

# Rajalakshmi Engineering College

Name: UMAR FAROOK

Email: 240701573@rajalakshmi.edu.in

Roll no: 240701573

Phone: 9791730398

Branch: REC

Department: CSE - Section 5

Batch: 2028

Degree: B.E - CSE

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**

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 -16 -1

### ***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***

```
import java.util.*;  
  
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[] result = new int[n];  
        Stack<Integer> stack = new Stack<>();
```

```
for (int i = n - 1; i >= 0; i--) {
    while (!stack.isEmpty() && stack.peek() <= arr[i]) {
        stack.pop();
    }
    result[i] = stack.isEmpty() ? -1 : stack.peek();
    stack.push(arr[i]);
}
for (int i = 0; i < n; i++) {
    System.out.print(result[i] + " ");
}
```

Status : Correct

Marks : 10/10

## 2. 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.*;  
  
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 input = sc.nextLine();  
            if (input.startsWith("ADD")) {  
                String task = input.substring(4);  
                tasks.add(task);  
            } else if (input.equals("REMOVE")) {  
                if (!tasks.isEmpty()) {  
                    tasks.removeFirst();  
                }  
            } else if (input.equals("SHOW")) {  
                if (tasks.isEmpty()) {  
                    System.out.println("EMPTY");  
                } else {  
                    for (int j = 0; j < tasks.size(); j++) {  
                        System.out.print(tasks.get(j));  
                        if (j < tasks.size() - 1) System.out.print(" ");  
                    }  
                }  
            }  
        }  
    }  
}
```

```
        System.out.println();  
    }  
}  
}  
}
```

**Status : Correct**

**Marks : 10/10**

### 3. 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.*;  
  
class Main {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        int n = sc.nextInt();  
        ArrayList<Double> marks = new ArrayList<>();  
        for (int i = 0; i < n; i++) {  
            marks.add(sc.nextDouble());  
        }  
        double sum = 0;  
        for (int i = 0; i < marks.size(); i++) {  
            sum += marks.get(i);  
        }  
        double avg = sum / n;  
        System.out.printf("Average of the list: %.2f", avg);  
    }  
}
```

**Status : Correct**

**Marks : 10/10**