close();
    }
}
```

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

4. 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) {
        //type your code here
         Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int[] arr = new int[n];
        for(int i=0; i<n; i++) {
            arr[i] = sc.nextInt();
        }
        int l = sc.nextInt();
        int r = sc.nextInt();

        Stack<Integer> stack = new Stack<>();
        // Push subarray elements onto the stack
        for(int i = l; i <= r; i++) {
            stack.push(arr[i]);
        }

        // Pop from stack to reverse the subarray
        for(int i = l; i <= r; i++) {
            arr[i] = stack.pop();
        }

        // Print the modified array
        for(int i=0; i<n; i++) {
            System.out.print(arr[i]);
            if (i != n-1) System.out.print(" ");
        }

        sc.close();
    }

}
```

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