Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 9_Q1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Bobby is tasked with processing a sequence of numbers from a monitoring system. He needs to extract a strictly increasing subsequence using an ArrayList. The program should dynamically add numbers to the ArrayList only if they are greater than the last number currently stored in the list. Bobby aims to efficiently utilize the dynamic resizing and indexing features of the ArrayList to solve this problem.

Help Bobby implement this solution.

Input Format

The first line of input consists of an integer N, representing the number of elements.

The second line consists of N space-separated integers, representing the elements.

Output Format

The output prints the list of integers in increasing sequence, ignoring out-of-order elements.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 7
3 5 9 1 11 7 13
Output: [3, 5, 9, 11, 13]

Answer

```
import java.util.*;

class Main

{

   public static void main(String args[])
{

      Scanner sc = new Scanner(System.in);
      int n = sc.nextInt();

      ArrayList<Integer> incList = new ArrayList<>();

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

{

         int num = sc.nextInt();
```

```
        if(incList.isEmpty() || num > incList.get(incList.size() - 1))
{

            incList.add(num);

}

}
        System.out.println(incList);

}
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 9_Q2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Vikram loves listening to music and wants to create a simple playlist manager using Java Collections. The playlist supports the following operations:

"ADD <song>" Adds the song to the end of the playlist."REMOVE <song>" Removes the first occurrence of the song from the playlist. If the song is not found, do nothing."SHOW" Displays all songs in the playlist in order. If the playlist is empty, print "EMPTY"."NEXT" Moves to the next song in the playlist and prints its name. If the playlist is empty, print "EMPTY".

The playlist maintains a "current song" position that starts at the first song when it's added. The NEXT command moves to the next song and prints it, wrapping around to the first song after reaching the last song. When removing songs, the current position adjusts accordingly to maintain

proper navigation.

Help Vikram implement this playlist manager.

Input Format

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

The next n lines, each containing a command:

- "ADD <song>"
- "REMOVE <song>"
- "SHOW"
- "NEXT"

Output Format

For each "SHOW" command, print the songs in order, separated by spaces.

For each "NEXT" command, print the next song in the playlist.

If no song exists, print "EMPTY".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 7
ADD song1
ADD song2
SHOW
NEXT
REMOVE song2
SHOW
NEXT

Output: song1 song2
song2
song1
song1

Answer

```
import java.util.Scanner;
import java.util.LinkedList;

class Main

{


   public static void main(String args[])

{


      Scanner sc = new Scanner(System.in);
      int n = Integer.parseInt(sc.nextLine());

      LinkedList<String> playList = new LinkedList<>();
      int currentIndex = -1;

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

{


         String input = sc.nextLine();
         if(input.startsWith("ADD "))

{


            String song = input.substring(4);
            playList.add(song);
            if(currentIndex == -1)

{


               currentIndex = 0;


}
```

```
}
        else if(input.startsWith("REMOVE "))

{


            String song = input.substring(7);
            int index = playList.indexOf(song);
            if(index != -1)

{


                playList.remove(index);
                if(playList.isEmpty()) currentIndex = -1;


}else if(index < currentIndex)

{


                currentIndex--;


}else if(index == currentIndex)

{


                currentIndex = currentIndex % playList.size();


}


}else if(input.equals("SHOW"))

{
```

```
            if(playList.isEmpty())
    {

                System.out.println("EMPTY");

    }else

    {

                for(String song : playList)
    {

                    System.out.print(song + " ");

    }
                System.out.println();

    }
    }else if(input.equals("NEXT"))
    {

                if(playList.isEmpty())
    {

                System.out.println("EMPTY");
```

```
}else
{

        currentIndex = (currentIndex + 1) % playList.size();
        System.out.println(playList.get(currentIndex));


}

}

}

}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 9_Q3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Assist Pranitha in developing a program that takes an integer N as input, representing the number of names to be read. Then read N names and store them in an ArrayList. Finally, input a search string and output the frequency of that string in the list of names.

Note: Some parts of the code are provided as snippets, and you need to complete the remaining sections by writing the necessary code.

Input Format

The first line of input consists of an integer N, representing the number of names to be read.

The following N lines consist of N names, as a string.

The last line consists of a string, representing the name to be searched.

Output Format

The output prints a single integer, representing the frequency of the specified name in the given list.

If the specified name is not found, print 0.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
Alice
Bob
Ankit
Alice
Pranitha
Alice
Output: 2

Answer

```
import java.util.*;

class Main

{


   public static void main(String[] args)

{


      Scanner sc = new Scanner(System.in);
      int n = Integer.parseInt(sc.nextLine());

      ArrayList<String> names = new ArrayList<>();
```

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

            String name = sc.nextLine();
            names.add(name);


}

        String searchName = sc.nextLine();

        int count = 0;

        for(String name : names)

{


            if(name.equals(searchName))

{


                count++;


}


}
        System.out.println(count);


}

}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

REC_2028_OOPS using Java_Week 9_PAH

Attempt : 1
Total Mark : 30
Marks Obtained : 30

Section 1 : Coding

1. Problem Statement

Arun is building a task manager to keep track of tasks using a LinkedList. The task manager supports the following operations:

"ADD <task>" Adds the given task to the end of the list."REMOVE" Removes the first task from the list."SHOW" Displays all tasks in the list in order. If the list is empty, print "EMPTY".

Help Arun implement this functionality using a LinkedList.

Input Format

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

The next n lines, each containing a command:

- "ADD <task>"
- "REMOVE"
- "SHOW"

Output Format

For each "SHOW" command, the output prints the tasks in order, separated by spaces.

