Assignment-2

Student Name: Akash Singh
UID: 22BCS12046
Branch: CSE
Section/Group: IOT-618/A
Semester: 6th
Date of Performance: 4/04/25
Subject Name: Java Lab
Subject Code: 22CSH-359

Problem-2

Aim:

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 <= i < nums1.length, find the index j such that nums1[i] == nums2[j] and determine the next greater element of nums2[j] in 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.

Implementation/Code:

```
import java.util.*;

class Solution {
    public int[] nextGreaterElement(int[] nums1, int[] nums2) {
        Map<Integer, Integer> nextGreaterMap = new HashMap<>();
        Stack<Integer> stack = new Stack<>();

        // Traverse nums2 in reverse order
        for (int num : nums2) {
            // Maintain a decreasing stack
            while (!stack.isEmpty() && stack.peek() <= num) {
                stack.pop();
            }
            // Store the next greater element in the map
            nextGreaterMap.put(num, stack.isEmpty() ? -1 : stack.peek());
            stack.push(num);
```

```
        }

        // Process nums1 and fetch results from the map
        int[] result = new int[nums1.length];
        for (int i = 0; i < nums1.length; i++) {
            result[i] = nextGreaterMap.get(nums1[i]);
        }

        return result;
    }

    public static void main(String[] args) {
        Solution sol = new Solution();
        int[] nums1 = {4, 1, 2};
        int[] nums2 = {1, 3, 4, 2};
        System.out.println(Arrays.toString(sol.nextGreaterElement(nums1, nums2)));
    }
}
```

3. Output:

Output

```
[-1, -1, 4]

=== Code Execution Successful ===
```

Problem-4

1. Aim:

How can we encode three given strings by **splitting them into three parts** (Front, Middle, and End) based on their length? How do we ensure that if the length is a **multiple of 3**, the parts are equal; if it's **(3x+1)**, the **middle** gets an extra character; and if it's **(3x+2)**, the **front and end** get extra characters? Once split, how do we **concatenate** the parts to form three output strings using the rules: Output1 = FRONT1 + MIDDLE2 + END3, Output2 = MIDDLE1 + END2 + FRONT3, and Output3 = END1 + FRONT2 + MIDDLE3? Finally, how do we **toggle the case** of all characters in Output3 to complete the encoding process?

2. Implementation/Code:

```
    import java.util.*;
    import java.util.stream.Collectors;

    class Student {     String name;
    double marks;     // Constructor
  public Student(String name, double marks) {
      this.name = name;
     this.marks = marks;
   }

   // Display method
 public void display() {
  System.out.println(name + " (Marks: " + marks + ")");
   }
}

public class StudentFilterSort {
  public static void main(String[] args) {
      // Creating a list of students
      List<Student> students = new ArrayList<>();
     students.add(new Student("Alice", 80));
    students.add(new Student("Bob", 72));
  students.add(new Student("Charlie", 90));
```

```
        students.add(new Student("David", 65));
        students.add(new Student("Eve", 85));

        System.out.println("Students who scored above 75%, sorted by marks:");

        // Filtering students with marks > 75%, sorting in descending order, and collecting results
        List<Student> filteredStudents = students.stream()
            .filter(student -> student.marks > 75)  // Filter students above 75%
            .sorted(Comparator.comparingDouble((Student student) -> -student.marks)
.thenComparing(student -> student.name))  // Sort by marks (descending) & name (ascending)
            .collect(Collectors.toList());

        // Displaying the sorted students
    if (filteredStudents.isEmpty()) {
            System.out.println("No students scored above 75%.");
        } else {
                filteredStudents.forEach(Student::display);
            }
        }
    }
```

3. Output:

```
Enter first string: Aman
Enter second string: Rahul
Enter third string: John
Output 1: Ahn
Output 2: maulJ
Output 3: NrAOH

=== Code Execution Successful ===
```

