EECS2031ON Winter 2021 | Programming Exam 1 | Due: Sat, March 20th 23:59

In the first programming exam, we are going to assess your ability for the following skills:

- Ability to solve non-trival problems using basic algorithms and data structures
- Ability to develop and test programs written in the C programming language
- Ability to describe the memory management model through use of two-dimensional arrays or pointers.
- Basic coding style

Coding environment:

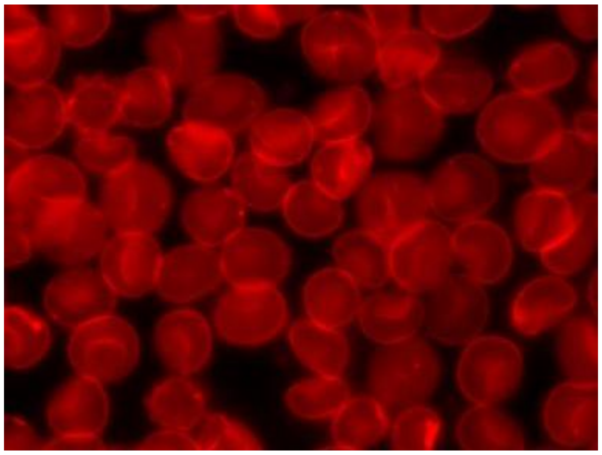
Similar to the weekly labs, you can work by connecting to the lab, or, work locally. Make sure the final delivable will comiple in the lab environment.

Before you start, I would like to remind you that this assignment is different from the weekly labs that you have done, as it accounts as an "exam" that you take home.

For weekly labs you cannot share your solutions with peers but you can discuss ideas. For this assignment, however, **you can neither share your solution with peers nor get help from them**. Please have a look at the statement at the top of `cellCount.c` before you start doing the assignment. When you are finished you need to sign this statement by writing your information (full name and student number) in the gap provided. Please be aware that your assignment WILL NOT be marked if you do not sign the declaration; and that your code will be checked by both a plagiarism detection tool and us to make sure your solution to this problem is unique.

Problem Description:

In order to count a number of blood cells in a sample, a blood cell slide is created and looked at through a microscope's lens. To get a clear understanding of what a blood cell slide looks like, please see the picture below that was borrowed from Wikimedia (https://commons.wikimedia.org/wiki/File:Sedimented_red_blood_cells.jpg).



We were able to write a code the reads this image and convert it to a two-dimensional array, where the blood cells in red are represented with 1 and the rest is represented with zero. Then we got a small sample of this two-dimensional array for which we need to write a code that counts the number of blood cells. A blood cell is a group of 1s that are connected horizontally, vertically, or diagonally. As shown in the left pictures below, where there are 7 blood cells.

Task 1:

For this task, you are going to complete two functions. One to label each blood cell (how is it defined?) with a unique number and another to count the number of the unique blood cells.

1. The first function gets a 10x10 two-dimensional array of 0s and 1s, and labels each blood cell with a unique number. For example, if the given image (i.e., two-dimensional array) is something like the left picture below, one sample solution can be something like the right picture. It is not important how you choose the numbers for the cell, as long as it is not zero or a negative number. Also, the numbers do not need to be in a consecutive order. For example, one might label the cells in the example below with arbitrary numbers of 20, 3, 4, 8, 18, 7, 1. The signature of this function looks like below:

```
    void color(int image[IMAGE_SIZE][IMAGE_SIZE]);
```

This function gets the 10x10 two-dimensional array (like the left picture) and label each group of 1s (a distinct cell) with a unique number (like the right picture).

2. Another function gets a 10x10 two-dimensional array that has been labelled, like in right picuture below, and return the number of distinct blood cells. This array represents each unique cell with a unique number. Please see the right picture below in which you can see

there are 7 distinct cells, each represented with a unique number. If the right picture below is sent to this function, it should return 7, as there are 7 distinct groups of data. The signature of this function is as follow:

```
int cellCount(int image[IMAGE_SIZE][IMAGE_SIZE]);
```

```
0  0  1  1  0  0  1  0  0  1                 0  0  2  2  0  0  3  0  0  4
1  0  1  1  0  0  1  1  0  1                 8  0  2  2  0  0  3  3  0  4
1  0  0  1  1  0  1  1  0  1                 8  0  0  2  2  0  3  3  0  4
1  1  0  0  0  0  0  0  0  0                 8  8  0  0  0  0  0  0  0  0
1  0  0  1  1  1  0  0  1  0                 8  0  0  5  5  5  0  0  5  0
0  0  0  0  1  0  0  1  1  0                 0  0  0  0  5  0  0  5  5  0
0  0  1  0  0  1  0  1  0  0                 0  0  6  0  0  5  0  5  0  0
0  0  1  1  0  0  1  0  0  0                 0  0  6  6  0  0  5  0  0  0
0  0  1  0  0  0  0  0  1  1                 0  0  6  0  0  0  0  0  7  7
0  1  1  0  0  0  1  1  1  1                 0  6  6  0  0  0  7  7  7  7
```

To solve the problems, you are free to define as many helper functions as needed. To test your code, we only look at the return value and the content of the array, expecting that each blood cell is labeled with a unique number.

