Rajalakshmi Engineering College

Name: Mohit Jha
Email: 241901056@rajalakshmi.edu.in
Roll no:
Phone: 9445934493
Branch: REC
Department: CSE (CS) - Section 1
Batch: 2028
Degree: B.E - CSE (CS)

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

Raman, a computer science teacher, is responsible for registering students for his programming class. To streamline the registration process, he wants to develop a program that stores students' names and allows him to retrieve a student's name based on their index in the list.

Raman has decided to use an ArrayList to store the names of students, as it provides efficient dynamic resizing and indexing.

Write a program that enables Raman to input the names of students and fetch a student's name using the specified index. If the entered index is invalid, the program should return an appropriate message.

Input Format

The first line of input consists of an integer n, representing the number of students to register.

The next n lines of input consist of the names of each student, one by one.

The last line of input is an integer, representing the index (0-indexed) of the element to retrieve.

Output Format

If the index is valid (within the bounds of the ArrayList), print "Element at index [index]: " followed by the element (student name as string).

If the index is invalid, print "Invalid index".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
Alice
Bob
Ankit
Alice
Prajit
2
Output: Element at index 2: Ankit

Answer

```
import java.util.ArrayList;
import java.util.Scanner;
class NameManager {
   private ArrayList<String> names;

   public NameManager() {
      names = new ArrayList<String>();
   }

   public void addName(String name) {
      names.add(name);
   }
```

```
    public String getNameAtIndex(int index) {
        if (index >= 0 && index < names.size()) {
            return names.get(index);
        } else {
            return null;
        }
    }
}
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        NameManager manager = new NameManager();

        int n = sc.nextInt();
        sc.nextLine(); // consume newline

        for (int i = 0; i < n; i++) {
            String name = sc.nextLine();
            manager.addName(name);
        }

        int index = sc.nextInt();
        String result = manager.getNameAtIndex(index);

        if (result != null) {
            System.out.println("Element at index " + index + ": " + result);
        } else {
            System.out.println("Invalid index");
        }

        sc.close();
    }
}
```

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

2. 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;

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> validWords = new ArrayList<>();
```

```
        for (int i = 0; i < n; i++) {
            String word = sc.nextLine();
            if (countVowels(word) <= 2) {
                validWords.add(word);
            }
        }
        for (String word : validWords) {
            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***

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

```
import java.util.*;

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();
        }
        sc.close();

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

        for (int i = 0; i < n; i++) {
            while (!stack.isEmpty() && prices[stack.peek()] <= prices[i]) {
                stack.pop();
            }
            span[i] = stack.isEmpty() ? (i + 1) : (i - stack.peek());
            stack.push(i);
        }

        for (int i = 0; i < n; i++) {
            System.out.print(span[i] + " ");
        }
    }
}
```

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 {


    public static void main(String[] args) {
        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();
        sc.close();
```

```
        Stack<Integer> stack = new Stack<>();

        for (int i = l; i <= r; i++) {
            stack.push(arr[i]);
        }

        for (int i = l; i <= r; i++) {
            arr[i] = stack.pop();
        }

        for (int i = 0; i < n; i++) {
            System.out.print(arr[i] + " ");
        }
    }
}
```

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