Overview

In this lab we will be going over how to write, compile, and run C files. First we will write some basic C code with macros and get it running correctly. Then we will inspect how macros really work. **<u>You should do all of this in your VM, not windows or OSX.</u>**

Writing, compiling and running your own C code

1. Open your favorite text editor and create a file named “hello_world.c”. In this file define a main function and make it print “hello world” or your favorite alternative. If you don’t already know how to do this from lecture I encourage using Google or asking the people around you.
2. Save this file
3. Compile the file:
    a. Navigate to the directory you saved your file in
    b. Type “gcc hello_world.c -o hello_world” and hit enter
        i. “gcc” is the name of the compiler we are using
        ii. “hello_world.c” is the C file we want to compile
        iii. “-o hello_world” tells the compiler what we want to name the new file
            1. NOTE: we could just have easily created a file called “this_lab_is_boring.exe” by running “gcc hello_world.c -o this_lab_is_boring.exe” there is nothing special about this name
    c. If there are errors in your code fix them
4. You should now have a file named “hello_world” (no .c suffix)
5. Run the compiled file:
    a. Type “./hello_world” into your terminal and hit enter
6. You should now see your program run.

Making a slightly more useful file

1. Create a new C file and name it “recursion_is_easy.c”
2. Create another main function (or copy it from your hello_world file)
3. Create a recursive function called “factorial” that has 1 integer parameter “n” and returns the factorial of n.
    a. HINT: C reads the file top to bottom so if you define a function after it is used C will complain. To get around this either define factorial above main (hacky) or create a function prototype at the top of the file (recommended).
4. Modify your main function to **print the result of factorial for 1,2,5,7** in decimal (base 10) and hex (base 16)
    a. HINT: the function “printf” allows you to format your output (example below)

```
int years_old = 900;
printf(“When %d years old you reach, look as good you will not”, years_old);
// %d is for decimal (base 10)
// %x is for hex (base 16)
// %s is for strings
// if you are curious see the printf documentation for more options
```

5. Test your code by compiling and running again

Adding in Macros

Let’s clean up our current file by making a macro named “FACT_FORMAT” that defines the string we are going to print without the values (it should just equal the current string you are formatting). Now your print statements should look something like

```
#define FACT_FORMAT “fact(%d) == %d == 0x%x”

printf(FACT_FORMAT, 1, factorial(1), factorial(1))
...
printf(FACT_FORMAT, 7, factorial(7), factorial(7))
```

Compile and run your code

Inspecting Macros

Macros are actually just a find-replace tool built into the C-preprocessor. To examine what is going in we are going to utilize a gcc flag that will only run the preprocessor.

```
gcc recursion_is_easy.c -o recursion_is_easy --save_temps
```

This “--save-temps” option will let us see all the intermediate steps the compiler takes.

1. **Open the created .i file** (recursion_is_easy.i)
2. If you used #include to include stdio.h (in order to print) then you will see a bunch of confusing lines at the top of your file, **scroll all the way down to the bottom of your file where your code is.**
3. Look specifically where your macro used to be used, what is different?
4. #define will actually just find every place you use the macro and replace it with the macro’s definition.

The grand finale

Now that you have some experience writing, compiling, and running C files you have to make a program that converts a person's name into their equivalent Star Wars name. This program should take in four parameters:

1. First name
2. Last name
3. Mother's maiden name
4. City of birth

It will then combine these into the person's Star Wars name with the following algorithm.

First name	First 3 letters of your first name + first 2 letters from your last name
Last name	First 2 letters from your mother's maiden name + first 3 letters of the city you were born in

For an example we can use my newest kitten:

1. First name : **Alp**honse
2. Last name : **Pe**teet
3. Mother's maiden name: **Lo**omis
4. City of birth: **Atl**anta

This would result in his name being Alppe Loatl.

Deliverables

Show a TA your "recursion_is_easy" file running as well as the output from your Star Wars name generator.