Marking Scheme: [60 points out of 80]

- 30 points for the correctness of `color()` function.
- 10 points for the correctness of `cellCount()` function.
- 20 points for code style. Please note that even if you do not get any points for the execution of the code, you may still be eligible to receive these 20 points. Any complex structure in your code should come with a comment explaining what that part of the code is supposed to do. Correct Indentation and using meaningful variable and function names are also important.

Task2:

For this task, you redesign the funciton using pointer notation, instead of array index notation.

Task 2.1:

For this task you are asked to re-implement function `color()`, where the input parameter is defined as a pointer. For this, you're required to use pointer arithmetic instead of using array indices to get access to the elements of the array. The function signature is:

```
void colorPtr(int* image);
```

Note that, if you use helper functions for this task, then all the helper functions that take an 2D array as argument should be defined as taking `int * image` as parameter. And, in the helpfer functions you should also use pointer notation, no array index notation should be used throguhout the solutions in accessing the array element. (It is okay to use [] <u>only</u> when you <u>declare</u> an extra array in the functions).

Task 2.2:

For this task you are asked to re-implement function `cellCount()`, where the input parameter is defined as a pointer. For this, you're required to use pointer arithmetic instead of using array indices to get access to the elements of the array. The function signature is:

```
    int cellCountPtr(int * image);
```

Note that, if you use helper functions for this task, then all the helper functions that take an 2D array as argument should be defined as taking `int * image` as parameter. And, all helper functions should also use pointer notation, no array index notation should be used throguhout the solutions in accessing array elements. (It is okay to use [] <u>only</u> when you <u>declare</u> extra arrays.)

Marking Scheme: [20 points out of 80]

- 15 points for the correctness of `colorPtr()` function.
- 5 points for the correctness of `cellCountPtr()` function.

Bonus question [20 extra points]

Redesign the `color()` function and implement it using recursion.

The function signature is (still) :

```
    int colorRecursively(int image[IMAGE_SIZE][IMAGE_SIZE])
```

but you can define recursive helper functions to perform the task.

Sample output

If implemeted correctly, both versions of `color()` should label the array with 7 unique lables and both versions of `countColor` will output 7. An example is given below (the labels can be any unique numbers).

```
------ Image Contents -------

00, 00, 20, 20, 00, 00, 21, 00, 00, 22,

23, 00, 20, 20, 00, 00, 21, 21, 00, 22,

23, 00, 00, 20, 20, 00, 21, 21, 00, 22,

23, 23, 00, 00, 00, 00, 00, 00, 00, 00,

23, 00, 00, 25, 25, 25, 00, 00, 25, 00,

00, 00, 00, 00, 25, 00, 00, 25, 25, 00,

00, 00, 26, 00, 00, 25, 00, 25, 00, 00,

00, 00, 26, 26, 00, 00, 25, 00, 00, 00,

00, 00, 26, 00, 00, 00, 00, 00, 28, 28,

00, 26, 26, 00, 00, 00, 28, 28, 28, 28,

-----------------------------

Total number of cells in the image: 7
```

Note that the input 2D array argument for the functions can be a 10x10 array with any shape and number of blood cells. Your function should correctly label and count the cells in the arrays. You may want to modify the given 2D array to test your solutions. (But the submitted code should be the original main code.)

Note:

Your program should compile in the lab environment. When compiling the program, you may see a couple of warnings from the proivided functions and your function, such as the follows.

```
warning: initialization from incompatible pointer type [enabled by default]
     int* ptr = cellImg_;
                ^
cellCount.c:143:5: warning: passing argument 1 of ‘printImgArray’ from incompatible
pointer type [enabled by default]
     printImgArray(ptr);
```

You should ignore those warnings. It is fine as long as there are not errors and a.out is generated.

What to submit, where to submit:

You submit one file called cellCount.C, using submit using command

```
submit 2031ON PE1 cellCount.c
```

or, use websubmit at https://webapp.eecs.yorku.ca/submit/

- Login using you eecs user name
  - If you keep on getting passport York login page when following the link, then clear the browsing history, cache, and cookie of your browser, and then try the link again.
- Select **Course: 2031ON**, and then select **Assginment: PE1**