BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI (RAJ.)

Second Semester 2019-20 CS F111 Computer Programming

LABORATORY SESSION #12

(Recursion, Command line arguments, file pointers)

1. Write a C program that asks the user to enter the name (string) and roll no (integer) of N no. of students and stores the information in a structure variable. It writes these information in the following format in a file named as *output.txt*. The file was initially opened for writing. The file is then closed. It is opened again and fgets is used to read and then display the content of the file till the end of file.

Sample I/O:

I/P:

```
Enter the roll no: 10
Enter name of the student: Aayush
Enter the roll no: 11
Enter name of the student: Deepa
```

O/P:

```
Roll Number of the student: 10
Name of the student: Aayush
Roll Number of the student: 11
Name of the student: Deepa
```

2. Write a program that takes two integers N1, N2 and a task t as command line arguments and performs the corresponding task (as given below) on N1 and N2. You may use *if else* or *switch case* or any combination of them. There is no need to use functions. You may use any number of additional variables in your program. If the number of parameters passed is not 4 (including ./a.out), an error message should be printed. You may assume that the user will always enter in the order N1, N2, t. No need to check if the order is correct.

Task (t)	Output
0	Exit
1	Swap the values of N1 and N2 and print them
2	Compute N1 + N2 and print the sum

Sample I/O:

1) Input: ./a.out 1 2 1
   Output: Value of N1 : 2 Value of N2 : 1

2) Input: ./a.out 1 2
   Output: Number of arguments should be 4.

3) Input: ./a.out 1 2 0
   Output: <Nothing will be printed… Just EXIT>

3. Write a C program that first reads the number of elements n to be kept in an integer array A (Maximum size 20), and then reads n distinct elements in sorted order. It also reads an integer x as input and then checks for the presence or absence of x in the array A.

a. Write a recursive function linear_search that linearly checks each element (i.e., checks the elements one by one). It also stores the number of times the function is called in a global variable count_lin_search.
b. Write a second recursive function ternary_search that does the following:

1. The array is broken up into three parts of roughly equal length.
2. If x is equal to the largest element of the first part, it returns that array location. If x is smaller than the largest element of the first part, it recursively searches inside the first part. Similar logic is followed for recursively searching within the second and third parts.
3. It stores the number of times the function is called in a second global variable called count_ternary_search.

c. The main function should have minimal code. It reads the input and calls the above two functions one by one and prints the array location returned by each function if x exists in A. Otherwise, it prints "Given element does not exist in the array" for each call. It also prints the number of times the individual functions are called. You may choose suitable parameters for your recursive functions.

Sample I/O:
1.

```
A = 3 7 12 40 84 91
x = 12

Linear Search
12 is at position 2
Function is called 3 times.

Ternary Search
12 is at position 2
Function is called 4 times
```

2..

```
A = 3 7 12 40 84 91 121 167 189 378 569 678
x = 190

Linear Search
190 is not present
Function is called 8 times.

Ternary Search
190 is not present
Function is called 5 times.
```