**Lab # 08 Functions and Pointers in C**

**Pre-Lab Reading 1:**

**Addresses of variables and Pointers in C**

When a C programmer says that a certain variable is a “pointer”, what does that mean? It is hard to see how a variable can point to something, or in a certain direction. It is hard to get a grip on pointers just by listening to programmer’s jargon. In our discussion of C pointers, therefore, we will try to avoid this difficulty by explaining pointers in terms of programming concepts we already understand. The first thing we want to do is explain the rationale of C’s pointer notation.

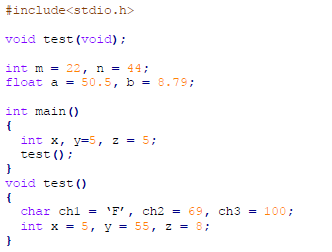
**Pre-Lab Reading 2:**

**Returning multiple values from a function**

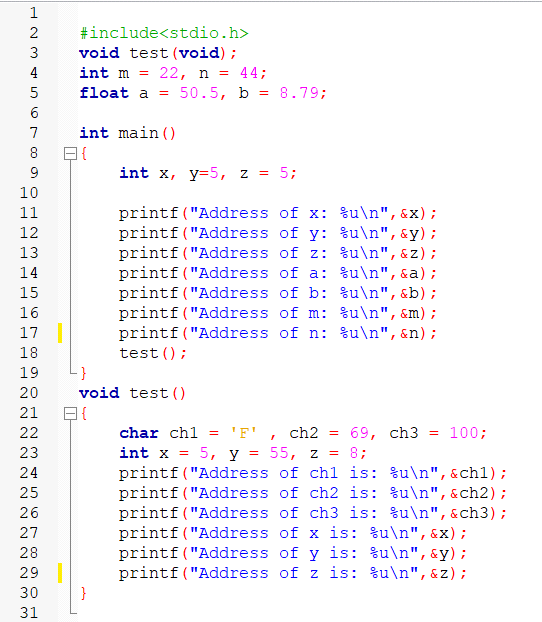
In C, we cannot return multiple values from a function directly. In this section, we will see how to use some trick to return more than one value from a function.

**In-Lab Task 1:**

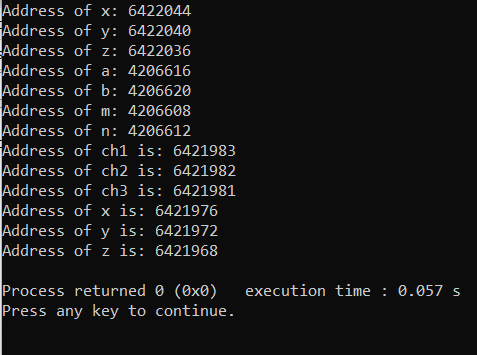
For the given program find (and print) the addresses assigned to all the variables against their names.



