1.4.1. Rotate String

Problem Statement: 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.

Code:

```
class Solution {
public:
   bool rotateString(string s, string goal) {
      if (s.length() != goal.length())
         return false;
      if (s.length() == 0)
         return true;
      string temp = s + s;
      return temp.find(goal) != string::npos;
   }
};
```

1.4.2. Find the Index of the First Occurrence in a String

Problem Statement: 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.

Code:

```
class Solution {
public:
    vector<int> computeLPS(string pattern) {
        int n = pattern.length();
        vector<int> lps(n, 0);
        int len = 0;
        int i = 1;
        while (i < n) {
            if (pattern[i] == pattern[len]) {
                len++;
                lps[i] = len;
                i++;
            }  else {
                if (len != 0) {
                    len = lps[len - 1];
                } else {
                    lps[i] = 0;
                    i++;
                } } }
        return lps;  }
    int strStr(string haystack, string needle) {
        int m = haystack.length();
        int n = needle.length();
        if (n == 0) return 0;
        if (m < n) return -1;
        vector<int> lps = computeLPS(needle);
        int i = 0;
        int j = 0;
        while (i < m) {
            if (needle[j] == haystack[i]) {
                i++;
                j++;
            }
            if (j == n) {
                return i - j;
            }
            else if (i < m && needle[j] != haystack[i]) {
                if (j != 0) {
                    j = lps[j - 1];
                }
                else {
                    i++;
                }
            }
        }
        return -1;
    }
};
```