



## Experiment-4

**Student Name:** Nikhil Niraj Singh

**UID:** 22BCS10133

**Branch:** BE-CSE

**Section/Group:** KPIT-901/B

**Semester:** 6<sup>th</sup>

**Date of Performance:** 18/01/25

**Subject Name:** Advanced Programming Lab - 2

**Subject Code:** 22CSP-351

### 1. Aim:

1. Problem: 1.4.1: Rotate String. 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.
2. Problem: 1.4.2: Find the Index of the First Occurrence in a String. 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*.

### 2. Objective:

1. Problem 1.4.1: Determine if one string can be transformed into another by performing a series of left-to-right rotations.
2. Problem 1.3.2: Find the index of the first occurrence of a substring (*needle*) in a given string (*haystack*) or return -1 if the substring is not found.

### 3. Implementation/Code:

1.)

```
#include <iostream>
```

```
#include <string>
```

```
using namespace std;
```

```
bool rotateString(string s, string goal)
```

```
{
```

```
    if(s.length() != goal.length())
```

```
        return false;
```

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

int main()
{
    string s, goal;

    cout << "Enter the first string (s): ";
    cin >> s;

    cout << "Enter the second string (goal): ";
    cin >> goal;

    if(rotateString(s, goal))
    {
        cout << "Yes, the string s can be rotated to become the string goal." <<
endl;
    }
    else
    {
        cout << "No, the string s cannot be rotated to become the string goal." <<
endl;
    }

    return 0;
}
```

2.)

```
#include <iostream>
```

```
#include <string>
```

```
using namespace std;
```

```
int strStr(string haystack, string needle)
```

```
{
```

```
    if (needle.empty())
```

```
        return 0;
```

```
    for (int i = 0; i <= haystack.size() - needle.size(); i++)
```

```
    {
```

```
        if (haystack.substr(i, needle.size()) == needle)
```

```
        {
```

```
            return i;
```

```
        }
```

```
    }
```

```
    return -1;
```

```
}
```

```
int main()
```

```
{
```

```
    string haystack, needle;
```

```
    cout << "Enter the haystack string: ";
```

```
    cin >> haystack;
```

```
    cout << "Enter the needle string: ";
```

```
    cin >> needle;
```

```
    int index = strStr(haystack, needle);
```

```
    if (index != -1)
```

```
    {
```

```
        cout << "The first occurrence of \"" << needle << "\" in \"" << haystack <<
        "\" is at index: " << index << endl;
    }
    else
    {
        cout << "The substring \"" << needle << "\" is not found in \"" << haystack
        << "\"." << endl;
    }

    return 0;
}
```

## 4. Output:

1.

```
PS D:\class_problem\ap\exp_3> cd "d:\class_problem\ap\exp_4\" ; if ($?) { g++ test1.cpp -o test1 } ; if ($?) { .\test1 }
Enter the first string (s): abcde
Enter the second string (goal): bdcea
No, the string s cannot be rotated to become the string goal.
PS D:\class_problem\ap\exp_4> cd "d:\class_problem\ap\exp_4\" ; if ($?) { g++ test1.cpp -o test1 } ; if ($?) { .\test1 }
Enter the first string (s): abcde
Enter the second string (goal): bcdea
Yes, the string s can be rotated to become the string goal.
PS D:\class_problem\ap\exp_4> █
```

2.

```
PS D:\class_problem> cd "d:\class_problem\ap\exp_4\" ; if ($?) { g++ test2.cpp -o test2 } ; if ($?) { .\test2 }
Enter the haystack string: hello
Enter the needle string: ll
The first occurrence of "ll" in "hello" is at index: 2
PS D:\class_problem\ap\exp_4> █
```

**5. Time Complexity:**

1.  $O(n+m)$
2.  $O(n-m+1)$

**6. Space Complexity:**

1.  $O(n)$
2.  $O(1)$

**7. Learning Outcome:**

1. Understand string manipulations and rotations.
2. Learn how to check for substrings efficiently.
3. Develop problem-solving skills for string-related algorithms.
4. Gain knowledge of substring search techniques.
5. Practice using loops and conditionals for string traversal.
6. Enhance skills in optimizing string operations.