**Program code:**



**Output:**



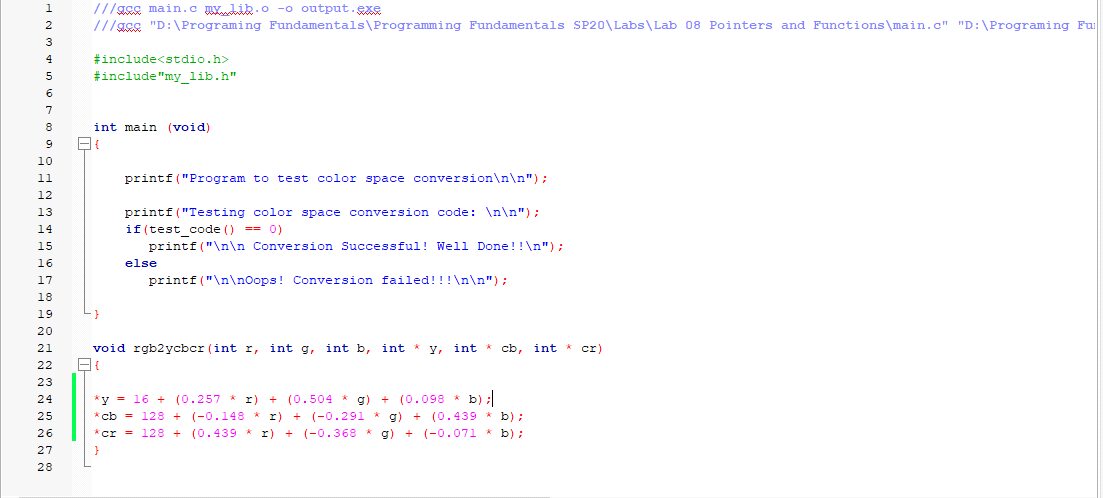
**In-Lab Task 2:**

**Color Space Conversion**

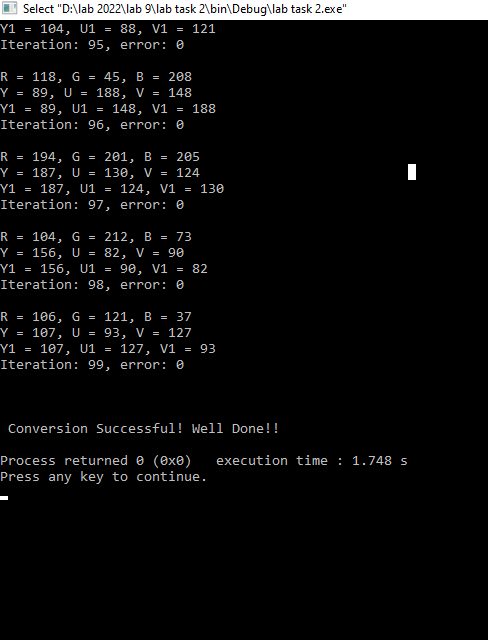
Write a C function which takes three integer inputs, corresponding to RGB components of a colored pixel, and coverts them to the YUV color space. The function should have the following prototype.

void rgb2ycbcr(int r, int g, int b, float \* y, float \* Cr, float \* Cb);

**Program Code:**



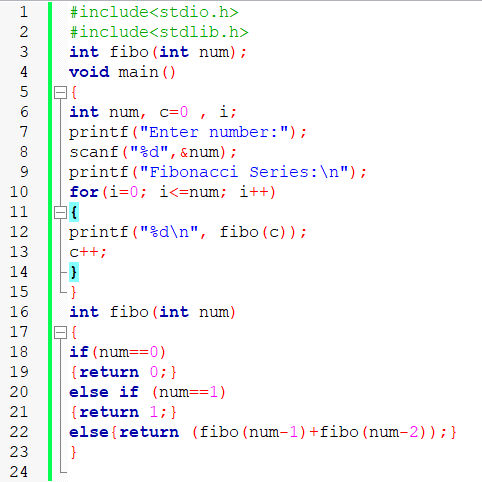
**Output:**



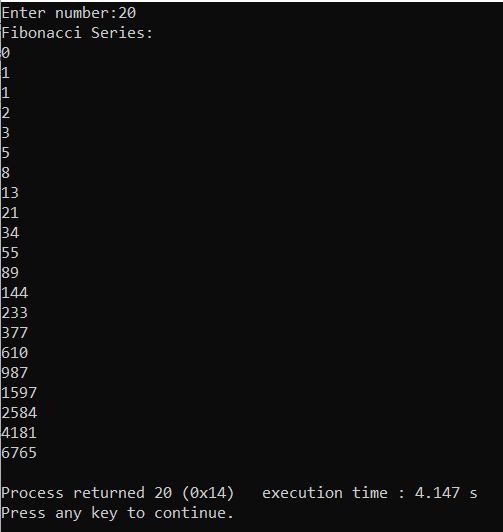
**Post Lab Task: Recursion**

Write a recursive version for the Fibonacci function developed in class.

**Program code:**



**Output:**



**Critical Analysics:**