

Name-Shivam Saini      uid-22bcs14156  
Sec-636/B              date-10/04/2025

**Problem 1.** Consider a function **public String matchFound(String input 1, String input 2)**, where

- **input1** will contain only a single word with only 1 character replaces by an underscore ‘\_’
- **input2** will contain a series of words separated by colons and no space character in between
- **input2** will not contain any other special character other than underscore and alphabetic characters.

The methods should return output in a String type variable “**output1**” which contains all the words from input2 separated by colon which matches with input 1. All words in output1 should be in uppercase.

**Code:**

```
public class MatchFinder {  
    public static String matchFound(String input1, String input2) {  
        String[] words = input2.split(":");  
        StringBuilder output1 = new StringBuilder();  
        for (String word : words) {  
            if (word.length() == input1.length()) {  
                boolean match = true;  
                for (int i = 0; i < word.length(); i++) {  
                    if (input1.charAt(i) != '_' && input1.charAt(i) != word.charAt(i)) {  
                        match = false;  
                        break;  
                    }  
                }  
                if (match) {  
                    if (output1.length() > 0) {  
                        output1.append(":");  
                    }  
                }  
            }  
        }  
    }  
}
```

```

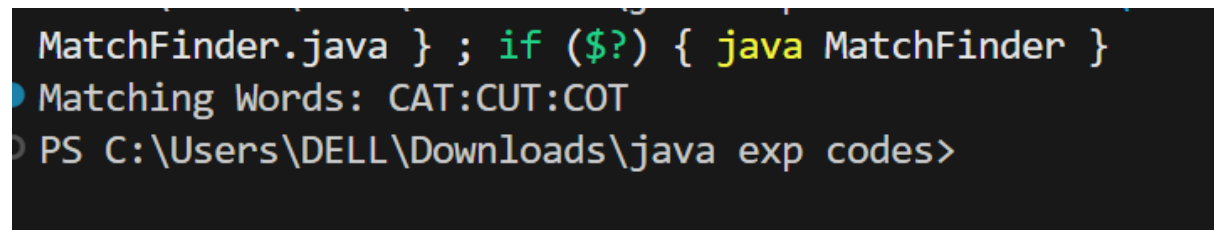
        output1.append(word.toUpperCase());
    }
}

return output1.toString();
}

// You can test it like this
public static void main(String[] args) {
    String input1 = "c_t";
    String input2 = "cat:cut:cot:bat:rat";
    String result = matchFound(input1, input2);
    System.out.println("Matching Words: " + result);
}
}

```

OUTPUT



```

MatchFinder.java } ; if ($?) { java MatchFinder }
Matching Words: CAT:CUT:COT
PS C:\Users\DELL\Downloads\java exp codes>

```

## Problem 2:

String `t` is generated by random shuffling string `s` and then add one more letter at a random position.  
Return the letter that was added to `t`.

**Hint:**

**Input:** `s = "abcd", t = "abcde"`

**Output:** `"e"`

**Code:**

```
public class FindAddedCharacter {
```

```

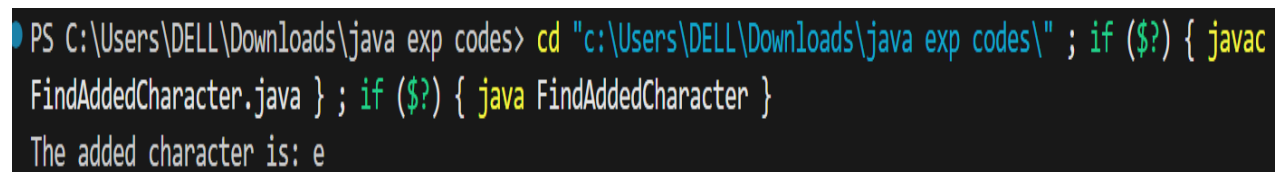
public static char findTheDifference(String s, String t) {
    int sumS = 0;
    int sumT = 0;
    for (char ch : s.toCharArray()) {
        sumS += ch;
    }
    for (char ch : t.toCharArray()) {
        sumT += ch;
    }
    return (char)(sumT - sumS);
}

// Test the function

public static void main(String[] args) {
    String s = "abcd";
    String t = "abcde";
    char result = findTheDifference(s, t);
    System.out.println("The added character is: " + result); // Output: e
}
}

