**Batch: C4-1 Roll No.:16010123217**

**Experiment / assignment / tutorial No. 5**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **TITLE:** Write a program in C to demonstrate use of character arrays and strings |

**AIM:**

1. Write a program that searches for a substring within a given string.
2. Write a program to check if one string is the rotation of another.

\_\_\_\_**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Expected OUTCOME of Experiment:**

Apply the concepts of arrays and strings(CO3).

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Books/ Journals/ Websites referred:**

1. Programming in C, second edition, Pradeep Dey and Manas Ghosh, Oxford University Press.
2. Programming in ANSI C, fifth edition, E Balagurusamy, Tata McGraw Hill.
3. Introduction to programming and problem solving , G. Michael Schneider ,Wiley India edition.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Problem Definition:**

1. The program searches for a substring within a given string and returns the starting index if found, or -1 otherwise.

Example:

|  |  |
| --- | --- |
| Test case 1:  Input:  String: Programming  Substring: ing  Output:  Index: 8 | Test case 2:  Input:  String: Programming  Substring: Python  Output:  Index: -1 |

1. The program checks whether a given string is the rotation of the other.

Example:

|  |  |
| --- | --- |
| Test case 1:  Input:  String 1: abcd  String 2: bcda  Output:  Yes | Test case 2:  Input:  String 1: abcd  Substring: dcba  Output:  No |

**Algorithm:**

**1)**

1. Start
2. Declare an array **str** to store the input string with a maximum size of 100 characters.
3. Prompt the user to input a string.
4. Read the string until a newline character is encountered using **scanf** with the format specifier **%[^\n]**, which reads characters until a newline character is encountered and stores them in the **str** array.
5. Declare an array **SubString** to store the input substring with a maximum size of 100 characters.
6. Prompt the user to input a substring.
7. Read the substring until a newline character is encountered using **scanf** with the format specifier **" %[^\n]"**, which reads characters until a newline character is encountered and stores them in the **SubString** array.
8. Find the substring in the string using the **strstr** function, which returns a pointer to the first occurrence of the substring in the string or NULL if the substring is not found.
9. Check if the substring is found:
   * If the substring is not found (**result == NULL**), print "Index: -1".
   * If the substring is found:
     + Calculate the index of the first occurrence of the substring in the string by subtracting the address of the substring from the address of the string (**(result - str)**).
     + Print "Index: " followed by the calculated index.
10. End.