If no tasks exist, print "EMPTY".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
ADD homework
ADD project
SHOW
REMOVE
SHOW
Output: homework project
project

Answer

```
import java.util.*;

public class Main

{


   public static void main(String[] args)

{


      Scanner sc = new Scanner(System.in);
      int n = Integer.parseInt(sc.nextLine());

      LinkedList<String> tasks = new LinkedList<>();
```

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

            String command = sc.nextLine();

            if (command.startsWith("ADD "))

{

                String task = command.substring(4);
                tasks.add(task);

} else if (command.equals("REMOVE"))

{

                if (!tasks.isEmpty())

{

                    tasks.removeFirst();

}

} else if (command.equals("SHOW"))

{

                if (tasks.isEmpty())

{
```

```
            System.out.println("EMPTY");
        } else
        {
            for (String task : tasks)
        {
              System.out.print(task + " ");
        }
            System.out.println();
        }
    }
}
```

Status : Correct *Marks : 10/10*

2. Problem Statement

Rekha is a teacher who wants to calculate the average of marks scored by her students in a test. She needs to store all the marks dynamically

because the number of students may vary each time. Using an ArrayList allows her to easily add any number of marks without worrying about the initial size.

Help her implement the task.

Input Format

The first line of input is an integer n, representing the number of students..

The second line of input consists of n double values, representing the marks of each student, separated by a space.

Output Format

The output prints: "Average of the list: " followed by the average value formatted to two decimal places.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
1.0 2.0 3.0 4.0 5.0
Output: Average of the list: 3.00

Answer

```
import java.util.*;

public class Main

{


   public static void main(String[] args)

{


      Scanner sc = new Scanner(System.in);
      int n = Integer.parseInt(sc.nextLine().trim());
```

```
        String[] input = sc.nextLine().trim().split(" ");
        ArrayList<Double> marks = new ArrayList<>();
        for (int i = 0; i < n; i++)

{


            marks.add(Double.parseDouble(input[i]));


}
        double sum = 0.0;
        for (double mark : marks)

{


            sum += mark;


}
        System.out.printf("Average of the list: %.2f\n", sum / n);


}

}
```

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

3. Problem Statement

Aditi is analyzing stock market trends and wants to find the Next Greater Element (NGE) for each stock price in a list. The Next Greater Element for an element x in an array is the first element to the right that is greater than x. If no greater element exists, return -1 for that position.

Your task is to help Aditi by efficiently computing the Next Greater Element for each element in the given array using a Stack.

Example:

Input:

6

4 5 2 10 8 6

Output:

5 10 10 -1 -1 -1

Explanation:

For each element:

4 5 (next greater element)5 10 2 10 10 -1 (No greater element)8 -1 6

Input Format

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

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

Output Format

The output prints n space-separated integers representing the Next Greater Element for each element in the array.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 6
4 5 2 10 8 6
Output: 5 10 10 -1 -1 -1

Answer

```
// You are using Java
import java.util.*;

public class Main
{
```

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

        Scanner sc = new Scanner(System.in);
        int n = Integer.parseInt(sc.nextLine().trim());
        String[] input = sc.nextLine().trim().split(" ");
        int[] arr = new int[n];
        int[] nge = new int[n];
        Stack<Integer> stack = new Stack<>();

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

            arr[i] = Integer.parseInt(input[i]);

}

        for (int i = n - 1; i >= 0; i--)
{

            while (!stack.isEmpty() && stack.peek() <= arr[i])
{

                stack.pop();

}
            nge[i] = stack.isEmpty() ? -1 : stack.peek();
            stack.push(arr[i]);
```

```
}
        for (int val : nge)
{

            System.out.print(val + " ");

}

}
}
```

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

Rajalakshmi Engineering College

Name: Dharshini A
Email: 241001049@rajalakshmi.edu.in
Roll no: 241001049
Phone: 8056618801
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

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 {

public static void main(String[] args){


    Scanner sc = new Scanner(System.in);
    int m = Integer.parseInt(sc.nextLine().trim());
    LinkedList<Integer> list1 = new LinkedList<>();
    if (m > 0)

{


        String[] arr1 = sc.nextLine().trim().split(" ");
        for (int i = 0; i < m; i++)

{
```

```
            list1.add(Integer.parseInt(arr1[i]));

    }

    }
        int n = Integer.parseInt(sc.nextLine().trim());
        LinkedList<Integer> list2 = new LinkedList<>();
        if (n > 0)

    {

            String[] arr2 = sc.nextLine().trim().split(" ");
            for (int i = 0; i < n; i++)

    {

                list2.add(Integer.parseInt(arr2[i]));

    }

    }
        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 val : merged)
{
```

```
        System.out.print(val + " ");

    }


    }
    }
```

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

2. 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.*;

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++)
```

```
{

        String song = sc.nextLine();
        playList.add(song);


}

    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**

3. 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();

      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.println(arr[i] + " ");

        }

    }
}
```

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.*;

public class Main
{
```

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

        Scanner sc = new Scanner(System.in);
        int n = Integer.parseInt(sc.nextLine().trim());
        String[] input = sc.nextLine().trim().split(" ");
        int[] prices = new int[n];
        int[] span = new int[n];
        Stack<Integer> stack = new Stack<>();

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

            prices[i] = Integer.parseInt(input[i]);


}

        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 s : span)
    {

            System.out.print(s + " ");

    }

    }
    }
```

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