

# Rajalakshmi Engineering College

Name: Dharshana shri S  
Email: 240701119@rajalakshmi.edu.in  
Roll no: 240701119  
Phone: 9361711541  
Branch: REC  
Department: CSE - Section 9  
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)5 102 1010 -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***

```
// You are using Java
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];
        int[] nge = new int[n];

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

```

    }
    Stack<Integer> st = new Stack<>();

    for (int i = n - 1; i >= 0; i--) {
        while (!st.isEmpty() && st.peek() <= arr[i]) {
            st.pop();
        }
        if (!st.isEmpty()) {
            nge[i] = st.peek();
        } else {
            nge[i] = -1;
        }
        st.push(arr[i]);
    }

    for (int i = 0; i < n; i++) {
        System.out.print(nge[i]);
        if (i < n - 1) System.out.print(" ");
    }
}
}

```

**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.*;
```

```
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); // Add task to the end of the list  
            } else if (command.equals("REMOVE")) {  
                if (!tasks.isEmpty()) {
```

```
tasks.removeFirst(); // Remove first task from the list
}
} else if (command.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();
    }
}
```

### 3. Problem Statement

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**

// You are using Java

import java.util.\*;

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

**Status : Correct**

**Marks : 10/10**

# Rajalakshmi Engineering College

Name: Dharshana shri S  
Email: 240701119@rajalakshmi.edu.in  
Roll no: 240701119  
Phone: 9361711541  
Branch: REC  
Department: CSE - Section 9  
Batch: 2028  
Degree: B.E - CSE

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.ArrayList;
```

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

```
public class Main {
```

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

```
        Scanner sc = new Scanner(System.in);
```

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

```
        sc.nextLine();
```

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

```
        for (int i = 0; i < N; i++)
```

```
        {
```

```
            String name = sc.nextLine();
```

```
            whitelist.add(name);
```

```
        }
```

```
        String searchName = sc.nextLine();
```

```
        int frequency = 0;
```

```
        for (String name : whitelist) {
```

```
            if (name.equals(searchName))
```



```
    {  
        frequency++;  
    }  
    System.out.println(frequency);  
}  
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Dharshana shri S  
Email: 240701119@rajalakshmi.edu.in  
Roll no: 240701119  
Phone: 9361711541  
Branch: REC  
Department: CSE - Section 9  
Batch: 2028  
Degree: B.E - CSE

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.LinkedList;
import java.util.Scanner;

class PlaylistManager {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        sc.nextLine();

        LinkedList<String> playlist = new LinkedList<>();
        int currentIndex = -1;
        for (int i = 0; i < n; i++) {
            String command = sc.nextLine().trim();

            if (command.startsWith("ADD")) {
                String song = command.substring(4);
                playlist.add(song);
                if (currentIndex == -1) currentIndex = 0;
            }
            else if (command.startsWith("REMOVE")) {
                String song = command.substring(7);

                int index = 0;
                boolean found = false;
                for (String s : playlist) {
                    if (s.equals(song)) {
                        found = true;
                        break;
                    }
                    index++;
                }

                if (found) {
                    playlist.remove(song);
                    if (playlist.isEmpty()) {
                        currentIndex = -1;
                    }
                    else if (index < currentIndex) {
                        currentIndex--;
                    }
                    else if (index == currentIndex) {
                        if (currentIndex >= playlist.size()) currentIndex = 0;
                    }
                }
            }
        }
    }
}
```

```

    }
    else if (command.equals("SHOW")) {
        if (playlist.isEmpty()) {
            System.out.println("EMPTY");
        } else {
            for (int j = 0; j < playlist.size(); j++) {
                System.out.print(playlist.get(j));
                if (j != playlist.size() - 1) System.out.print(" ");
            }
            System.out.println();
        }
    }
    else if (command.equals("NEXT")) {
        if (playlist.isEmpty()) {
            System.out.println("EMPTY");
        } else {
            currentIndex = (currentIndex + 1) % playlist.size();
            System.out.println(playlist.get(currentIndex));
        }
    }
}
sc.close();
}
}

```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: Dharshana shri S  
Email: 240701119@rajalakshmi.edu.in  
Roll no: 240701119  
Phone: 9361711541  
Branch: REC  
Department: CSE - Section 9  
Batch: 2028  
Degree: B.E - CSE

Scan to verify results



## 2024\_28\_III\_OOPS Using Java Lab

### 2028\_REC\_OOPS using Java\_Week 9\_Q1

Attempt : 2  
Total Mark : 10  
Marks Obtained : 0

#### 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**

-

**Status :** Skipped

**Marks :** 0/10