CSCI-UA.0201

Computer Systems Organization

Programming Assignment #1

The total score of this assignment is 50 points.

In this first programming assignment you will write C code. You have five C assignment to solve all included in the same C file: lab1.c
This means all your work will be inside this C file: lab1.c.

The five assignments are described below. But before you start coding, please read the following list of bullets first.

- Read ALL the comments on the C file before you write anything.
- Before writing code, trying to come up with a good algorithm on a piece of paper, as we said in class.
- It will help you a lot if you read and understand the main() function.
- Include your name and NetID in the corresponding comment at the top of the file.
- Do not change any code in the file where it states so. So do not change function declarations or the main function.
- However, you are free to add extra functions if you want.
- The first step is to download the file lab1.tar from the course website. Then, from your CIMS account, type: **tar -xvf lab1.tar** You will find a new directory called lab1 created after the execution of the previous command. Inside that directory will find all what you need. Continue reading this document first.

You compile your code as:

gcc -Wall -std=c99 -o lab1 lab1.c

You then execute the lab by typing:

./lab1 x [input]

Where:
x can take the value of 1 to 5, corresponding to each one of the assignments below. [input] the argument of the corresponding assignment if any. Some assignments do not need arguments.

To help you, you are provided with an executable file lab1ref that you can run and see its output for each one of the five assignment. You can consider this file as the model answer. Your executable must produce the same output.
To start working on each assignment, you can type:
./lab1ref x
where x takes the value from 1 to 5.

When you press enter, you will get a message that shows you the extra arguments you have to give for each assignment. For example, for assignment 1, you need to provide 3 numbers, which makes the command line: ./lab1ref 1 num1 num2 num3
and so one.

Here are the descriptions of the assignments. You will also find more info in the comments on top of each function the C file.

1. f1 [5 points]:

That function f1 takes three numbers start, end, and incr. They are all positive nonzero and start <= end.
You must print on the screen: start start+incr start+2incr … end
For example, if the command line is:
./lab1 1 2 10 3
The first number indicates assignment for f1, stat = 2, end = 10, and incr = 3.
The output on the screen is:
2 5 8 10
Note that "end" must be printed even if it is not start+(x*incr). In the example above, for example, 10 cannot be presented as 2 +(something*3) yet we have to print 10.

2. f2 [10 points]:

In this function, you are presented with a file. That file contains only characters and spaces (i.e. no numbers). You need to open that file, read its contents, and store the content into another file with the same name as the original with ".reverse" added to the filename. That new file contains the same characters as the old one but with lower case characters transformed to upper case, and vice versa. Spaces in the original file must be left intact.
For instance, if the input file is called: info.txt and contains:
Aa BB cc   d
When you type:
./lab1 2 info.txt
f2 will generate a new file with the name info.txt.reverse that contains:
aA bb CC   D

You will find a test file in the directory lab1 called testf2.in that you can use to test your code. But yoy can easily create other test cases.

3. f3 [15 points]:

This function has as input a filename and a number. The file contains 9 integers. Each integer can be either 1 or 0. The second argument is a number of generations that can be a positive number.
The numbers in the file form a 3x3 matrix.
So, if the file contains: 1 1 1 0 1 0 1 0 1

This means the matrix is:
1 1 1
0 1 0
1 0 1
We will call this generation 0. Each item in the matrix, we can call it cell, has neighbors, on top, bottom, left, right, and diagonals. The number of neighbors differs depending on the position of the cell. For example, the cell at position (0,0), which is the top left, has only three neighbors. While the middle cell, at (2, 2) has 8 neighbors.
For each cell, count the neighbors with value "1". If that number is 2 or 3, this cell becomes 1 (or stays 1 is it is already 1) in the next generation. Otherwise, the cell becomes 0, or stays 0 if it is already 0, in the next generation.
For example, the next generation of the above matrix will be:
1 1 1
0 0 0
0 1 0
Finally, after you are done with all number of generations, print the matrix on the screen.

You will find a test file in the directory lab1 called testf3.in that you can use to test your code. But yoy can easily create other test cases.

4. f4 [15 points]:

This function is given an array of n integers and must print them in descending order, followed by a new line.
So, if the input array has: 9 1 2 7
It must print: 9 7 2 1

5. f5 [5 points]:

This function accepts two integers as arguments and prints on the screen all the non-prime numbers between these two numbers, including the two numbers themselves if they are non-prime, followed by a new line after the last number.
So, if the two numbers are 1 and 10, you must print:
1 2 4 6 8 9 10

Final notes:

- The main philosophy of this programming assignment is to get you to try several things. You have to read a code and understand it (the code of the main function and the function declarations). You have to implement several functions that touch upon many concepts of C.
- Please, write as many comments as possible with your code so that we can give you partial credit in case your code has something wrong.
- You must not alter the declarations of the functions originally in lab1.c but feel free to add any extra functions you want.

- You must strictly follow the output format or else you will lose points. Your output must be exactly as lab1ref.
- You just need to submit your lab1.c

And ….

HAVE FUN!