

# Rajalakshmi Engineering College

Name: Zaina Raheen A.P

Email: 241801328@rajalakshmi.edu.in

Roll no: 241801328

Phone: 9384899474

Branch: REC

Department: AI & DS - Section 4

Batch: 2028

Degree: B.E - AI & DS

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***

```
// You are using Java
import java.util.*;
class IncreasingSubsequence {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int N = sc.nextInt();
        ArrayList<Integer> result = new ArrayList<>();
        for (int i = 0; i < N; i++) {
            int num = sc.nextInt();
            if (result.isEmpty() || num > result.get(result.size() - 1)) {
                result.add(num);
            }
        }
        System.out.println(result);
    }
}
```

**Status : Correct**

**Marks : 10/10**

# Rajalakshmi Engineering College

Name: Zaina Raheen A.P

Email: 241801328@rajalakshmi.edu.in

Roll no: 241801328

Phone: 9384899474

Branch: REC

Department: AI & DS - Section 4

Batch: 2028

Degree: B.E - AI & DS

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**

```
// You are using Java
import java.util.*;
class PlaylistManager {
    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 = 0;
        for (int i = 0; i < n; i++) {
            String commandLine = sc.nextLine();
            String[] parts = commandLine.split(" ", 2);
            String command = parts[0];
            if (command.equals("ADD")) {
                playlist.add(parts[1]);
            } else if (command.equals("REMOVE")) {
                String song = parts[1];
                int index = playlist.indexOf(song);
                if (index != -1) {
                    playlist.remove(index);
                    if (index < currentIndex || currentIndex >= playlist.size()) {
                        currentIndex = Math.max(0, currentIndex - 1);
                    }
                }
            } else if (command.equals("SHOW")) {
                if (playlist.isEmpty()) {
                    System.out.println("EMPTY");
                } else {
                    for (String song : playlist) {
                        System.out.print(song + " ");
                    }
                    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));
                }
            }
        }
    }
}
```

}

**Status : Correct**

**Marks : 10/10**