Experiment 2.1

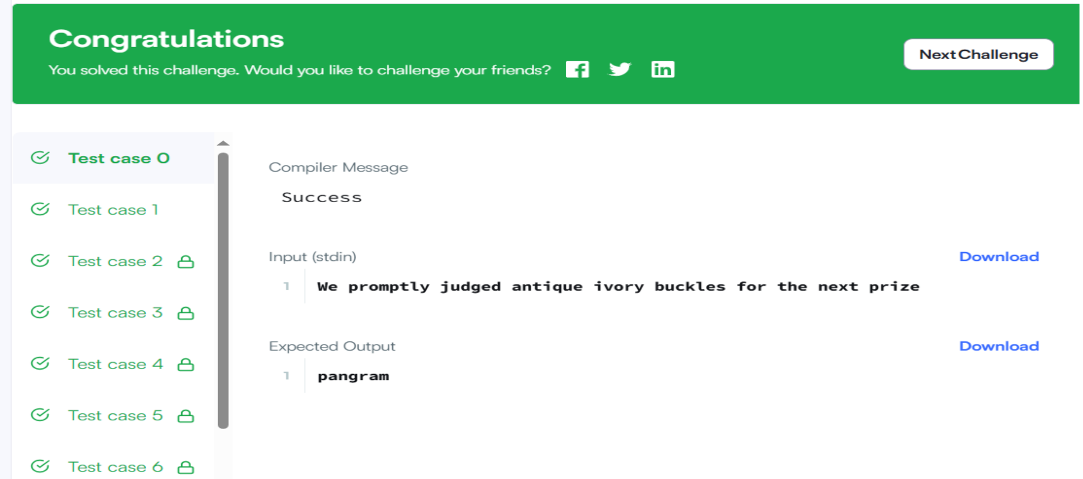
Student Name: Mohit Kumar Sharma UID:22BCS16518
Branch: BE CSE Section/Group: FL-IoT-604-A
Semester: 5th Date of Performance:14/08/2024
Subject Name: Advance Programming Lab-1 Subject Code: 22CSP-314

Problem 1:

1. **Aim:** A *pangram* is a string that contains every letter of the alphabet. Given a sentence determine whether it is a pangram in the English alphabet. Ignore case. Return either pangram or not pangram as appropriate.
   Objective: a**.** To learn about String data Structure.
   · b. To learn different approaches to find pangram.
2. **Algo:**
   - Initialize an array of size 26 to mark the presence of each letter in the alphabet.
   - Loop through the string, convert each letter to lowercase, and mark its corresponding index in the array.
   - Check the array to ensure every index has been marked; if any index is unmarked, return "not pangram."
   - If all indices are marked, return "pangram."
3. **Code:**

```
#include <iostream>
#include <vector>
#include <string>
#include <cctype>
#include <cstdlib>
using namespace std;
string pangrams(string s) {
    vector<bool> alphabet(26, false);  // Track each letter
    int index;
    for (char c : s) {
        if (isalpha(c)) {
            index = tolower(c) - 'a';
            alphabet[index] = true;
        }
    }
    for (bool present : alphabet) {
        if (!present) {
            return "not pangram";
        }
    }
    return "pangram";
}
int main() {
    string s;
    getline(cin, s);
    string result = pangrams(s);
    cout << result << endl;
    return 0;
}
```

4. Output:



Congratulations

You solved this challenge. Would you like to challenge your friends?

Compiler Message

```
Success
```

Input (stdin)

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

Expected Output

```
pangram
```

Time Complexity: O(n)

Problem 2:

1. **Aim:** There is a sequence of words in CamelCase as a string of letters, , 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*.

