Experiment - 1.4.1

Student Name: Nikhil Kumar **UID: 22BCS15501**

Branch: BE-CSE **Section/Group: 22KPIT-902/B**

Semester: 6th **Date of Performance: 15/01/25**

Subject Name: AP LAB-II **Subject Code: 22CSP-351**

1. **Aim:** In this experiment, students will learn about String Matching, Hashing, and Heap data structures.

   Problem 1.4.1: Rotate String

2. **Objective:**

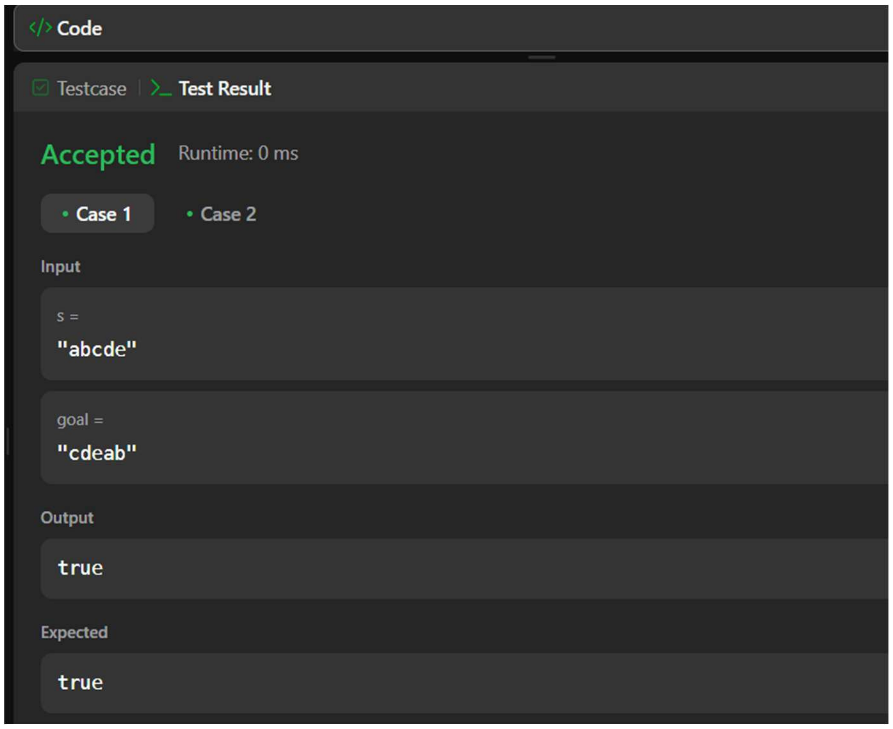
   - **Problem Statement 1:** Given two strings s and goal, return true if and only if s can become goal after some number of shifts on s. A shift on s consists of moving the leftmost character of s to the rightmost position.

3. **Implementation/Code:**

a)

```
class Solution {
public:
    bool rotateString(string s, string goal) {
        if (s.length() != goal.length()) {
            return false;
        }

        string concatenated = s + s;

        return concatenated.find(goal) != string::npos;
    }
};
```

4. **Output**



```
Accepted   Runtime: 0 ms

Input
s =
"abcde"
goal =
"cdeab"

Output
true

Expected
true
```

Time Complexity: O(n)
Space Complexity: O(n)

5. Learning Outcome

- Understand how string concatenation (s + s) can be used to simplify solving problems involving cyclic shifts.
- The code demonstrates the use of string::find to search for a substring within a string. You learn how to efficiently use built-in functions in C++ for string searching.
- Learn how to print boolean values (true or false) as words (true/false) using cout << boolalpha, which is useful when you want more readable outputs instead of 1 or 0.

Experiment-1.4.2

1. **Aim:** In this experiment, students will learn about String Matching, Hashing, and Heap data structures.

   Problem 1.4.2: Find the Index of the First Occurrence in a String

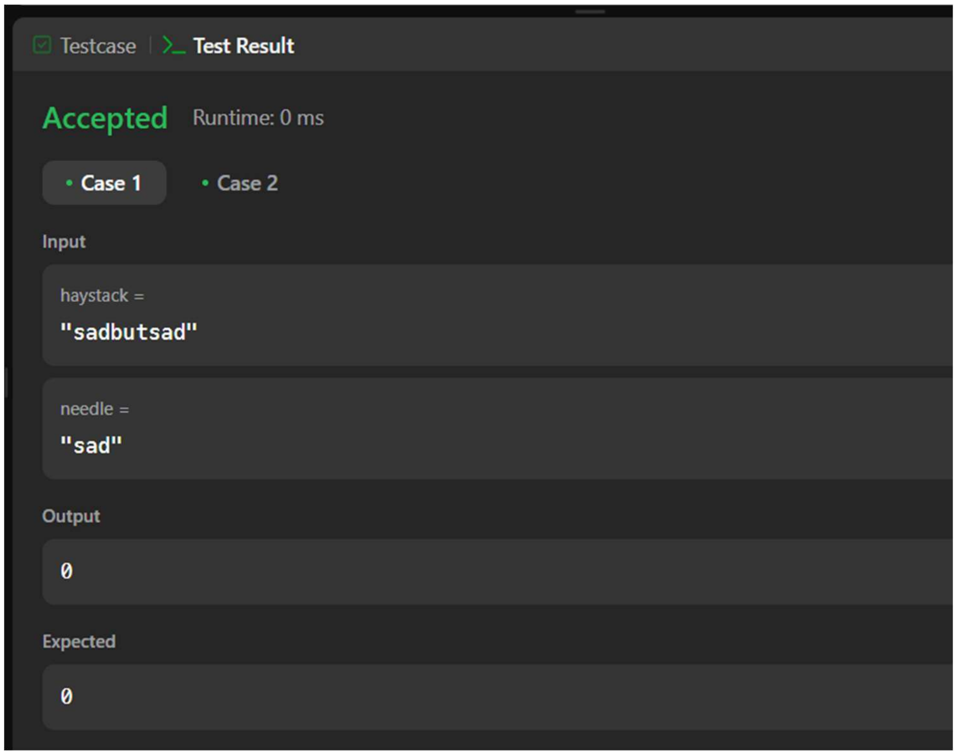
2. **Objective:**

   - **Problem Statement 2:** Given two strings needle and haystack, return the index of the first occurrence of needle in haystack, or -1 if needle is not part of haystack.

3. **Implementation/Code:**

```
class Solution {
public:
    int strStr(string haystack, string needle) {
        int haystackLen = haystack.length();
        int needleLen = needle.length();

        if (needleLen == 0) {
            return 0;
        }

        if (needleLen > haystackLen) {
            return -1;
        }

        for (int i = 0; i <= haystackLen - needleLen; i++) {
            if (haystack.substr(i, needleLen) == needle) {
                return i;
            }
        }

        return -1;
    }
};
```

4. Output


Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

```
haystack =
"sadbutsad"

needle =
"sad"
```

Output

```
0
```

Expected

```
0
```

Time Complexity: O(n*m)
Space Complexity: O(m)

5. Learning Outcome

- Learn how to implement a basic string search algorithm that finds the first occurrence of a substring (needle) in a string (haystack).
- Handling special cases, such as when the needle string is empty, by returning a predefined value (0 in this case).
- Demonstrating clear separation of functionality with the strStr function and the main function, improving readability and modularity.