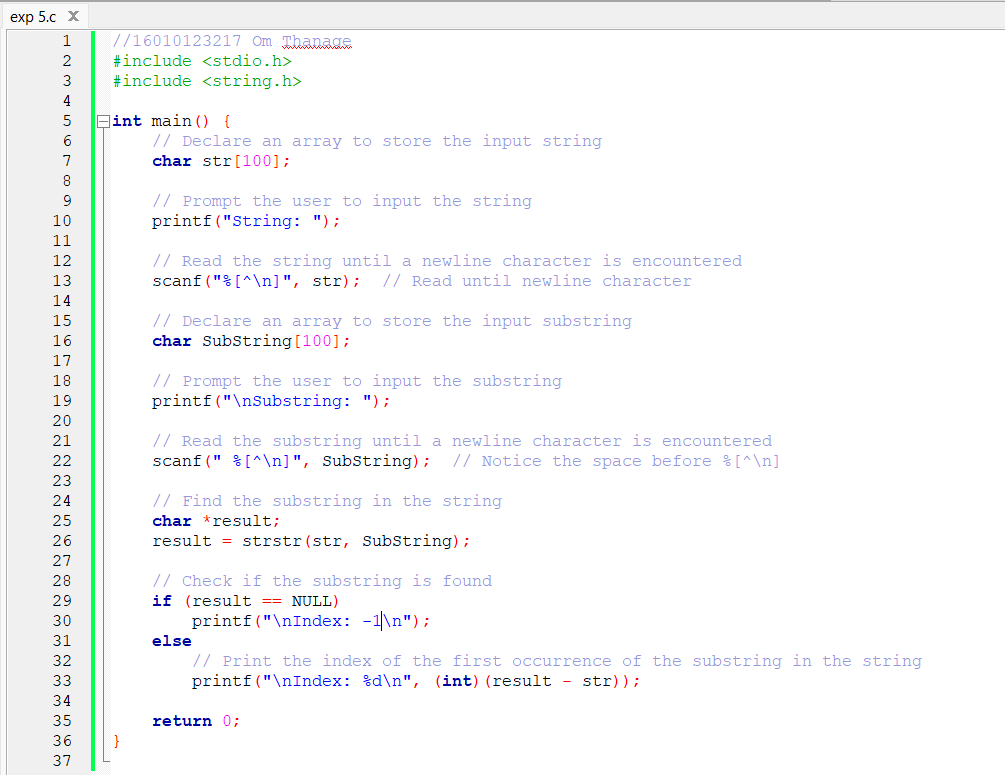
**2)**

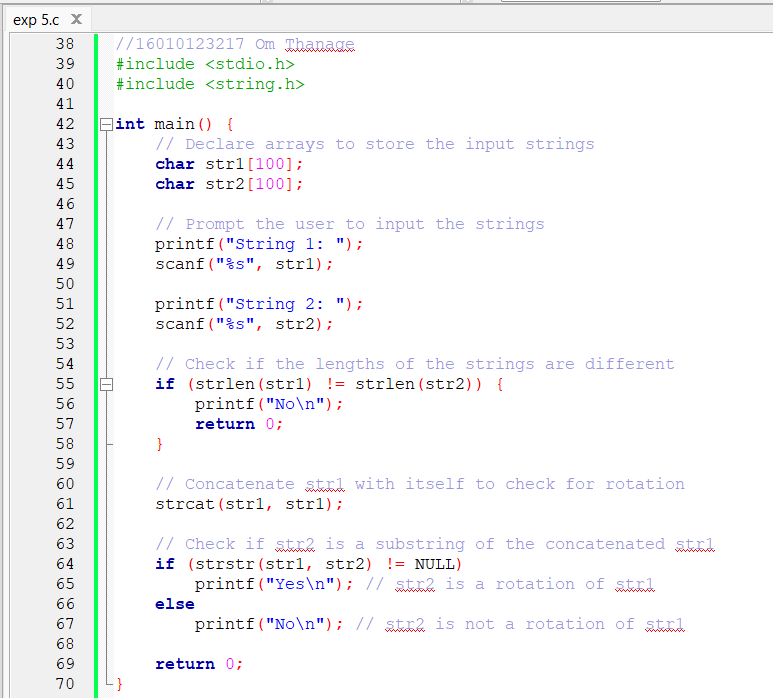
1. Start
2. Declare two arrays **str1** and **str2** to store the input strings with a maximum size of 100 characters each.
3. Prompt the user to input the first string (**str1**).
4. Read the input string (**str1**) using **scanf** with the **%s** format specifier, which reads a sequence of non-whitespace characters and stores them in the **str1** array.
5. Prompt the user to input the second string (**str2**).
6. Read the input string (**str2**) using **scanf** with the **%s** format specifier, which reads a sequence of non-whitespace characters and stores them in the **str2** array.
7. Check if the lengths of **str1** and **str2** are different by comparing their lengths using the **strlen** function:
   * If the lengths are different, print "No" to indicate that **str2** cannot be a rotation of **str1**, and exit the program.
8. Concatenate **str1** with itself to create a new string that contains all possible rotations of **str1**. This is done using the **strcat** function.
9. Check if **str2** is a substring of the concatenated **str1** using the **strstr** function:
   * If **strstr** returns a non-NULL value, it means that **str2** is found within the concatenated **str1**, indicating that **str2** is a rotation of **str1**.
     + Print "Yes" to indicate that **str2** is a rotation of **str1**.
   * If **strstr** returns NULL, it means that **str2** is not found within the concatenated **str1**, indicating that **str2** is not a rotation of **str1**.
     + Print "No" to indicate that **str2** is not a rotation of **str1**.
10. End.

