Rajalakshmi Engineering College

Name: Jai Shuriya J
Email: 240701200@rajalakshmi.edu.in
Roll no: 240701200
Phone: 8754381941
Branch: REC
Department: CSE - Section 10
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

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***

2. Problem Statement

Mesa, a store manager, needs a program to manage inventory items. Define a class ItemType with private attributes for name, deposit, and cost per day. Create an ArrayList in the Main class to store ItemType objects, allowing input and display.

Note: Use "%-20s%-20s%-20s" for formatting output in tabular format, display double values with 1 decimal place.

Input Format

The first line of input consists of an integer n, representing the number of items.

For each of the n items, there are three lines:

1. The name of the item (a string)
2. The deposit amount (a double value)
3. The cost per day (a double value)

Output Format

The output prints a formatted table with columns for name, deposit and cost per day.

Refer to the sample output for formatting specifications.

Sample Test Case

```
Input: 3
Laptop
10000.0
250.0
Light
1000.0
50.0
Fan
1000.0
100.0
Output: Name                Deposit             Cost Per Day
Laptop              10000.0             250.0
Light               1000.0              50.0
Fan                 1000.0              100.0
```

Answer

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

class ItemType {
   private String name;
   private Double deposit;
   private Double costPerDay;

   public String toString() {
      return String.format("%-20s%-20s%-20s", name, deposit, costPerDay);
   }

   public ItemType(String name, Double deposit, Double costPerDay) {
      super();
      this.name = name;
      this.deposit = deposit;
      this.costPerDay = costPerDay;
   }
}
class ArrayListObjectMain {
   public static void main(String args[]) {
      List<ItemType> items = new ArrayList<>();
      Scanner sc = new Scanner(System.in);
      int n = Integer.parseInt(sc.nextLine());

      for (int i = 0; i < n; i++) {
         String name = sc.nextLine();
         Double deposit = Double.parseDouble(sc.nextLine());
         Double costPerDay = Double.parseDouble(sc.nextLine());
         items.add(new ItemType(name, deposit, costPerDay));
      }
      System.out.format("%-20s%-20s%-20s", "Name", "Deposit", "Cost Per Day");
      System.out.println();

      for (ItemType item : items) {
         System.out.println(item);
      }
   }
}
```

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

Aarav is developing a music playlist application where users can manage their favorite songs. He wants to implement a feature that allows users to reorder the playlist by moving a song from one position to another.

You need to implement a function that performs the following operations using a LinkedList:

Add songs to the playlist in the given order.Move a song from a specified position to another position in the playlist.Print the final playlist after all operations.

Input Format

The first line of the input consists of an integer n representing the number of songs.

The next n lines, each containing a string representing a song name.

After the songs are given the next line contains an integer m, the number of move operations.

The next m lines, each containing two integers x and y representing the move operation where the song at position x (0-based index) should be moved to position y.

Output Format

The output prints the final playlist, each song on a new line.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
SongA
SongB
SongC
SongD
SongE
2
2 4
0 3

Output: SongB
SongD
SongE
SongA
SongC

Answer

```
import java.util.*;
public class Main {
   public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        sc.nextLine();
        LinkedList<String> playlist = new LinkedList<>();
        for (int i = 0; i < n; i++) playlist.add(sc.nextLine());
        int m = sc.nextInt();
        for (int i = 0; i < m; i++) {
            int x = sc.nextInt();
            int y = sc.nextInt();
            String song = playlist.remove(x);
            playlist.add(y, song);
        }
        for (String song : playlist) System.out.println(song);
    }
}
```

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