Problem-6

1. Aim:

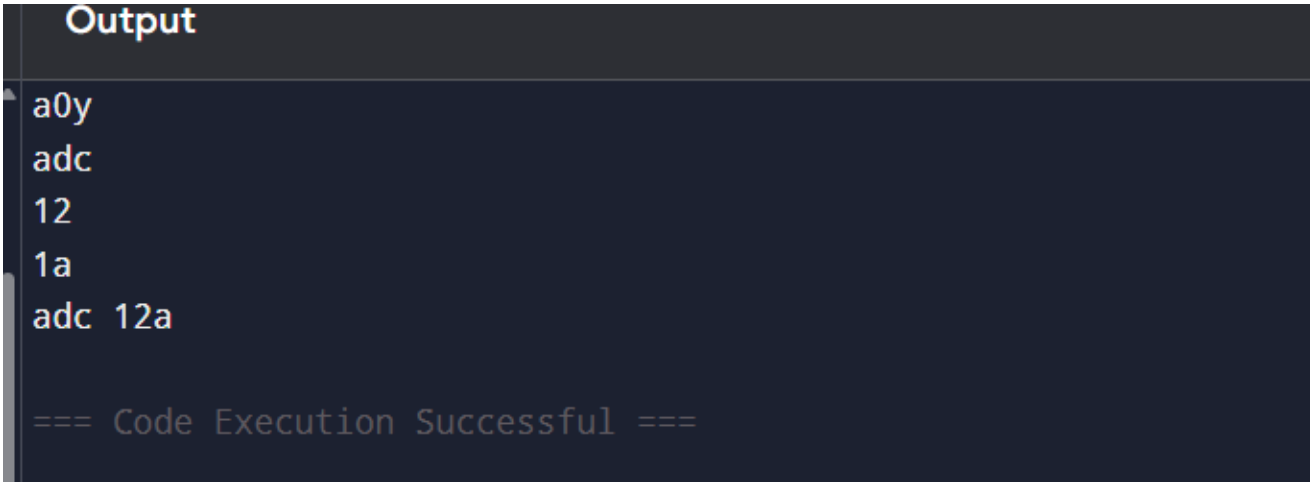
Given a String (In Uppercase alphabets or Lowercase alphabets), new alphabets is to be appended with following rule:
(i) If alphabet is present in input string, use numeric value of that alphabet. E.g. a or A numeric value is 1 and so on. New alphabet to be appended between 2 alphabets :
(a) If (sum of numeric value of 2 alphabets) %26 is 0, then append 0. E.g. string is ay. Numeric value of a is 1, y is 25. Sum is 26. Remainder is 0, new string will be a0y.
(b) Otherwise (sum of numeric value of 2 alphabets) %26 numeric value alphabet is to appended. E.g. ac is string. Numeric value of a is 1, c is 3, sum is 4. Remainder with 26 is 4. Alphabet to be appended is d. output will be adc.
(ii) If digit is present, it will be same in output string. E.g. string is 12, output string is 12.
(iii) If only single alphabet is present, it will be same in output string. E.g. input string is 1a, output will be 1a.
(iv) If space is present, it will be same in output string. E.g. string is ac 12a, output will be adc 12a.
Constraint: Whether string alphabets are In Uppercase or Lowercase, appended alphabets must be in lower case. Output string must also be in lowercase.

Implementation/Code:

```
public class StringEncoder {
   public static String encodeString(String s) {
      StringBuilder result = new StringBuilder();
      s = s.toLowerCase(); // Convert entire input to lowercase

      for (int i = 0; i < s.length(); i++) {
         result.append(s.charAt(i)); // Append current character

         if (i < s.length() - 1 && Character.isLetter(s.charAt(i)) && Character.isLetter(s.charAt(i +
1))) {
            int num1 = s.charAt(i) - 'a' + 1;
            int num2 = s.charAt(i + 1) - 'a' + 1;
            int sum = num1 + num2;
            result.append(sum % 26 == 0 ? "0" : (char) ('a' + (sum % 26) - 1));
         }
```

```
        }
        return result.toString();
    }

    public static void main(String[] args) {
        System.out.println(encodeString("ay"));      // Output: a0y
        System.out.println(encodeString("ac"));      // Output: adc
        System.out.println(encodeString("12"));      // Output: 12
        System.out.println(encodeString("1a"));      // Output: 1a
        System.out.println(encodeString("ac 12a"));  // Output: adc 12a
    }
}
```

Output:



Problem-8

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. Hint: Input: s = "abcd", t = "abcde" Output: "e

Implementation/Code:

```
public class FindAddedCharacter {
    public static char findTheDifference(String s, String t) {
        char result = 0;

        // XOR all characters from both strings
        for (char c : s.toCharArray()) result ^= c;
        for (char c : t.toCharArray()) result ^= c;

        return result;
    }

    public static void main(String[] args) {
        String s = "abcd";
        String t = "abcde";
        System.out.println(findTheDifference(s, t));  // Output: e
    }
}
```

Output:

```
Output

e

=== Code Execution Successful ===
```

Problem-1

Aim:

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.

Implementation/Code:

```
public class MatchFinder {
    public static String matchFound(String input1, String input2) {
        String[] words = input2.split(":"); // Split input2 into words
        StringBuilder output1 = new StringBuilder();
        input1 = input1.toUpperCase(); // Convert input1 to uppercase

        for (String word : words) {
            if (matchesPattern(input1, word.toUpperCase())) {
                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;

        for (int i = 0; i < pattern.length(); i++) {
            if (pattern.charAt(i) != '_' && pattern.charAt(i) != word.charAt(i)) {
                return false;
            }
        }
```

```
        return true;
    }

    public static void main(String[] args) {
        System.out.println(matchFound("h_llo", "hello:hullo:hallo:hilly")); // Output:
HELLO:HULLO:HALLO
        System.out.println(matchFound("t_st", "test:best:past:toast")); // Output: TEST
        System.out.println(matchFound("c_d", "cad:cod:cat:cud")); // Output: CAD:COD:CUD
    }
}
```

Output:

```
HELLO:HULLO:HALLO
TEST
CAD:COD:CUD

=== Code Execution Successful ===
```