Top of Form

**Implementation details:**

1) Code:

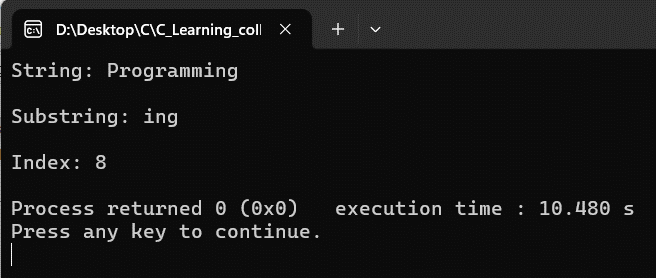


2) Code:

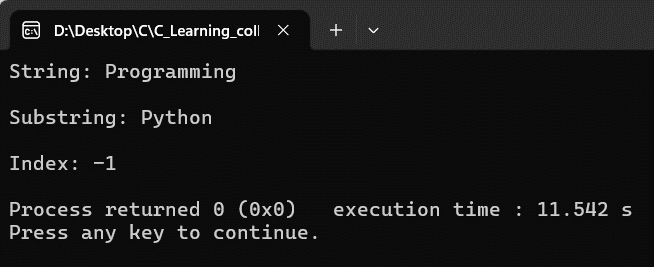
**Output(s):**

**1)**

Test case 1:

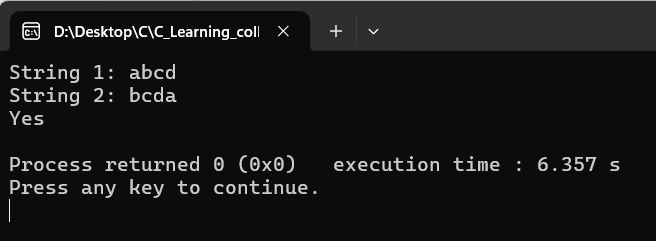


Test case 2:

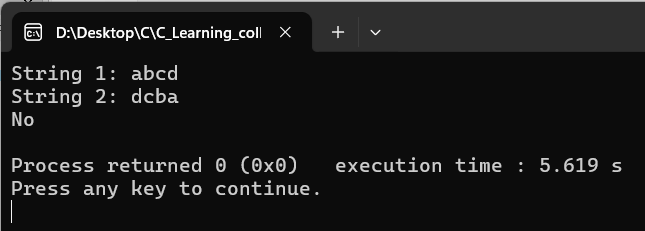


**2)**

Test case 1:



Test case 2:

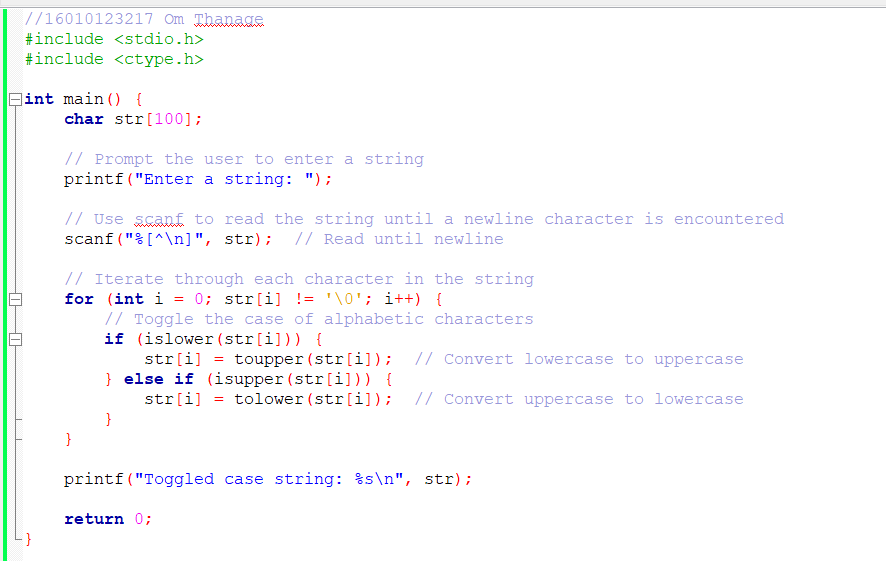


**Conclusion:**

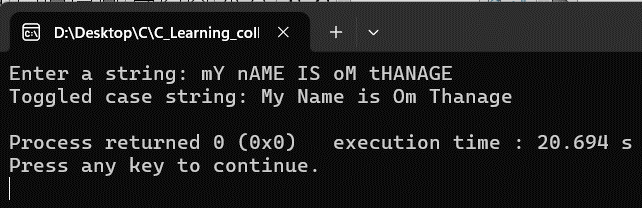
In this experiment I learned the concepts of arrays and strings for checking the if a substring is present in the given string and rotation of the string.

