HW Submission requirements:

1) Put your name and ID number on the top of EACH assignment (in comments for programs)
2) Name each file the name written in blue above each problem
3) Put all the files into a folder and zip it
3) Name the zipped folder with all assignments HW#.zip (-5 points for incorrect name)

NOTES:

1. **INCLUDE A README FOR EACH PROGRAM *(-10 points) Remember a README should let the user (my TA) know how to run your program***
2. **MAKE SURE YOUR PROGRAM IS PROPERLY INDENTED. *(-10 points)***
3. **MAKE SURE ANY COMMENTS YOU INCLUDE ARE MEANINGFUL *(-10 points) Do not just put random comments on every line of code***
4. **DO NOT CODE EVERYTHING IN MAIN *(Automatic 0)***

NOTE: For the programs, part of the grading rubric will be putting functionality in the classes. For example, if you make classes but do not put functionality (or very little functionality), points will be deducted. Do not just make a class to make a class and then put all the work in main-the work should be contained in functions in the classes. Example: if you have a program where a person orders from a restaurant, you could have a Person class and a Restaurant class. The functionality of actually ordering can be a function in the Person class NOT something that happens in the main.

Problem 1 (20 points)-True/False Submit a document called Answers.doc

*Answer the following true/false questions. You must correctly state **WHY** your answer is true or false in order to receive credit.*

Code fragment 1:

```
Info_box::Info_box(std::string first_name, std::string last_name)
 {
        set_title("--Registered Person--");
        set_size_request(150, 100);
        add(vbox);

        label.set_text("First name: "+first_name);
        label.set_padding(10,10);
        vbox.pack_start(label);

        label1.set_text("Last name: "+last_name);
        label1.set_padding(10,10);
        vbox.pack_start(label1);
```

```
        ok_button.set_label("Ok");

        ok_button.signal_pressed().connect(sigc::mem_fun(*this,&Info_box::ok_function));

        vbox.pack_start(ok_button);
        show_all_children();
}
```

Code fragment 2:

```
template <class T>
T plus_sign(T n1, T n2)
{
        T ret;

        ret = n1 + n2;
        return ret;
}
```

1. *set_text()* and *set_padding()* are in the same class.
2. *Code fragment 1* is defining a constructor.
3. We can say that the class *Info_box* has at least 4 widgets declared.
4. If *Info_box* is inherited from the *Window* class, we can say for sure that the function *add()* is kept in the *Window* class.
5. *Info_box box1;* would be a valid line of code.
6. *add()* and *show_all_children()* are in the same class.
7. As soon as we create an *Info_box* object, we will have a GUI pop up with a button, first name and last name (in that order-stacked on top of each other).
8. *T* can be any type except a *string*.
9. *T* can be any type except a *boolean*.
10. If *n1* was an integer and *n2* was a string, the function would not work.

Problem 2 (80 points)-Write a program. Submit a folder named Choice that contains the main function (in a file called choice.cpp) and any header files you include. Do not forget to include a makefile. AUTOMATIC 0 IF YOU CODE EVERYTHING IN choice.cpp)

Recreate either of the following programs from HW3 using a GUI:

1) program 2 (chili)
   or
2) program 3 (chipotle)

This means the movement throughout the program should be through the GUI ONLY (not text based). Don't forget to mention in your README which program you are creating. Include any files you read into your program.