# National University of Computer and Emerging Sciences, Lahore Campus

| Course Name: | Programming Fundamentals | Course Code: | CS 118 |
|---|---|---|---|
| Program: | BS(CS) | Semester: | Fall 2020 |
| Duration: | 150 Minutes | Total Points: | 100 |
| Paper Date: | 22-02-2021 | Page(s): | 3 |
| Exam Type: | Lab Final Exam | Sections: | BCS-1A |
| | | | BCS-1B |
| | | | BCS-1F |
| | | | BDS-1B |
| | | | BSE-1A |
| | | | BSE-1B |

Registration No._____

## Instructions:

- Attempt all questions
- Submit ONLY .cpp File in this format (Create a .cpp and rename it with your Roll Number e.g., L20_1234_Q1.cpp). Put all .cpp files in a folder. Rename the folder with your Roll No. L20_1234.
- Submit the folder on \\cactus\Xeon\Fall 2020\Asad Ullah\Final Exam\Slot 1\BXX-1Y, where XX denotes your degree program and Y represents your section.
- Submit both questions on Google Classroom in your respective sections..
- Plagiarism will result in an F grade in the lab.
- No cell phones are allowed. Sharing of USBs or any other items is not allowed.

---

## Problem No. 1: Longest Common Prefix                                    [Marks: 20]

*Write a C++ function to print "Longest Common Prefix using Character by Character Matching". The function takes a 2D char array and the number of rows as input parameters. The function must print the longest common prefix on the console.*

You will use the following given piece of code to test the function. You are unallowed to make any change in the main.

```cpp
#include<iostream>
using namespace std;

const int rows = 5, cols = 30;
void LongestCommonPrefix(char arr[][cols], int rows);

int main() {

    char arr[rows][cols] = { "CharacterLess", "Character ", "CharArray",
"CharteredAccountant" };

    LongestCommonPrefix(arr, rows);

    return 0;

}
```

**Example No. 1**

Input : {"CharacterLess ", "Character ", "CharArray", "CharteredAccountant"}
Output : "Char"

Explanation

In the above example, we have found maximum alphabets that appear at the beginning of each word. Also, consider the following scenarios:

    (a) If we add "Champion" to the above list. The output will be "Cha". As the common alphabets at the beginning of each word in the above list are "Cha".

    (b) If we add "Overcharge" to the above list, it will output nothing. As we must compare alphabets that appear at the start of each word.

**Example No. 2**

Input : {"apple", "ape", "april"}
Output : "ap"

Explanation

The first two alphabets in each of the word are the same while the difference occurs in the third alphabet. That's why the output is "ap".

## Problem No. 2: Kth Smallest

**[Marks: 30]**

Write a C++ program to determine the kth smallest element in an array. Declare an array of size 10 and initialize it with - 1. You need to take input in an array until the user enters - 1 or the user has entered 10 values. When the input in the array is terminated, the system will tell how many valid entries are stored in the array and will print the array after sorting.

Now, the user enters a number $k$ where $k$ is smaller than the **count of elements entered by the** user. You need to find the $k^{th}$ smallest element in the given array.

    The elements of the array may repeat.

    **Sample behaviour of application:**

| | |
|---|---|
| Input: | Enter max=10 values or - 1 to stop: 6 10 4 3 20 15 4 – 1 |
| Output: | Your valid entries are: **7** <br> sorted array: **3, 4, 4, 6, 10, 15, 20** |
| Input: | Now enter k: **3** |
| Output: | The Kth Smallest Value is: **6** |

## Problem No. 3: Lexicographically minimum string rotation

**[Marks: 50]**

The order in which items are arranged in a conventional dictionary; alphabetical order is known as Lexicographical order. Given two different words of the same length, say a = $a_1 a_2...a_k$ and b = $b_1 b_2...b_k$, the order of the two words depends on the alphabetic order of the symbols in the first place "i" where the two words differ (counting from the beginning of the words): a < b if and only if $a_i < b_i$ in the underlying order of the alphabet A.

For instance, consider two strings A = "abced" and B = "abcde". The first three characters are similar; hence, they don't any information on the lexicographic minimum. If we compare $a_4$ and b, we found that $b_4 < a_4$; therefore, B < A. In other words, B is lexicographically smaller than A.

Write a C++ program to find the lexicographic minimum. For example, for the array BCABDADAB, the lexicographic minimum is ABBCABDAD.

Given the array BCABDADAB, we have to generate the following combinations using left rotation:

BCABDADAB
CABDADABB
ABDADABBC
BDADABBCA
DADABBCAB
ADABBCABD
DABBCABDA
ABBCABDAD
BBCABDADA

Now, if we sort it in the Lexicographical order, we obtain:

ABBCABDAD
ABDADABBC
ADABBCABD
BBCABDADA
BCABDADAB
BDADABBCA
CABDADABB
DABBCABDA
DADABBCAB

The lexicographic minimum is  ABBCABDAD.

## More Examples:

| Input:  | GEEKSQUIZ |
| Output: | EEKSQUIZG |

| Input:  | GFG |
| Output: | FGG |

| Input:  | GEEKSFORGEEKS |
| Output: | EEKSFORGEEKSG |

You must create following functions:

```
void rotateleft(char str[]);
```
// let's say str = abcd, the rotated str will be, str = bcda

```
void sort2DAlphabetically(char arr[][50], int rows);
```
//This function sorts a 2D char array in ascending alphabetical order.

Following is a simple solution.

1. Create a char array of size 50, name as 'str'.
2. Take input in 'str' from the user.
3. Compute the length of str, say it 'strLength'
4. Now, create a 2D char array (named as arr) of size (rows = 50, cols = 50) to store all rotations of 'str'. [Although the rows and cols are 50 but the actual rows and cols which will be consumed, will be equal to the 'strLength']
5. Now do the following task:
   ```
   for i = 0 to strLength
           rotateleft(str)
           strcpy(arr[i], str)
   ```
6. Now Sort arr:
   ```
   sort2DAlphabetically(arr, strLength);
   ```

7. Print **arr[0]**

Note: You are unallowed to use built-in functions except the strcpy.