**Post Lab Questions**

1. Write a C program to toggle case of each character in a string i.e. if a character is in uppercase, change it to lower case and vice-versa.



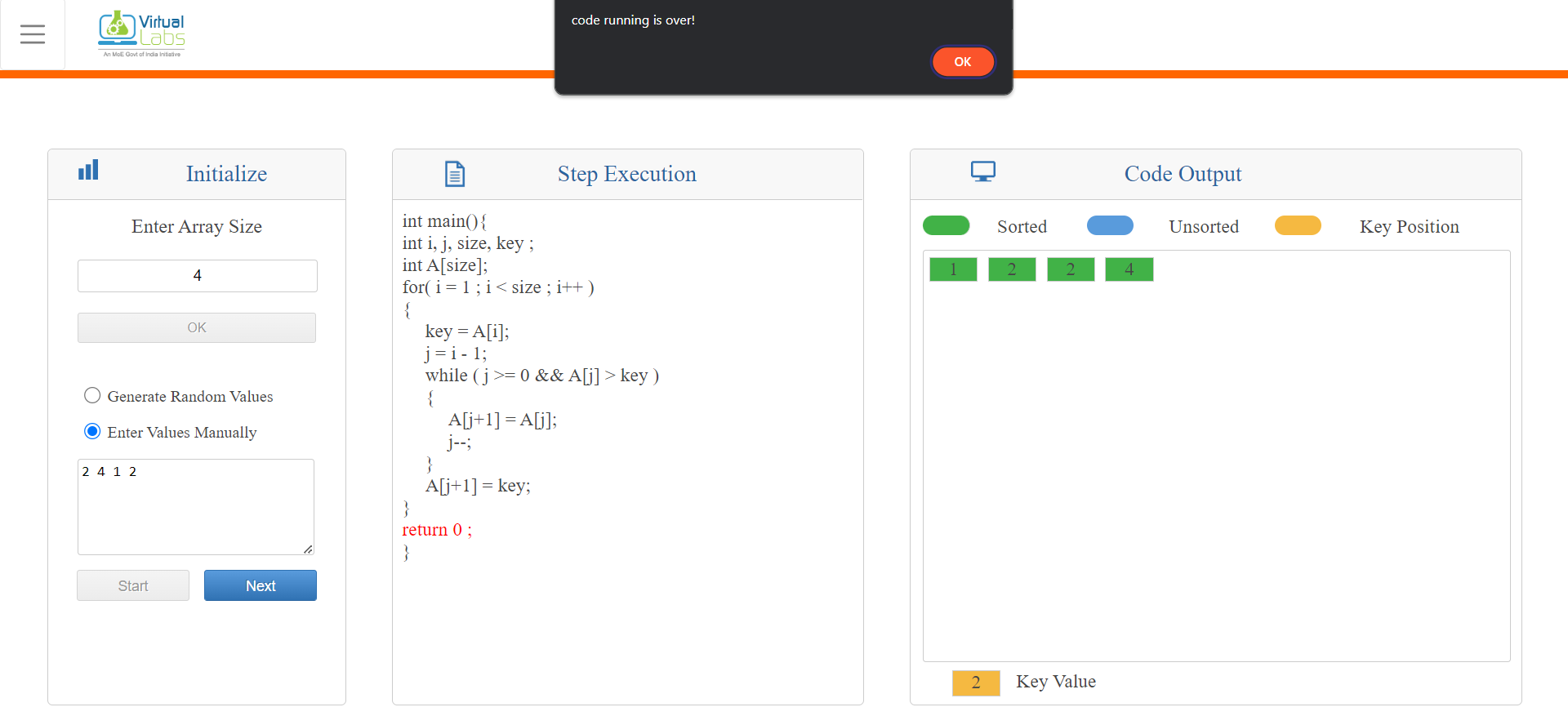
