In order to get credit for the lab, you need to be checked off by the end of lab.

This lab is NOT group work. It is your individual work!!!

(2 pts) Conditional Compilation/Include Safeguards: One of the useful features of the preprocessor is to conditionally include code. For instance, this is often referred to as a DEBUG macro:

```c
#ifdef DEBUG
    /* your debug code here */
#endif

    /* your non-debug code here*/
```

For this task, ensure that all of your print statements are wrapped in DEBUG macros. You can compile with a `–D DEBUG` to define it or leave it out!!!

```
g++ prog.cpp -D DEBUG
```

Another way we use conditional compilation is with header safeguards. At the top of every header file, you should include the following to avoid including multiple headers.

```c
#ifndef CREATURE_H
    #define CREATURE_H

    /* your header/interface code here */
#endif
```

(2 pts) Begin Assignment 3 Inheritance:
Inheritance in OOP creates an “is a” relationship. In your assignment 3, you have “is a” and “has a” relationships. Determine the relationship among the World, Creature, Demon, Human, Elf, Balrog, and Cyberdemon classes.

To create the “is a” relationship in code is having a base class and having other classes inherit from this base class. For example, class creature {
    protected:
        int shared_var;
    public:
        creature();
};
class human : public creature {
    int human_var;
    public:
        human();
};

In this example, your human has the variable human_var, as well as the shared_var variable from the creature. Your creature only has access to the shared_var, and you can create a creature or a human object in this model, depending on your needs. In this example, a human “is a” creature and inherits features from the creature.

Begin designing and writing the template for your classes depending on the “has a” and “is a” relationships. In addition, you can begin writing accessor and mutator functions for each class. Make sure you put your safeguards in your header files!!!

(6 pts) Practice Demo – (1 hour)

1. Your TA will provide you a guideline and two questions. Pick one question, and return the other back to the TA.

2. Next, you will open a terminal, and maximize the window!!! (TAs will make sure you have putty maximized)

3. Create a directory called test, mkdir test.

4. Now, change into the test directory and create your .h and .cpp files to solve the problem.

5. You are allow to compile and run your program until you complete the exercise. You will automatically lose 3 pts:
   
   If you need to ask the TA how to do something or look at your notes/internet. You will automatically lose 6 pts, if your solution is not complete (or EXTREMELY close to completion). Use your notes, rather than not finishing!!!

6. Run and show your code to a TA to receive your 3, 6, or 10 points for this lab, otherwise your grade will be a zero and cannot change!!!

7. Remove the test directory, cd .. rmdir test