Experiment – 1.2

Student Name: Avinash Jena **UID: 20BCS2690**

Branch: CSE **Section/Group: 707/B**

Subject Name: Competitive Coding II **Subject Code: 20CSP-351**

1. Aim: Rotate String

2. Objective:

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.

- For example, if s = "abcde", then it will be "bcdea" after one shift.

Example 1:

Input: s = "abcde", goal = "cdeab"

Output: true

3. Code:

```
class Solution {
    public boolean rotateString(String s, String goal) {
        if (s.length() != goal.length()) return false;
        if (s.equals(goal)) return true;
        StringBuilder sb = new StringBuilder();
        for (int i = 1; i < s.length(); i++) {
            sb.append(s.substring(1)).append(s.charAt(0));
            s = sb.toString();
            if (s.equals(goal)) return true;
```

```
sb.setLength(0);
        }
        return false;
    }
}
```

4. Output:

Java Auto

```java
class Solution {
    public boolean rotateString(String s, String goal) {
        if (s.length() != goal.length()) return false;
        if (s.equals(goal)) return true;
        StringBuilder sb = new StringBuilder();
        for (int i = 1; i < s.length(); i++) {
            sb.append(s.substring(1)).append(s.charAt(0));
            s = sb.toString();
            if (s.equals(goal)) return true;
            sb.setLength(0);
        }
        return false;
    }
}
```

Testcase Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

s =
"abcde"

goal =
"cdeab"

Output

true

Expected

true

Console Run Submit

5. Aim: Repeated String Match

6. Objective:

Given two strings a and b, return the minimum number of times you should repeat string a so that string b is a substring of it. If it is impossible for b to be a substring of a after repeating it, return -1.

Notice: string "abc" repeated 0 times is "", repeated 1 time is "abc" and repeated 2 times is "abcabc".

Example 1:

Input: a = "abcd", b = "cdabcdab"
Output: 3
Explanation: We return 3 because by repeating a three times "abcdabcdabcd", b is a substring of it.

7. Code:

```
class Solution {
    public int repeatedStringMatch(String A, String B) {
        if(A==null || B==null)
            return -1;
        int[] pref = longPrefSub(B);
        StringBuilder temp = new StringBuilder();
        int count = 0;
        while(temp.length()<B.length()){
            temp.append(A);
            count++;
        }
        if(isFound(temp.toString(), B, pref))
            return count;
        if(isFound(temp.append(A).toString(), B, pref))
            return count+1;
        return -1;
    }

    private int[] longPrefSub(String word) {
```

```
int[] pattern = new int[word.length()];
        Arrays.fill(pattern, -1);
        int i = 1, j = 0;
        while (i < word.length()) {
            if (word.charAt(i) == word.charAt(j)) {
                pattern[i] = j;
                i++;
                j++;
            } else if (j > 0) {
                j = pattern[j - 1] + 1;
            } else {
                i++;
            }
        }
        return pattern;
    }
    private boolean isFound(String string, String substring, int[] pattern) {
        int i = 0, j = 0;
        while (i + substring.length() - j <= string.length()) {
            if (string.charAt(i) == substring.charAt(j)) {
                if (j == substring.length() - 1) {
                    return true;
                }
                i++;
                j++;
            } else if (j > 0) {
                j = pattern[j - 1] + 1;
            } else {
                i++;
            }
        }
        return false;
    }
}
```

8. Output:

Java Auto

```java
class Solution {
    public int repeatedStringMatch(String A, String B) {
        if(A==null || B==null)
            return -1;
        int[] pref = longPrefSub(B);
        StringBuilder temp = new StringBuilder();
        int count = 0;
        while(temp.length()<B.length()){
            temp.append(A);
            count++;
        }
        if(isFound(temp.toString(), B, pref))
            return count;
        if(isFound(temp.append(A).toString(), B, pref))
            return count+1;
        return -1;
    }

    private int[] longPrefSub(String word) {
        int[] pattern = new int[word.length()];
        Arrays.fill(pattern, -1);
        int i = 1, j = 0;
        while (i < word.length()) {
            if (word.charAt(i) == word.charAt(j)) {
                pattern[i] = j;
                i++;
                j++;
            } else if (j > 0) {
                j = pattern[j - 1] + 1;
            } else {
```

Testcase **Result**

Accepted Runtime: 0 ms

Testcase Result

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

a =
"abcd"

b =
"cdabcdab"

Output

3

Expected

3

Contribute a testcase

Console Run Submit