**Output:**



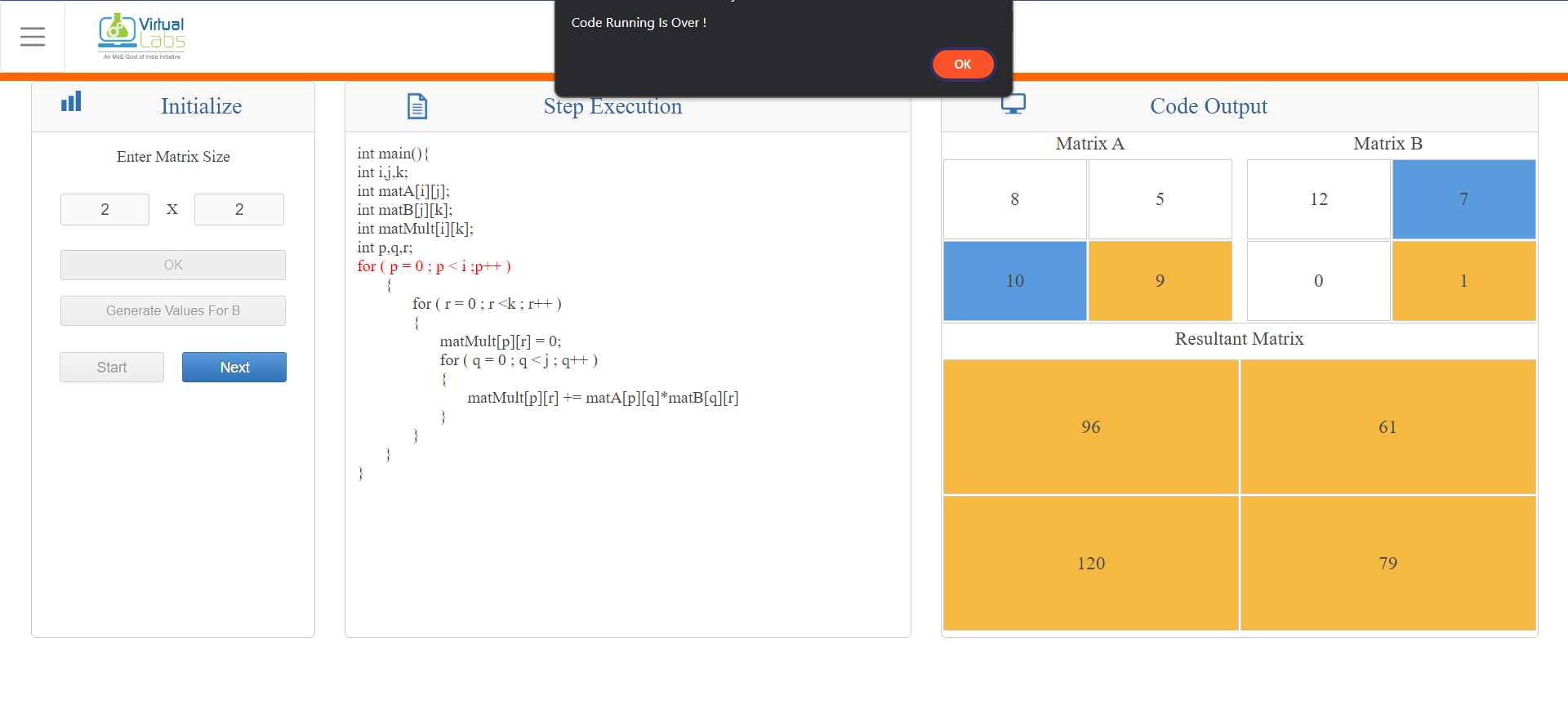
1. Virtual Lab for Arrays

<https://cse02-iiith.vlabs.ac.in/exp/arrays/simulation.html>

1-D array



2-D array



**Date: \_\_\_\_\_\_\_\_\_\_\_\_\_ Signature of faculty in-charge**