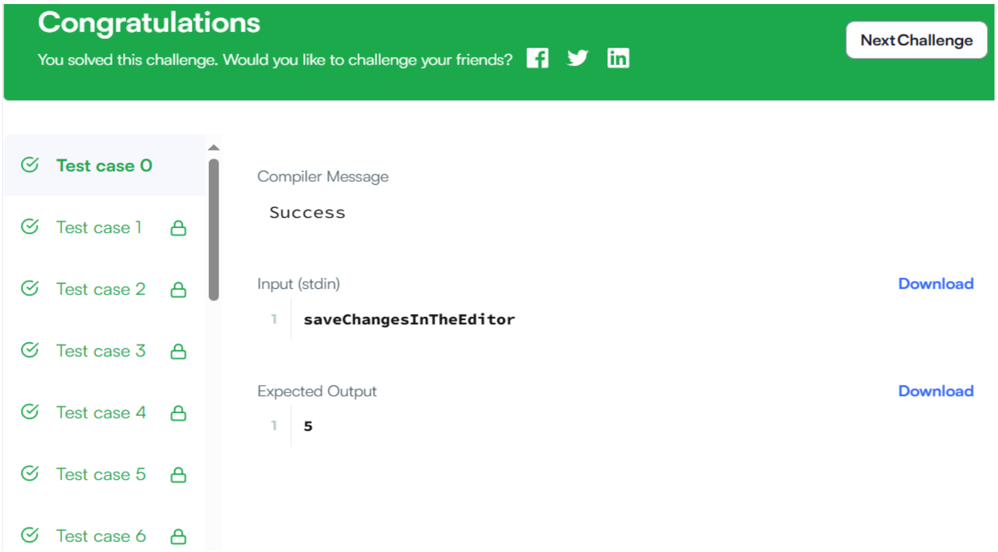
Given s, determine the number of words in s.

2. **Objective:** Learn different approaches to determine camel case.
3. **Algo:**
    - Start by initializing a counter count to 1, since the first word is in lowercase.
    - Loop through each character in the string s.
    - Increment the counter every time an uppercase letter is found, indicating the start of a new word.
    - Return the counter as the total number of words in the CamelCase string.
4. **Code:**

```
#include <iostream>
#include <string>
#include <cctype>

using namespace std;

// Function to count the number of words in a camelCase string
int camelcase(const string& s) {
    if (s.empty()) {
        return 0; // No words in an empty string
    }
    int count = 1; // Start with 1 because the first word is always lowercase

    for (char c : s) {
        if (isupper(c)) {
            count++;
        }
    }

    return count;
}

// Function to read a string from the user with validation
string readInput() {
    string input;

    cout << "Enter a camelCase string: ";
    getline(cin, input);

    // Validate the input to ensure it's non-empty and only contains alphabetic
characters
    while (input.empty() ||
input.find_first_not_of("abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLM
NOPQRSTUVWXYZ") != string::npos) {
        cout << "Invalid input. Please enter a non-empty camelCase string with
only alphabetic characters: ";
        getline(cin, input);
    }

    return input;
}
```

```
// Main function
int main() {
    // Read input from the user
    string s = readInput();

    // Calculate the number of words in the camelCase string
    int result = camelcase(s);

    // Output the result
    cout << "The number of words in the camelCase string is: " << result <<
endl;

    // Additional Information
    cout << "Explanation: The string \"" << s << "\" contains " << result << "
word(s)." << endl;

    // Example Breakdown
    cout << "Example Breakdown:\n";
    for (size_t i = 0; i < s.length(); i++) {
        if (i == 0) {
            cout << "Word 1: ";
        } else if (isupper(s[i])) {
            cout << "\nWord " << camelcase(s.substr(0, i + 1)) << ": ";
        }
        cout << s[i];
    }
    cout << endl;

    return 0;
}
```

5. Output:



Congratulations

You solved this challenge. Would you like to challenge your friends?

Compiler Message

```
Success
```

Input (stdin)

```
saveChangesInTheEditor
```

Expected Output

```
5
```

Time Complexity: O(n)

Learning Outcomes:

i. Applying basic algorithm concept.
ii. Learn String manipulation.
iii. Learn about case insensitivity.
iv. Learn to handle edge cases.
v. Understand basic string processing.