PBLJ_Complex Problems (lab Assignment)

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.

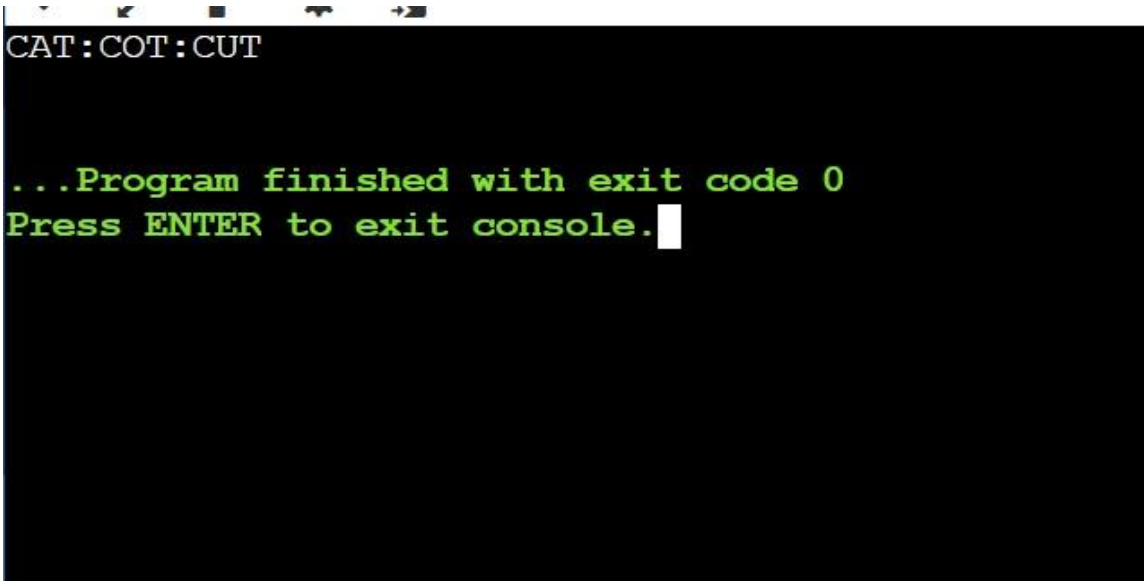
The methods should return output in a String type variable "output1" which contains all the words from input2 separated by colon which matches with input 1. All words in output1 should be in uppercase.

Code:-

```
public class MatchFinder {

   public static String matchFound(String input1, String input2) {

      StringBuilder output1 = new StringBuilder();

String[] words = input2.split(":")        for (String

word : words) {            if (matchesPattern(input1,

word)) {               if (output1.length() > 0) {

output1.append(":");

            }

            output1.append(word.toUpperCase());

         }

      }

      return output1.toString();

   }

  private static boolean matchesPattern(String pattern, String word) {

if (pattern.length() != word.length()) {            return false;

      }

      int underscoreIndex = pattern.indexOf('_');

if (underscoreIndex == -1) {            return false;


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

         if (i != underscoreIndex && pattern.charAt(i) != word.charAt(i)) {
```

```
            return false;
        }
    }
    return true;
  }
  public static void main(String[] args) {
    String input1 = "c_t";
    String input2 = "cat:cot:cut:bat:rat";

    String result = matchFound(input1, input2);
    System.out.println(result); // Output: CAT:COT:CUT
  }
}
```

OUTPUT:-



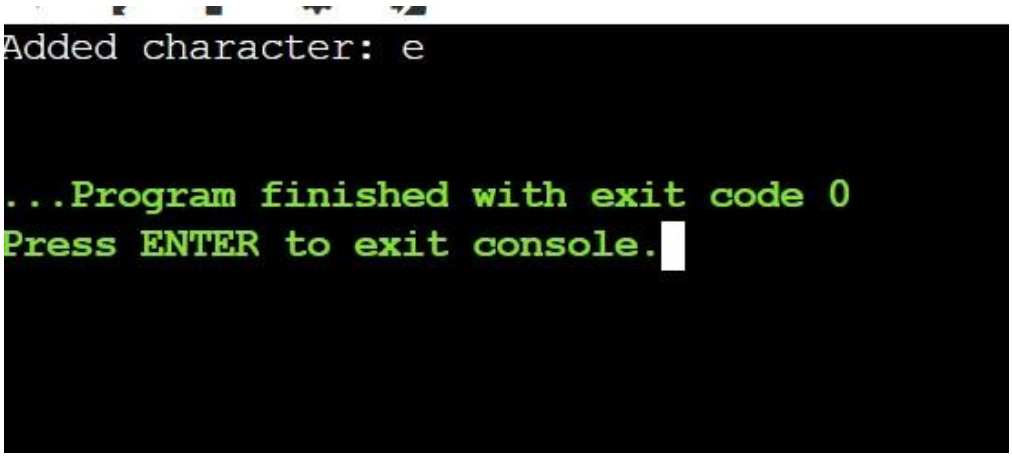
Problem 4: String t is generated by random shuffling string s and then add one more letter at a random position. Return the letter that was added to t.

Hint: Input: s = "abcd", t = "abcde" Output: "e" **Code:-**

```
public class FindAddedLetter {
```

```
   public static char findTheDifference(String s, String t) {
int result = 0;
      // XOR all characters in s
for (char c : s.toCharArray()) {
result ^= c;
      }
      // XOR all characters in t
for (char c : t.toCharArray()) {
result ^= c;
      }
      // The remaining value is the added character
return (char) result;
   }
   public static void main(String[] args) {
      String s = "abcd";
String t = "abcde"
      char addedChar = findTheDifference(s, t);
     System.out.println("Added character: " + addedChar);
}
```

Output: -



Problem 6: 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 nonempty 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.*;

public class BalancedParentheses {     public

static boolean isBalanced(String s) {

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

for (char c : s.toCharArray()) {

          if (c == '(' || c == '{' || c == '[') {

stack.push(c);

          } else if (c == ')' || c == '}' || c ==

']') {                if (stack.isEmpty()) {

return false;

              }

              char top = stack.pop();

if ((c == ')' && top != '(') ||

(c == '}' && top != '{') ||

(c == ']' && top != '[')) {

return false;

              }

          }

      }

      return stack.isEmpty();

   }

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

```
        Scanner scanner = new Scanner(System.in);
while (scanner.hasNext()) {
            String input = scanner.next();
            System.out.println(isBalanced(input));
        }
        scanner.close();
    }
}
```

OUTPUT:-

```
(){}{}
true
{()}
true
{(})
false
[{()}]
true
```

Problem 10: 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 <= nums.length <= 105 • -109 <= nums[i] <= 109 • nums is a nondecreasing array. • -109 <= target <= 10 Code:-

```
import java.util.*; public class
FindTargetRange {     public
static int[] searchRange(int[]
nums, int target) {        int first =
```

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

          }
       }
       return result;
   }


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

```
        System.out.println("Enter the number of elements:");
int n = scanner.nextInt();        int[] nums = new int[n];
        System.out.println("Enter the elements in sorted order:");
for (int i = 0; i < n; i++) {            nums[i] = scanner.nextInt();
        }
        System.out.println("Enter the target value:");
int target = scanner.nextInt();
        System.out.println(Arrays.toString(searchRange(nums, target)));        scanner.close();
    }
```

}**OUTPUT:-**