LAB ASSIGNMENT(Fast Learner)

Student Name: Aakarsh Chandna **UID: 22BCS10344**
Branch: CSE **Section:618(A)**

Semester: 6th **DOP:04/04/2025**

Subject: Java **Subject Code: 22CSH-359**

Problem Statement 1

1. **Aim:** Encoding Three Strings: Anand was assigned the task of coming up with an encoding mechanism for any given three string.

2. **Code:**

```
import java.util.Scanner; public
class StringEncoder {
   // Method to split the string into Front, Middle, and End
parts    public static String[] splitString(String str) {        int len =
str.length();        int partSize = len / 3;        int remainder = len % 3;
int frontSize = partSize + (remainder == 2 ? 1 : 0);        int
middleSize = partSize + (remainder == 1 ? 1 : 0);        int endSize =
partSize;
      String front = str.substring(0, frontSize);
      String middle = str.substring(frontSize, frontSize + middleSize);
      String end = str.substring(frontSize + middleSize);

      return new String[]{front, middle, end};
   }
```

```
    // Method to toggle the case of a string     public
static String toggleCase(String str) {
StringBuilder toggled = new StringBuilder();         for
(char ch : str.toCharArray()) {             if
(Character.isUpperCase(ch)) {
toggled.append(Character.toLowerCase(ch));
            } else {
                toggled.append(Character.toUpperCase(ch));
            }
        }
        return toggled.toString();
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Taking three input strings
        System.out.print("Enter first string: ");
        String input1 = scanner.nextLine();
        System.out.print("Enter second string: ");
        String input2 = scanner.nextLine();
        System.out.print("Enter third string: ");
        String input3 = scanner.nextLine();

        // Splitting strings into parts
        String[] parts1 = splitString(input1);
```

```
        String[] parts2 = splitString(input2);

        String[] parts3 = splitString(input3);
        // Creating the output strings based on the given rules

        String output1 = parts1[0] + parts2[1] + parts3[2]; // FRONT1 + MIDDLE2 + END3

        String output2 = parts1[1] + parts2[2] + parts3[0]; // MIDDLE1 + END2 + FRONT3

        String output3 = parts1[2] + parts2[0] + parts3[1]; // END1 + FRONT2 + MIDDLE3


        // Toggling case for output3
output3 = toggleCase(output3);


        // Displaying results

        System.out.println("\nFinal Encoded Outputs:");

        System.out.println("Output 1: " + output1);

        System.out.println("Output 2: " + output2);
System.out.println("Output 3: " + output3);

scanner.close();

    }

}
```

4. Output:

```
Enter first string: John
Enter second string: Johny
Enter third string: Janardhan

Final Encoded Outputs:
Output 1: Jnhan
Output 2: ohnyJan
Output 3: NJOARD
```

Problem Statement 2

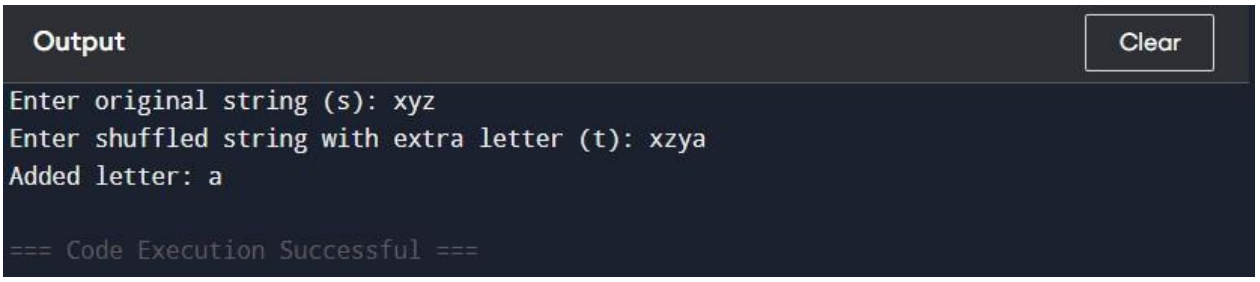
1. **Aim:** String t is generated by random shuffling string s and then add one more letter at a random position. Return the letter that was added to t.

2. **Code:**

```
import java.util.Scanner;

public class FindExtraLetter {
   public static char findAddedLetter(String s, String t) {
char result = 0;

      // XOR all characters in both
strings       for (char c : s.toCharArray())
{           result ^= c;
      }
      for (char c : t.toCharArray()) {
result ^= c;
      }       return result; // The
extra letter
   }
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);

      // Taking user input
      System.out.print("Enter original string (s): ");
      String s = scanner.nextLine();

      System.out.print("Enter shuffled string with extra letter (t): ");
      String t = scanner.nextLine();
```

```
        // Finding and displaying the extra letter
    char extraLetter = findAddedLetter(s, t);
        System.out.println("Added letter: " + extraLetter);

        scanner.close();
    }
}
```

3. Output:



Problem Statement 3

1. **Aim:** 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.

