

# Rajalakshmi Engineering College

Name: Nakshatra Pa  
Email: 241901062@rajalakshmi.edu.in  
Roll no: 241901062  
Phone: 8838047354  
Branch: REC  
Department: CSE (CS) - Section 1  
Batch: 2028  
Degree: B.E - CSE (CS)

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 : 20

#### 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 {
```

```
// You are using Java
```

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

```
        //type your code here
```

```
        import java.util.*;
```

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

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

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

```
        LinkedList<Integer> list1 = new LinkedList<>();
```

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

```
            list1.add(sc.nextInt());
```

```
        }
```

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

```
        LinkedList<Integer> list2 = new LinkedList<>();
```

```

    for (int i = 0; i < n; i++) {
        list2.add(sc.nextInt());
    }

    LinkedList<Integer> mergedList = new LinkedList<>();
    int i = 0, j = 0;

    // Merge while maintaining sorted order
    while (i < list1.size() && j < list2.size()) {
        if (list1.get(i) <= list2.get(j)) {
            mergedList.add(list1.get(i));
            i++;
        } else {
            mergedList.add(list2.get(j));
            j++;
        }
    }

    // Add remaining elements from list1
    while (i < list1.size()) {
        mergedList.add(list1.get(i));
        i++;
    }

    // Add remaining elements from list2
    while (j < list2.size()) {
        mergedList.add(list2.get(j));
        j++;
    }

    // Print merged list
    for (int num : mergedList) {
        System.out.print(num + " ");
    }
}

}

}

```

**Status : Wrong**

**Marks : 0/10**

## 2. 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.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(); // Consume newline after integer input

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

// Read student names

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

students.add(sc.nextLine());

}

int index = sc.nextInt();

// Check if index is valid

if (index >= 0 && index < students.size()) {

System.out.println("Element at index " + index + ": " + students.get(index));

} else {

System.out.println("Invalid index");

}

}

}

**Status : Correct**

**Marks : 10/10**

### 3. 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.*;
```

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

        int n = sc.nextInt();
        sc.nextLine(); // Consume newline

        LinkedList<String> playlist = new LinkedList<>();

        // Add songs to the playlist
        for (int i = 0; i < n; i++) {
            playlist.add(sc.nextLine());
        }

        int m = sc.nextInt();

        // Perform move operations
        for (int i = 0; i < m; i++) {
            int x = sc.nextInt();
            int y = sc.nextInt();

            String song = playlist.remove(x); // Remove song at index x
            playlist.add(y, song);           // Add it at index y
        }

        // Print final playlist
        for (String song : playlist) {
            System.out.println(song);
        }
    }
}
```

**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**

// You are using Java

import java.util.Scanner;

import java.util.Stack;

```
public class StockSpan {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        int n = sc.nextInt();
        int[] prices = new int[n];
        int[] span = new int[n];

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

        Stack<Integer> stack = new Stack<>();

        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 i = 0; i < n; i++) {
            System.out.print(span[i] + " ");
        }
    }
}
```

241901062  
}  
}  
}

Status : Wrong

Marks : 0/10

241901062

241901062

241901062

241901062

241901062

241901062

241901062

241901062

241901062

241901062

241901062

241901062

241901062

241901062

241901062