CS19001: Programming and Data Structures Lab

Lab Test:2 (EVEN-PC) **Section:15** **Date: 29-Oct-2018**

Instructions:

- You have to **submit only two .c program files** (and nothing else) in the mentioned two submission links (in Moodle).
- Please obey the file-naming convention as follows: ***RollNo_MachineNo*****_LT2_Prog1.c** (for **Problem-1**) and ***RollNo_MachineNo*****_LT2_Prog2.c** (for **Problem-2**).
  [Please write your **own** Roll-Number and Machine-Number as mentioned.]
- Submission Deadline: **29-Oct-2018, 12:00 NOON (!! STRICT !!)**

Problem-1: [Matrix-Rotate]

Write a C-program which –

- Takes a (non-zero) positive integer N from the user and dynamically allocates space for an NxN array
- Initializes the array with user inputs. Once the input is provided and the user hits an enter button, print the array **nicely** in a NxN form
- Once the user again hits the enter button (2nd time in total), print the array with 90° **clockwise rotation**. This should happen "in place", i.e. on the original array. You CANNOT define ANY extra array in your program.
- Every time the user hits the enter button, the last printed array is again rotated 90° clockwise and printed. Once the user types "exit", the program terminates.

Execution example:

```
 Input: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 (User hits enter)

1st print → user hits enter(2nd print) → user hits enter(3rd
print)

0   1   2   3            12  8   4  0         15  14  13  12
4   5   6   7            13  9   5  1         11  10  9   8
8   9   10  11           14  10  6  2         7   6   5   4
12  13  14  15           15  11  7  3         3   2   1   0

(user types "exit" and hits enter)→ program terminates
```

Problem-2: [Search-Prefix]

Given a string *x*, any other string *y* is called the *prefix* of *x* if there exists some other string z such that *yz* = *x*. For example, *ab* is a prefix of *ababc*.

Write a C program which –

- Takes as input two integers m > 0 and n>0. Takes as input two strings $s_1$and s_2 of length *m* and *n* and stores them with suitable dynamic memory allocation.
- Reports the number of instances where s_1 is occurring as a **prefix** in some substring of s_2. The program ignores NULL substrings of s_2 and avoids repetitions of the same substring at many places in s_2. (considers unique substrings only)

Example: If the entered strings are $s_1 =$ aba and $s_2 =$ ababac, then

ALL possible unique substrings of ababac are :

a,b,c,
ab,ba,ac,
aba,bab,bac,
abab,baba,abac,
ababa,babac,
ababac.

In the above, s_1 is a prefix of the following strings: **aba,abab,abac,ababa,ababac.**

Hence answer is = **5**