

# Rajalakshmi Engineering College

Name: dhanesh waran

Email: 241901024@rajalakshmi.edu.in

Roll no: 241901024

Phone: 9363879477

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

### 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.*;
public class Main{
    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()){
                result.add(num);
            }
            else if(num>result.get(result.size()-1)){
                result.add(num);
            }
        }
        System.out.println(result);
    }
}
```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: dhanesh waran

Email: 241901024@rajalakshmi.edu.in

Roll no: 241901024

Phone: 9363879477

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\_MCQ

Attempt : 1

Total Mark : 15

Marks Obtained : 15

#### Section 1 : MCQ

1. What will be the output of the following code?

```
import java.util.ArrayList;

public class Main {
    public static void main(String[] args) {
        ArrayList<String> list = new ArrayList<>();
        list.add("Apple");
        list.add("Banana");
        list.remove("Apple");
        System.out.println(list);
    }
}
```

**Answer**

[Banana]

**Status :** Correct

**Marks :** 1/1

2. What will be the output of the following code?

```
import java.util.*;
class Main {
    public static void main(String[] args) {
        ArrayList<String> list = new ArrayList<>();
        list.add("apple");
        list.add("banana");
        list.add("cherry");
        list.add("banana");
        System.out.println(list.lastIndexOf("banana"));
    }
}
```

**Answer**

3

**Status :** Correct

**Marks :** 1/1

3. What is Collection in Java?

**Answer**

A group of objects

**Status :** Correct

**Marks :** 1/1

4. Which method is used to add an element to the top of the stack?

**Answer**

push()

**Status :** Correct

**Marks :** 1/1

5. How can you access the first element of an ArrayList named as list?

**Answer**

```
list.get(0);
```

**Status :** Correct

**Marks :** 1/1

6. What will be the output of the following code?

```
import java.util.*;
class Main {
    public static void main(String[] args) {
        ArrayList<Integer> list = new ArrayList<>();
        list.add(1);
        list.add(2);
        list.add(3);
        list.add(4);
        list.set(2, 10);
        System.out.println(list);
    }
}
```

**Answer**

[1, 2, 10, 4]

**Status :** Correct

**Marks :** 1/1

7. What will be the output of the following code?

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Stack<Integer> s = new Stack<>();
        s.push(10);
        s.push(20);
        s.push(30);
        System.out.println(s.peek());
    }
}
```

**Answer**

30

**Status :** Correct

**Marks :** 1/1

8. What is the correct way to create an ArrayList in Java?

**Answer**

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

**Status :** Correct

**Marks :** 1/1

9. What will be the output of the following code?

```
import java.util.*;
class Main {
    public static void main(String[] args) {
        ArrayList<Integer> list = new ArrayList<>();
        list.add(1);
        list.add(2);
        list.add(3);
        list.add(4);
        list.add(5);
        System.out.println(list.get(3));
    }
}
```

**Answer**

4

**Status :** Correct

**Marks :** 1/1

10. What will be the output of the following code?

```
import java.util.*;
class Main {
    public static void main(String[] args) {
        ArrayList<String> list = new ArrayList<>();
```

```
list.add("Java");
list.add("Python");
list.add("Java");
list.add("C++");
System.out.println(list.indexOf("Java"));
}
```

**Answer**

0

**Status :** Correct

**Marks :** 1/1

11. What will be the output of the following code?

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Stack<Integer> stack = new Stack<>();
        for (int i = 1; i <= 3; i++)
            stack.push(i * 2);
        stack.pop();
        stack.push(10);
        System.out.println(stack.peek());
    }
}
```

**Answer**

10

**Status :** Correct

**Marks :** 1/1

12. What does the addFirst() method of LinkedList do?

**Answer**

Adds an element to the beginning of the list

**Status :** Correct

**Marks :** 1/1

13. Which of the following methods removes and returns the last element from a LinkedList?

**Answer**

removeLast()

**Status :** Correct

**Marks :** 1/1

14. What will be the output of the following code?

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

```
public class Main {  
    public static void main(String[] args) {  
        ArrayList<Integer> list = new ArrayList<>();  
        list.add(10);  
        list.add(20);  
        list.add(30);  
        System.out.println("Size of the list: " + list.size());  
    }  
}
```

**Answer**

Size of the list: 3

**Status :** Correct

**Marks :** 1/1

15. What will be the output of the following code?

