The purpose of this homework assignment is just to welcome you to the class and make sure that you have the tools properly set up for future assignments.

Join Piazza

Piazza is the class forum. Joining the forum is not mandatory, however it is in your best interest. Besides using Piazza to ask homework or project questions, the TAs will announce office hour changes and cancelations on Piazza. It's up to you to check the forum regularly for these announcements. However, **please don't post code on the forum**.

While you're there, upload a photo of yourself too, make sure it's one where your face is clearly visible, not your favorite cartoon or video game character. There are a lot of you in this class so it's going to be challenging to remember everyone's names – help us try to get to know you a little better.

Installing the development tools

If you haven't already done it yet, go to the Resources page on the class website and follow the posted instructions to install the C++ compiler and Code::Blocks IDE on your computer. Then come back to this assignment and continue with the instructions below.

Creating a new project in Code::Blocks

Each programming assignment that you work on in this class, whether it is a small homework assignment or one of the larger projects, should exist in a separate **project** in Code::Blocks. This keeps your work well organized and ensures that work that you do on one assignment does not interfere with work on another assignment.

In order to create a new project, make sure Code::Blocks is running and follow the instructions below:

1. Choose the **File > New > Project...** menu option.
2. A wizard will appear asking you what kind of project you want to create. Select **Console Application.**
3. Click the **Go** button, then on the next screen click the **Next** button.
4. When asked what language you'd like to use select **C++** and then click **Next**.
5. For **Project title**, enter a meaningful name for the project – in this case something like "Homework1" would be appropriate.
6. For **Project Folder**, choose a location on your hard drive where you want Code::Blocks to store this project. Once you have done this for the first time, Code::Blocks will remember your decision and use this directory as the default location.
7. You should be able leave other options unchanged and continue through the rest of the wizard. At the end of the wizard clicking the **Finish** button will create the project.

Once the project is created, you will see it appear in the **Projects** window that is docked to the left of the Code::Blocks IDE. Expand the **Sources** folder and you will see a file named "main.cpp". The ".cpp" extension indicates that the file contains C++ source code. `main.cpp` is where you will write all of your code.

Writing your first program

Now you're going to write a very simple C++ program so that you can get yourself accustomed to the tools you'll be using in class. We'll be discussing all of the concepts below in lecture, but it's certainly not a bad thing if you want to go ahead and get started!

1. Double-click the "main.cpp" file to open it in the source code editor. You will see that Code::Blocks has already placed some placeholder code in the file.
2. Normally, it's fine to use this code as a starting point, but since we are learning to program this time we'll **delete everything in the file** so that you can start from scratch.
3. We need to use the **#include** keyword to include "header files" that contain features that we want to use in our programs. In this case, we need to include the iostream header, which lets us read and write text output. So, type the following as the first line of the file:

   ```
   #include <iostream>
   ```

4. After that line, enter the following:

   ```
   using namespace std;
   ```

   As a general rule, you'll want to include this line at the top of all of your programs, just under your **#include**s. The features that you included above are contained in a "namespace" named std. You can think of namespaces as "folders" that contain functions and data types; for example, std::cout refers to something named cout inside the std namespace. By writing **using namespace** std; you can write shorter code by leaving off the std:: prefix.

5. Every C++ program must have a **function** named `main`. The `main` function is the starting point of your program – the code that will be executed whenever you run it. Add this code to your source file:

   ```
   int main()
   {
        return 0;
   }
   ```

   This function is the smallest valid C++ program that you can write. What does it do? Nothing!

   The **int** before the function's name is its **return type**; the main function must return an integer that tells the operating system whether the program ran successfully or not. In our case, we return 0, which means everything went ok. Our programs in this class will never return anything other than 0.

   The parentheses after the name of the function represent **parameters (input)** that the function takes. Parameters allow us to pass values into functions. In this case, since the parentheses are empty, we're saying that the main function does not take any parameters. It optionally can, but we'll never use them.

Lastly, the curly braces after the function are where we place the code that belongs to that function. As we'll see throughout the course, we can separate the logic of our programs into multiple functions that keep the responsibilities of different parts of our programs organized.

6. Now let's have this program do something. We'll use the `cout` stream to print some text out to the screen. Type the following code just before the line with the **return** statement:

   ```
   cout << "Hello world!" << endl;
   ```

   The << symbol is an **operator** that we use to write data to an output stream; in this case, `cout`, which represents the screen. Other output streams can represent files on your disk. Lastly, the `endl` symbol says that the cursor should be moved down to the next line.

7. Now test your program by compiling and running it. To compile your project (also called "building" it), save the changes you've made to "main.cpp" and then click the "Build and Run" button in the toolbar. The button looks like a yellow gear with a green play button next to it. You will see a few lines of cryptic-looking output that reflect the C++ compiler being executed to build your code. If you've done everything correctly a window will pop up and your program will run. If you have errors (look for red text), make sure that what you've typed in looks like this, and try building again:

   ```
   #include <iostream>

   using namespace std;

   int main()
   {
        cout << "Hello world!" << endl;
        return 0;

   }
   ```

8. If your build was successful, After a few seconds, the output window should look something like this:

   ```
   Hello world!

   Process returned 0 (0x0)   execution time : 0.004 s
   ```

   If you see anything different (except for the time), then make sure you entered the program exactly as it was shown above and try it again. Congratulations! You've just finished your first C++ program.

What to Submit

For this assignment you should submit your `main.cpp` file from inside of your project directory.

This assignment will be graded automatically. Test your programs thoroughly before submitting them. Make sure that your programs produce correct results for every logically valid test case you can think of. Do not waste submissions on untested code, or on code that does not compile with the supplied code from the course website.

Web-CAT will assign a score based on runtime testing of your submission; your best score will be counted; the TAs will later verify that your best submission meets the stated restrictions, and assess penalties if not.

To submit this assignment:

1. Visit http://web-cat.cs.vt.edu in your web browser.
2. Enter your Virginia Tech PID and password in the appropriate fields on the log-in screen, and make sure that **Virginia Tech** is selected as the institution. Click **Login**.
3. The Web-CAT home screen will display useful announcements and assignments that are currently accepting submissions. Find the assignment that you want to submit in the table, and click the "Submit" button next to it.
4. Click the **Browse...** button and select the file you want to upload. The homework assignments and programming projects for this course should be self-contained in a single **main.cpp** file, so you can simply select that one file.
5. Click the **Upload Submission** button. The next page will ask you to review your selection to ensure that you have chosen the right file. If everything looks correct, click **Confirm**.

The next page will show that your assignment is currently queued for grading, with an estimated wait time. This page will refresh itself automatically, and when grading is complete you will be taken to a page with your results.

Pledge

Each of your program submissions must be pledged to conform to the Honor Code requirements for this course. Specifically, you **must** include the following pledge statement in the submitted file:

```
//    On my honor:
//
//    - I have not discussed the C++ language code in my program with
//      anyone other than my instructor or the teaching assistants
//      assigned to this course.
//
//    - I have not used C++ language code obtained from another student,
//      or any other unauthorized source, either modified or unmodified.
//
//    - If any C++ language code or documentation used in my program
//      was obtained from an allowed source, such as a text book or course
//      notes, that has been clearly noted with a proper citation in
//      the comments of my program.
//
//    <Student Name>
```

Failure to include this pledge in a submission will result in the submission being disallowed during code review.