```

### Output



```

PS C:\Users\DELL\Downloads\java exp codes> cd "c:\Users\DELL\Downloads\java exp codes\" ; if ($?) { javac
FindAddedCharacter.java } ; if ($?) { java FindAddedCharacter }
The added character is: e

```

### Problem 3:

The next greater element of some element  $x$  in an array is the first greater element that is to the right of  $x$  in the same array.

You are given two distinct 0-indexed integer arrays  $nums1$  and  $nums2$ , where  $nums1$  is a subset of  $nums2$ .

For each  $0 \leq i < \text{nums1.length}$ , find the index  $j$  such that  $\text{nums1}[i] == \text{nums2}[j]$  and determine the next greater element of  $\text{nums2}[j]$  in  $\text{nums2}$ . If there is no next greater element, then the answer for this query is -1.

Return an array `ans` of length `nums1.length` such that `ans[i]` is the next greater element as described above.

**Hint:**

Input: `nums1 = [4,1,2]`, `nums2 = [1,3,4,2]`

Output: `[-1,3,-1]`

Explanation: The next greater element for each value of `nums1` is as follows:

- 4 is underlined in `nums2 = [1,3,4,2]`. There is no next greater element, so the answer is -1.

- 1 is underlined in `nums2 = [1,3,4,2]`. The next greater element is 3.

- 2 is underlined in `nums2 = [1,3,4,2]`. There is no next greater element, so the answer is -1.

**Code:**

```
import java.util.*;

public class NextGreaterElement {

    public static int[] nextGreaterElement(int[] nums1, int[] nums2) {

        Map<Integer, Integer> map = new HashMap<>();

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

        for (int num : nums2) {

            while (!stack.isEmpty() && num > stack.peek()) {

                map.put(stack.pop(), num);

            }

            stack.push(num);

        }

        int[] ans = new int[nums1.length];

        for (int i = 0; i < nums1.length; i++) {

            ans[i] = map.getOrDefault(nums1[i], -1);

        }

        return ans;

    }

    public static void main(String[] args) {

        int[] nums1 = {4, 1, 2};
```

```

int[] nums2 = {1, 3, 4, 2};

int[] result = nextGreaterElement(nums1, nums2);

System.out.print("Output: ");

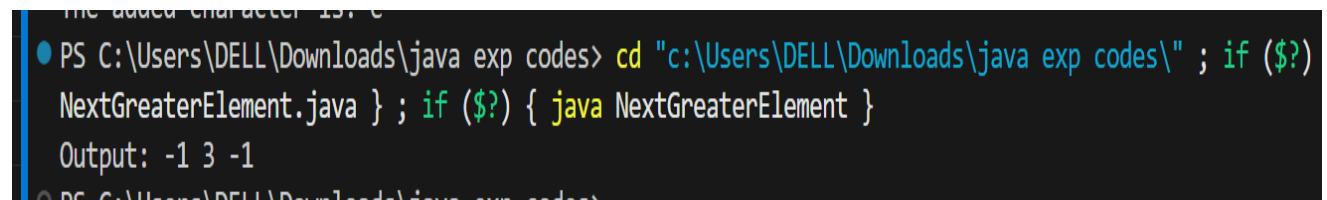
for (int num : result) {
    System.out.print(num + " ");
}

}

}

```

## OUTPUT



```

PS C:\Users\DELL\Downloads\java exp codes> cd "c:\Users\DELL\Downloads\java exp codes\" ; if ($?) {
NextGreaterElement.java } ; if ($?) { java NextGreaterElement }
Output: -1 3 -1

```

## Problem 4:

A string containing only parentheses is balanced if the following is true: 1. if it is an empty string 2. if A and B are correct, AB is correct, 3. if A is correct, (A) and {A} and [A] are also correct.

