

# Rajalakshmi Engineering College

Name: MRIDHULA DEVI M  
Email: 240701337@rajalakshmi.edu.in  
Roll no:  
Phone: 9840329629  
Branch: REC  
Department: CSE - Section 8  
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

Raman, a computer science teacher, is responsible for registering students for his programming class. To streamline the registration process, he wants to develop a program that stores students' names and allows him to retrieve a student's name based on their index in the list.

Raman has decided to use an ArrayList to store the names of students, as it provides efficient dynamic resizing and indexing.

Write a program that enables Raman to input the names of students and fetch a student's name using the specified index. If the entered index is invalid, the program should return an appropriate message.

##### *Input Format*

The first line of input consists of an integer n, representing the number of students to register.

The next n lines of input consist of the names of each student, one by one.

The last line of input is an integer, representing the index (0-indexed) of the element to retrieve.

### ***Output Format***

If the index is valid (within the bounds of the ArrayList), print "Element at index [index]: " followed by the element (student name as string).

If the index is invalid, print "Invalid index".

Refer to the sample output for formatting specifications.

### ***Sample Test Case***

Input: 5

Alice

Bob

Ankit

Alice

Prajit

2

Output: Element at index 2: Ankit

### ***Answer***

// You are using Java

import java.util.\*;

class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

if (!sc.hasNextInt()) return; // no input

int n = sc.nextInt();

ArrayList<String> students = new ArrayList<>();

for (int i = 0; i < n && sc.hasNext(); i++) {

students.add(sc.next());

```

    }

    int index = -1;
    if (sc.hasNextInt()) {
        index = sc.nextInt();
    }

    if (index >= 0 && index < students.size()) {
        System.out.print("Element at index " + index + ": " + students.get(index));
    } else {
        System.out.print("Invalid index");
    }
}
}
}

```

**Status :** Correct

**Marks :** 10/10

## 2. Problem Statement

Sarah, a warehouse manager, is managing a list of product names in her store's inventory system. She needs to perform basic operations like adding (inserting) new products, removing products that are sold out or discontinued, displaying all the products in stock, and searching for a specific product in the inventory list.

Sarah's goal is to manage the inventory using a list of product names (strings). The system allows her to perform the following operations using ArrayList:

Insert a Product: Sarah adds a new product to the inventory. Delete a Product: Sarah removes a product from the inventory when it's sold or discontinued. Display the Inventory: Sarah checks all the products currently available in the inventory. Search for a Product: Sarah searches for a specific product in the inventory to check if it's available.

### ***Input Format***

The input consists of multiple space-separated values representing different operations on a product list. Each operation follows a specific format:

1 <product\_name> - Adds <product\_name> to the product list.

2 <product\_name> - Removes <product\_name> from the product list if it exists.

3 - Print all products currently on the list.

4 <product\_name> - Checks if <product\_name> exists in the list.

### ***Output Format***

The output displays,

For (choice 1) prints, " <item> has been added to the list."

For (choice 2) prints, " <item> has been removed from the list."

For (choice 3) prints, "Items in the list:" followed by each item in the list on a new line, or "The list is empty." if the list is empty.

For (choice 4) prints, " <item> is found in the list." or " <item> not found in the list."

Refer to the sample output for formatting specifications.

### ***Sample Test Case***

Input: 1 apple 1 banana 2 apple 3 4 apple

Output: apple has been added to the list.

banana has been added to the list.

apple has been removed from the list.

Items in the list:

banana

apple not found in the list.

### ***Answer***

```
import java.util.ArrayList;
```

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

```
class StringListOperations {
```

```
    public static void insertItem(ArrayList<String> list, String item) {  
        list.add(item);
```

```

        System.out.println(item + " has been added to the list.");
    }

    public static void deleteItem(ArrayList<String> list, String item) {
        if (list.contains(item)) {
            list.remove(item);
            System.out.println(item + " has been removed from the list.");
        } else {

        }
    }

    public static void displayList(ArrayList<String> list) {
        if (list.isEmpty()) {
            System.out.println("The list is empty.");
        } else {
            System.out.println("Items in the list:");
            for (String item : list) {
                System.out.println(item);
            }
        }
    }

    public static void searchItem(ArrayList<String> list, String item) {
        if (list.contains(item)) {
            System.out.println(item + " is found in the list.");
        } else {
            System.out.println(item + " not found in the list.");
        }
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        ArrayList<String> list = new ArrayList<>();

        String input = sc.nextLine();
        String[] commands = input.split(" ");
        int i = 0;
        while (i < commands.length) {
            int choice = Integer.parseInt(commands[i]);
            switch (choice) {

```

```

case 1:
    if (i + 1 < commands.length) {
        StringListOperations.insertItem(list, commands[i + 1]);
        i += 2;
    } else {
        System.out.println("No string provided for insertion.");
        i++;
    }
    break;
case 2:
    if (i + 1 < commands.length) {
        StringListOperations.deleteItem(list, commands[i + 1]);
        i += 2;
    } else {
        System.out.println("No string provided for deletion.");
        i++;
    }
    break;
case 3:
    StringListOperations.displayList(list);
    i += 1;
    break;
case 4:
    if (i + 1 < commands.length) {
        StringListOperations.searchItem(list, commands[i + 1]);
        i += 2;
    } else {
        System.out.println("No string provided for searching.");
        i++;
    }
    break;
}
}
}
}
}

```

**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**

// You are using Java

import java.util.\*;

```
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();
        }
        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();
            }
            if (stack.isEmpty()) span[i] = i + 1;
            else span[i] = i - stack.peek();
            stack.push(i);
        }
        for (int i = 0; i < n; i++) {
            System.out.print(span[i]);
            if (i != n - 1) System.out.print(" ");
        }
    }
}
```

**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**

// You are using Java

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++) {
            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