```
import java.util.*;  
class Main {  
    public static void main(String[] args) {  
        ArrayList<Integer> list = new ArrayList<>();  
        list.add(10);  
        list.add(20);  
        list.add(30);  
        list.remove(1);  
        System.out.println(list);  
    }  
}
```

}  
}  
**Answer**

[10, 30]

**Status :** Correct

**Marks :** 1/1

# Rajalakshmi Engineering College

Name: dhanesh waran

Email: 241901024@rajalakshmi.edu.in

Roll no: 241901024

Phone: 9363879477

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

### 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.*;
public class Main{
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        int a=sc.nextInt();
        sc.nextLine();
        LinkedList<String> playList=new LinkedList<>();
        int current=-1;
        for(int i=0;i<a;i++){
            String command=sc.nextLine();
            String[] parts=command.split(" ",2);
            switch (parts[0]){
                case "ADD":
                    String songtoadd=parts[1];
                    playList.add(songtoadd);
                    if(playList.size()==1){
                        current=0;
                    }
                    break;
                case "REMOVE":
                    String songtoremove=parts.length>1? parts[1] : "";
                    int index=playList.indexOf(songtoremove);
                    if(index!=-1){
                        playList.remove(index);
                        if(playList.isEmpty()){
                            current=-1;
                        }
                        else if(index<current){
                            current--;
                        }
                        else if(index==current){
                            if(current>=playList.size()){
                                current=0;
                            }
                        }
                    }
                    break;
                case "SHOW":
                    if(playList.isEmpty()){
                        System.out.print("EMPTY");
                    }
            }
        }
    }
}
```

```

    }
    else{
        for(String s:playList){
            System.out.print(s+" ");
        }
    }
    break;
    case "NEXT":
        if(playList.isEmpty()){
            System.out.print("EMPTY");
        }
        else{
            current=(current+1)%playList.size();
            System.out.print(playList.get(current)+" ");
        }
        break;
    }
}
}
}
}

```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: dhanesh waran

Email: 241901024@rajalakshmi.edu.in

Roll no: 241901024

Phone: 9363879477

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

### 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**

// You are using Java

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

```
public class Main{
```

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

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

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

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

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

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

```
            names.add(sc.nextLine());
```

```
        }
```

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

```
        int count=0;
```

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

```
            if(name.equals(search)){
```

```
                count++;
```

```
            }
```

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

**Status :** Correct

**Marks : 10/10**

# Rajalakshmi Engineering College

Name: dhanesh waran

Email: 241901024@rajalakshmi.edu.in

Roll no: 241901024

Phone: 9363879477

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\_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***

```
import java.util.*;
public class Main{
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        int a=sc.nextInt();
        int arr[]=new int[a];
        for(int i=0;i<a;i++){
            arr[i]=sc.nextInt();
        }
        int result[]=new int[a];
        Stack<Integer>st =new Stack<>();
        for(int i=0;i<a;i++){
```

```

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

```

**Status :** Correct

**Marks :** 10/10

## 2. 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

// You are using Java

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

```
public class Main{
```

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

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

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

```
        ArrayList<Double> mark=new ArrayList<>();
```

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

```
            mark.add(sc.nextDouble());
```

```
        }
```

```
        double sum=0.0;
```

```
        for(int i=0;i<mark.size();i++){
```

```
            sum+=mark.get(i);
```

```
        }
```

```
        double d=sum/a;
```

```
        System.out.printf("Average of the list: %.2f",d);
```

```
    }
```

```
}
```

Status : Correct

Marks : 10/10

### 3. 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**

```
// You are using Java
import java.util.*;
public class Main{
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        int a=sc.nextInt();
        sc.nextLine();
        LinkedList<String> tasks=new LinkedList<>();
        for(int i=0;i<a;i++){
            String command=sc.nextLine();
            if(command.startsWith("ADD")){
                String task=command.substring(4);
                tasks.add(task);
            }
        }
    }
}
```

```

    }
    else if(command.equals("REMOVE")){
        if(!tasks.isEmpty()){
            tasks.removeFirst();
        }
    }
    else if(command.equals("SHOW")){
        if(tasks.isEmpty()){
            System.out.println("EMPTY");
        }
        else{
            for(String t:tasks){
                System.out.print(t+" ");
            }
            System.out.println();
        }
    }
}

