Experiment 5

Student Name: Zatch **UID:**

Branch: CSE **Section/Group:**

Semester:5 **Date of Performance:**

Subject Name: AP **Subject Code: 22CSP-314**

1. **Aim:**

   Problem Statement: - There is a sequence of words in CamelCase as a string of letters s, having the following properties:

   • It is a concatenation of one or more words consisting of English letters.

   • All letters in the first word are lowercase.

   • For each of the subsequent words, the first letter is uppercase and rest of the letters are lowercase. Given, determine the number of words in s

2. **Objective:**

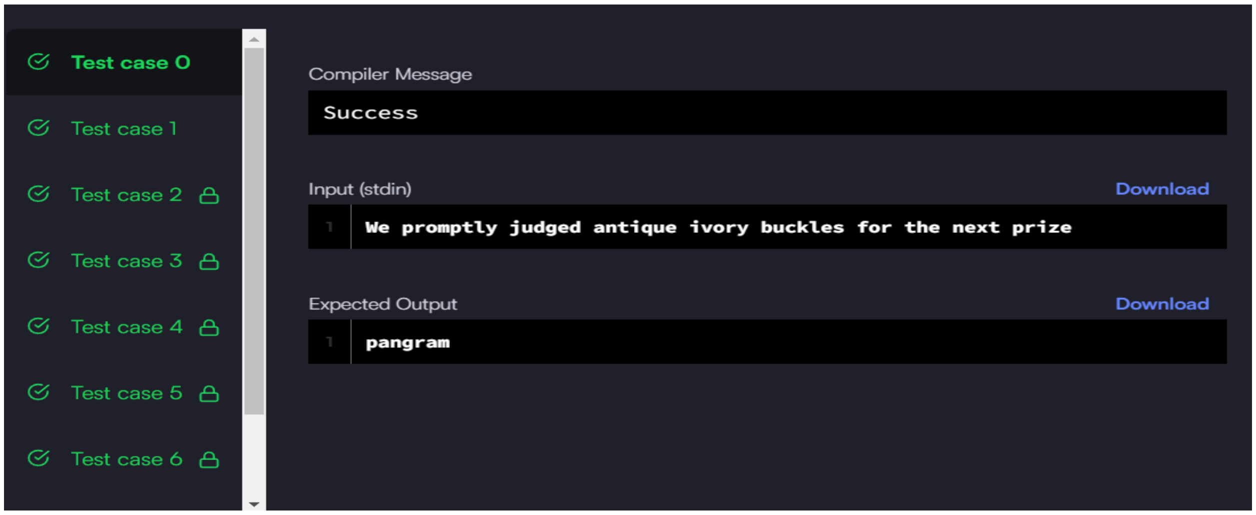
   The objective of this experiment is to implement a function in Java that counts the number of words in a camel case formatted string.

3. **Implementation/Code:**

```
class Result {

    public static String pangrams(String s) {
    // Write your code here
     s = s.toLowerCase();
```

```
        // Create a boolean array to track presence of each letter (26 letters in
alphabet)
        boolean[] seen = new boolean[26];

        // Iterate over each character in the string
        for (char c : s.toCharArray()) {
            // Check if character is a letter
            if (c >= 'a' && c <= 'z') {
                // Mark the character as seen
                seen[c - 'a'] = true;
            }
        }

        // Check if all letters have been seen
        for (boolean letterSeen : seen) {
            if (!letterSeen) {
                return "not pangram";
            }
        }

        return "pangram";

    }

}
```

4. Output:



Compiler Message

```
Success
```

Input (stdin)

```
We promptly judged antique ivory buckles for the next prize
```

Expected Output

```
pangram
```

Problem -2

1. **Aim:**

Problem Statement: - Louise joined a social networking site to stay in touch with her friends. The signup page required her to input a name and a password. However, the password must be strong. The website considers a password to be strong if it satisfies the following criteria:

• Its length is at least.