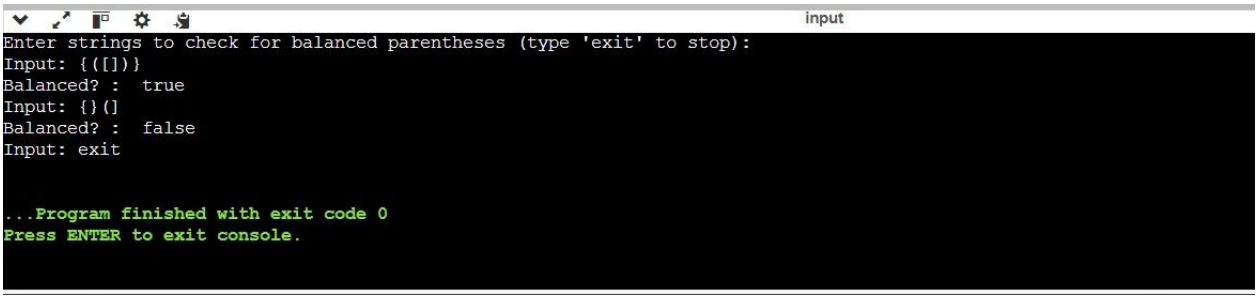
   Given a string, determine if it is balanced or not. Input Format There will be multiple lines in the input file, each having a single non-empty string. You should read input till end-offile. Output Format For each case, print 'true' if the string is balanced, 'false' otherwise.

2. **Code:**

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

public class BalancedParentheses {     public
static boolean isBalanced(String s) {
Stack<Character> stack = new Stack<>();
```

```
        for (char ch : s.toCharArray()) {
if (ch == '(' || ch == '{' || ch == '[') {
stack.push(ch);
            } else if (ch == ')' || ch == '}' || ch == ']')
{                 if (stack.isEmpty()) return false;
char top = stack.pop();                 if ((ch == ')'
&& top != '(') ||                     (ch == '}' && top
!= '{') ||                   (ch == ']' && top != '[')) {
return false;
                }
            }
}
return
stack.is
Empty()
;
    }

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

        System.out.println("Enter strings to check for balanced parentheses (type 'exit' to
stop):");
        while (true) {
            System.out.print("Input: ");
String input = scanner.nextLine();
            if (input.equalsIgnoreCase("exit")) break; // Stop on 'exit'
            System.out.println("Balanced? " + isBalanced(input));
        }
scanner.close();
    }
}
```

3. Output



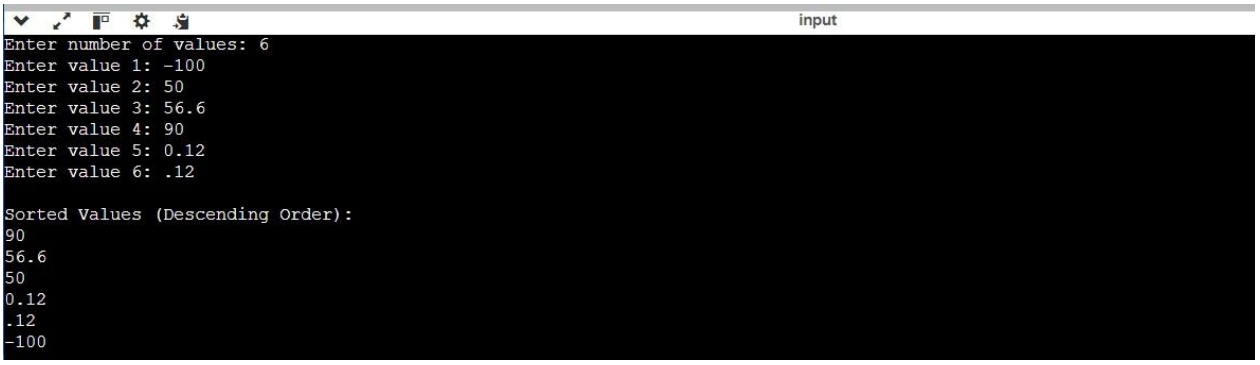
Problem Statement 4

1. **Aim:** Given an array of real number strings, sort them in descending order — but wait, there's more! Each number must be printed in the exact same format as it was read from stdin, meaning that is printed as , and is printed as . If two numbers represent numerically equivalent values (e.g.,), then they must be listed in the same order as they were received as input). You must rearrange array 's elements according to the instructions above.

2. **Code:**

```
import java.math.BigDecimal; import
java.util.*;

public class BigDecimalSort {
   public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
System.out.print("Enter number of values: ");
        int n = Integer.parseInt(scanner.nextLine()); // Read the number of inputs

        String[] numbers = new String[n];

        // Read input numbers as strings
for (int i = 0; i < n; i++) {
            System.out.print("Enter value " + (i + 1) + ": ");
numbers[i] = scanner.nextLine();
        }

        // Sort in descending order using BigDecimal for precision
        Arrays.sort(numbers, (a, b) -> new BigDecimal(b).compareTo(new BigDecimal(a)));

        // Print the sorted numbers while keeping the original format
System.out.println("\nSorted Values (Descending Order):");        for
(String num : numbers) {
            System.out.println(num);
        }
scanner.close();
    }
}
```

3. **Output:**



Problem Statement 5

1. **Aim:** 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.

2. **Code:**

```
import java.util.Scanner; 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};
   }
   // Binary search to find the first occurrence of target
private static int findFirst(int[] nums, int target) {        int
left = 0, right = nums.length - 1, index = -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) index = mid;
      }
      return index;
   }
   // Binary search to find the last occurrence of target
private static int findLast(int[] nums, int target) {        int
left = 0, right = nums.length - 1, index = -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) index = mid;
      }
      return index;
   }
```

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

        // Taking input from the user
        System.out.print("Enter number of elements in the array: ");
int n = scanner.nextInt();          int[] nums = new int[n];
        System.out.println("Enter sorted array elements:");
for (int i = 0; i < n; i++) {               nums[i] =
scanner.nextInt();
        }
        System.out.print("Enter target value: ");
int target = scanner.nextInt();
        int[] result = searchRange(nums, target);
        System.out.println("Output: [" + result[0] + ", " + result[1] + "]");
scanner.close();
    }
}
```

3. Output: