North South University
CSE115, Assignment 4, Spring 2021, Due: May 18th 2021.

Pointer:

1)Implement the following function which finds the largest element in an array (use pointer)
int max(int *p, int size);

2)Implement the following function which finds the length of a string using pointer operation.
int length(char *s);

Structure:

3) Create a structure called **Patient** with the following members: Registration_Number, Name, Age, Gender, Contact Number, Address, Physician_name [(3): all 3 names need not be filled initially], Corona_status (had corona in the past or not). Assume that there will not be more than **50 patients** in the hospital on a single day. Since people who had corona in the past might have complications, so in order to visit a physician, each patient need to inform whether he/she had corona in the past or not. **Populate** the array with information of **n** number of patients (the number **n** is provided by user). You may write a menu driven program or you may make function calls from main (). [**Use of global variable is not recommended for this question**].

a) Write a function to **display** names and registration number of all **patients** (who had corona in the past) in the **n** number of patients in the array.
b) Write a function to **append** patient information at the end of the array. [You should have condition to check array bounds]
c) Write a function to **delete** patient information **given a name by the user.** [Consider all names are unique].
d) Write a function to **modify** contact number member and physician_name member of a record **given a registration number of the patient**.

4). Create a structure called **Player** with the following members. struct Player { char name[20]; int age; char country[20]; char Position[20]; double Salary; double Rating; }; First, create an array of Player structures. Now, write a function that takes an array of Player structure as input and find out the highest paid player among all the players.

void highestPaidPlayer(struct Player *pl, int size);

NB: You should send the first address of the array in your function call and the pointer **pl** in the function header will accept it.

Recursion:

5) Write a program in C to calculate the power of any number using recursion

File:

6)Create a file named countline.txt. Insert the lines:
test line 1
test line 2
test line 3
test line 4
Write a program in C to display the content of the file and number of lines in a Text File. [Hint: Consider that the lines in your file has fixed length and your buffer size is big enough to accommodate each line]
Test Data: Input the file name to be opened: countline.txt
Expected Output: The content of the file test.txt are:
test line 1
test line 2
test line 3
test line 4
The lines in the file are: 4

7)Write a program in C to count a number of words and characters in a file.
Test Data: Input the file name to be opened: test.txt
Expected Output: The content of the file test.txt are: test line 1 test line 2 test line 3 test line 4
The number of words in the file test.txt are: 12
The number of characters in the file test.txt are: 36

8) typedef struct
```
{ char name[50];
 int id;
char dept[20];
double cgpa;
} student;
```
Write a function void saveByDept(char *fileName, char *deptName , student allStudents[], int size) that will save the information of all students who belong to the department with name pointed to by deptName into a text file. The name of the file is given as an input parameter, fileName

Two-Dimensional Array:

9) Take input of a 3x3 matrix and display the sum of its main diagonal element. For the following matrix, your program should display 12. (Because 5+3+4 = 12)

$$\begin{matrix} \mathbf{5} & 2 & 1 \\ 0 & \mathbf{3} & 7 \\ 6 & 8 & \mathbf{4} \end{matrix}$$

10) Write a program in C to accept two matrices and check whether they are equal.

Test Data:

```
Input Row number and Column number of the 1st matrix: 2   2
Input Row number and Column number of the 2nd matrix :  2   2
Input elements in the first matrix:
element - [0][0] : 1
element - [0][1] : 2
 element - [1][0] : 3
element - [1][1] : 4
Input elements in the second matrix :
element - [0][0] : 1
element - [0][1] : 2
 element - [1][0] :
3 element - [1][1] : 4
```

Expected Output :

```
The first matrix is: 1 2
                     3 4
The second matrix is: 1 2
                      3 4
The Matrices can be compared: Two matrices are equal.
```