• It contains at least one digit.

• It contains at least one lowercase English character.

• It contains at least one uppercase English character.

• It contains at least one special character. The special characters are: !@#$%^&*()-+.

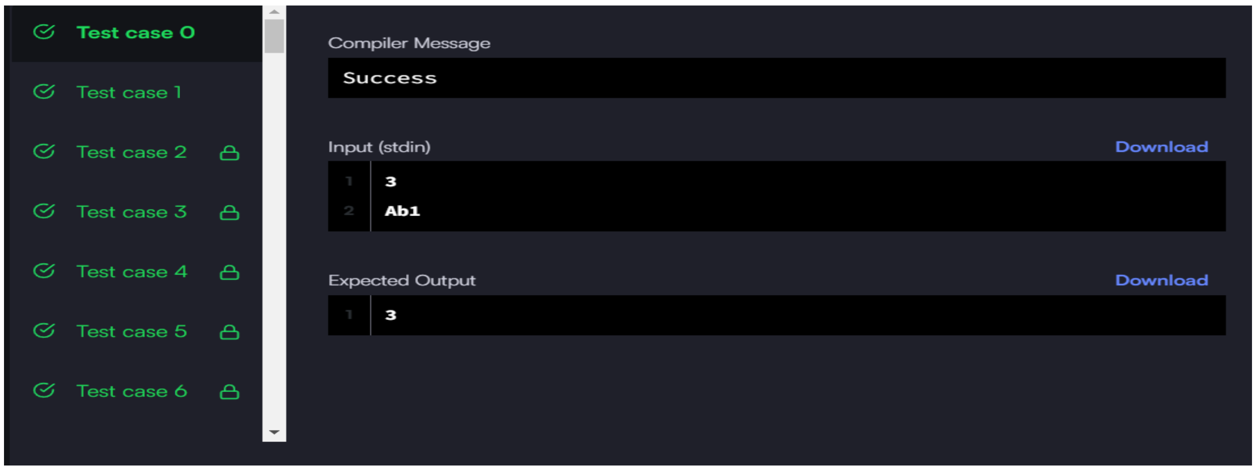
2. **Objective:**

   The objective of this experiment is to implement a function in Java that determines the minimum number of characters needed to make a given password "strong" according to specific criteria.

3. **Code:**

```
class Result {

    public static int minimumNumber(int n, String password) {
    // Return the minimum number of characters to make the password strong
     int count = 0;
        // Flags to check for each type of character
        boolean hasDigit = false;
        boolean hasLowerCase = false;
        boolean hasUpperCase = false;
        boolean hasSpecialChar = false;

        // Special characters set
        String specialCharacters = "!@#$%^&*()-+";
        // Check each character in the password
        for (char ch : password.toCharArray()) {
            if (Character.isDigit(ch)) {
                hasDigit = true;
            } else if (Character.isLowerCase(ch)) {
                hasLowerCase = true;
            } else if (Character.isUpperCase(ch)) {
                hasUpperCase = true;
            } else if (specialCharacters.contains(Character.toString(ch))) {
                hasSpecialChar = true;
            }
        }
        // Count missing character types
        if (!hasDigit) count++;
        if (!hasLowerCase) count++;
        if (!hasUpperCase) count++;
        if (!hasSpecialChar) count++;

int additionalLength = Math.max(6 - n, 0);

        additional length needed
        return Math.max(count, additionalLength);
  }
```

4. Output:



Compiler Message

```
Success
```

Input (stdin)

```
3
Ab1
```

Expected Output

```
3
```

5. Learning Outcome

i. Learn to efficiently process strings and manipulate individual characters

ii. Learn to work with strings in Java, including iterating over characters, checking character properties, and applying conditions based on those properties.

iii. Understand basic password strength requirements, which is essential for developing secure applications.

iv. Develop skills in analyzing strings and manipulating characters in Java, which is critical in many programming tasks.