Examples of some correctly balanced strings are: "{}()", "[{}]", "({})"

Examples of some unbalanced strings are: "{}(", "({})", "[[", "{}{" etc.

Given a string, determine if it is balanced or not.

### Input Format

There will be multiple lines in the input file, each having a single non-empty string. You should read input till end-of-file.

### Output Format

For each case, print 'true' if the string is balanced, 'false' otherwise.

### Sample Input

```
{ } ( { } ) { } ( [ ]
```

### Sample Output

```
true true false true
```

## Code

```

import java.util.*;
import java.io.*;

```

```

public class BalancedParentheses {

    // Method to check if a single string is balanced
    public static boolean isBalanced(String s) {
        Stack<Character> stack = new Stack<>();
        Map<Character, Character> pair = new HashMap<>();
        pair.put('(', '(');
        pair.put('}', '{');
        pair.put(']', '[');

        for (char ch : s.toCharArray()) {
            if (ch == '(' || ch == '{' || ch == '[') {
                stack.push(ch);
            } else if (ch == ')' || ch == '}' || ch == ']') {
                if (stack.isEmpty() || stack.pop() != pair.get(ch)) {
                    return false;
                }
            }
        }

        return stack.isEmpty();
    }

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

        while (sc.hasNextLine()) {
            String line = sc.nextLine().trim();
            if (!line.isEmpty()) {
                boolean result = isBalanced(line);
                System.out.println(result);
            }
        }

        sc.close();
    }
}

```

## Output:

```

Output: -1 3 -1
PS C:\Users\DELL\Downloads\java exp codes> cd "c:\Users\DELL\Downloads\java exp codes\" ; if ($?) {
BalancedParentheses.java } ; if ($?) { java BalancedParentheses }
cd "c:\Users\DELL\Downloads\java exp codes\" ; if ($?) { javac BalancedParentheses.java } ; if ($?) {
BalancedParentheses }
true

```

## Problem 5

Given an array of integers `nums` sorted in non-decreasing order, find the starting and ending position of a given target value.

If target is not found in the array, return `[-1, -1]`.

You must write an algorithm with  $O(\log n)$  runtime complexity.

**Example 1:**

**Input:** `nums = [5,7,7,8,8,10]`, `target = 8`

**Output:** `[3,4]`

**Constraints:**

- $0 \leq \text{nums.length} \leq 10^5$
- $-10^9 \leq \text{nums}[i] \leq 10^9$
- `nums` is a non-decreasing array.
- $-10^9 \leq \text{target} \leq 10^9$

**Code:**

```
public class FindFirstAndLastPosition {

    public static int[] searchRange(int[] nums, int target) {
        int[] result = new int[2];
        result[0] = findFirst(nums, target);
        result[1] = findLast(nums, target);
        return result;
    }

    private static int findFirst(int[] nums, int target) {
        int left = 0, right = nums.length - 1;
        int first = -1;
        while (left <= right) {
            int mid = left + (right - left) / 2;
            if (nums[mid] == target) {
                first = mid;
                right = mid - 1;
            } else if (nums[mid] < target) {
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        }
        return first;
    }

    private static int findLast(int[] nums, int target) {
        int left = 0, right = nums.length - 1;
        int last = -1;
        while (left <= right) {
            int mid = left + (right - left) / 2;
            if (nums[mid] == target) {
                last = mid;
                left = mid + 1;
            } else if (nums[mid] < target) {
```

```
        left = mid + 1;
    } else {
        right = mid - 1;
    }
}
return last;
}
public static void main(String[] args) {
    int[] nums = {5, 7, 7, 8, 8, 10};
    int target = 8;
    int[] result = searchRange(nums, target);
    System.out.println "[" + result[0] + "," + result[1] + ""];
}
}
```

### OUTPUT:

```
cd "c:\Users\DELL\Downloads\java exp codes\" ; if ($?) { javac FindFirstAndLastPosition.java } ; if ($?) {
    java FindFirstAndLastPosition }
true
```