```

**Status :** Correct

**Marks :** 10/10

# Rajalakshmi Engineering College

Name: dhanesh waran

Email: 241901024@rajalakshmi.edu.in

Roll no: 241901024

Phone: 9363879477

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 : 40

### Section 1 : Coding

#### 1. 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();

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

for (int i = 0; i < n; i++) {
    String song = sc.nextLine();
    playlist.add(song);
}

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 s : playlist) {
    System.out.println(s);
}
}

```

**Status :** Correct

**Marks :** 10/10

## 2. 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.*;

public 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> st = new Stack<>();

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

            while (!st.isEmpty() && prices[st.peek()] <= prices[i]) {
                st.pop();
            }

            span[i] = st.isEmpty() ? (i + 1) : (i - st.peek());

            st.push(i);
        }

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

**Status :** Correct

**Marks :** 10/10

### 3. Problem Statement

A teacher is filtering a list of words provided by students. Some words contain too many vowels, making them difficult for a spelling competition.

The teacher decides to remove all words that contain more than two vowels.

Help the teacher to implement it using ArrayList.

### ***Input Format***

The first line contains an integer N, representing the number of words in the list.

The next N lines contain a string representing the words (one per line).

### ***Output Format***

The output consists of words that contain two or less than two vowels, printed in the same order they appeared in the input. Each word is printed on a new line.

Refer to the sample output for the formatting specifications.

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

Input: 1

sri

Output: sri

### ***Answer***

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

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

```
class VowelFilter {
```

```
    public static void filterWords(int n, Scanner sc) {
        ArrayList<String> list = new ArrayList<>();
```

```
        // Read words
```

```
        for (int i = 0; i < n; i++) {
            list.add(sc.nextLine());
        }
```

```

// Process and print valid words
for (String word : list) {
    if (countVowels(word) <= 2) {
        System.out.println(word);
    }
}

// Helper function to count vowels
private static int countVowels(String word) {
    int count = 0;
    for (char ch : word.toCharArray()) {
        if ("aeiou".indexOf(ch) != -1) {
            count++;
        }
    }
    return count;
}

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        sc.nextLine();
        VowelFilter.filterWords(n, sc);
        sc.close();
    }
}

```

**Status :** Correct

**Marks :** 10/10

#### 4. Problem Statement

Rahul is working on a list manipulation problem where he needs to reverse a specific subarray using a stack. Given an array and two indices  $l$  and  $r$ , he wants to reverse only the portion of the array from index  $l$  to  $r$  (both inclusive) while keeping the rest of the array unchanged.

Since Rahul wants to solve this problem efficiently, he decides to use a stack to reverse the subarray in  $O(r - l)$  time.

Your task is to help Rahul by implementing this functionality.

### ***Input Format***

The first line contains an integer  $n$ , the size of the array.

The second line contains  $n$  space-separated integers  $arr[i]$ .

The third line contains two integers  $l$  and  $r$ , denoting the start and end indices of the subarray to reverse.

Note: The array follows 0-based indexing.

### ***Output Format***

The output prints the modified array after reversing the subarray between indices  $l$  and  $r$ .

Refer to the sample output for formatting specifications.

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

Input: 6

1 2 3 4 5 6

1 4

Output: 1 5 4 3 2 6

### ***Answer***

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

```
public class Main {
```

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

```
    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 l = sc.nextInt();  
int r = sc.nextInt();  
  
Stack<Integer> stack = new Stack<>();  
  
// Push elements of the subarray into stack  
for (int i = l; i <= r; i++) {  
    stack.push(arr[i]);  
}  
  
// Pop back reversed values into the array  
for (int i = l; i <= r; i++) {  
    arr[i] = stack.pop();  
}  
  
// Print final array  
for (int i = 0; i < n; i++) {  
    System.out.print(arr[i] + " ");  
}  
}
```

**Status :** Correct

**Marks : 10/10**