1. Learn to use an editor. It is suggested that you learn emacs. Although other editors (such as pico and gedit) may be simpler to learn, emacs provides numerous features that make a programmer’s life easier.

You can learn emacs by going through the built-in tutorial.

(a) If you’re using the X-windows (graphical) interface, you can fire up emacs in one of two ways:
   - You can launch a Terminal window. Having done so, you now enter the command “emacs &” into a terminal window. (Note the ampersand!)
   - You can launch Emacs directly from the graphical interface.

Once you’ve launched emacs, you can find its tutorial the Help menu in the emacs menu bar.

(b) If you’re using a text-based interface, you should enter the command “emacs” into your terminal window. (Unlike the previous case, there is no ampersand here!) Within emacs, you should type “C-_ t” (that’s control-underscore, followed by a “t”; no space should appear between them). This is not “C-_ C-t”; that is, don’t hold the control button down when you type the “t”!

2. Create a directory that will be dedicated to your work on this project. The shell command

   mkdir -p ~/private/cs1/proj0

will do this. (This will be explained further in class.) Make sure that any work you do is restricted to this directory. One way to ensure this is by issuing the shell command

   cd ~/private/cs1/proj0

before doing any work on this project. (For instance, you would do this before running emacs.)

3. Write and run a program that prints “Hello, world!”. The text of such a program is found on the last page of this document. You are to create this program in a computer file proj0.cc, compile it, and run it. Of course, you should use your name, rather than mine, along with the proper date!

   (a) Your first step in creating such a program is to put yourself in the working directory that you created in step 2 above, and then fire up emacs, telling it to edit the file proj0.cc. In other words, you’ll fire up a Terminal window, and issue the commands

      cd ~/private/cs1/proj0
      emacs proj0.cc &

   inside said window.

   (b) After you’ve written (and saved) a correct version of proj0.cc, you can compile it from within emacs. Assuming that you’ve followed directions so far, you should be within the proj0.cc buffer, which should be in C++-mode (look at the modeline at the bottom of the emacs buffer). Give the command “C-c C-c”, which should bring up the prompt “Compile command:” in the minibuffer (underneath the modeline), as well as its expected default response (which is “make -k”). Delete the “make -k”, replacing it with

      g++ -o proj0 -std=c++11 proj0.cc

---

1The details of launching these things involve playing around a bit within the graphical interface. I’ll demonstrate this in class.
2Here, the _ is a blank. Do not put blanks anywhere else in these lines!
If there are any compilation errors, you can go to the offending line by using “C-c C-e”; you can then fix the error and recompile as before.

(c) To test your program, go over to the Terminal window you created in step 3(a) above, and type the command

```plaintext
proj0
```

It should produce the output

```
Hello, world!
```
on a line by itself.

(d) Once your program is running correctly, use the photo program so you can have something to turn in. This command is run within a Terminal window; make sure that `~/private/cs1/proj0` is the working directory. Do not do this until the program is finished! The commands you should issue are as follows:

```plaintext
photo
cat proj0.cc
g++ -o proj0 -std=c++11 proj0.cc
proj0
exit
```

This will create a file, named `typescript`, in the current directory, containing a listing of `proj0.cc`, and a sample execution. You should “clean” the `typescript` by running the command

```plaintext
photo-clean < typescript > proj0.out
```
(The first “<” is optional.)

(e) You can submit a program in one of several ways.

- You can print it out on the printer in Room 812. You should not do this unless you are working in Room 812! If `proj0.out` is short (one page), use the command

  ```plaintext
  lpr proj0.out
  ```

  If `lpr` complains, you should use `enscript proj0.out`

  If `proj0.out` is longer than one page, use the command

  ```plaintext
  enscript -2r proj0.out
  ```

- If you’re working at home, you can transfer the `proj0.out` back to your home computer, and then print it out locally. How this is done will depend on what computer you have at home.

- You can mail it to me, using the command

  ```plaintext
  mail -s "Project 0" -r bovik@cs.cmu.edu agw < proj0.out
  ```

  Note that the `-r` flag specifies a return address. (You are suppose to substitute your preferred email address for bovik@cs.cmu.edu.) This will make it easier for me to send you an acknowledgment message.

  **Most emphatically:** Do not use Harry Q. Bovik’s email address; use your own!

**Note:** The online help document (found at [http://www.dsm.fordham.edu/help](http://www.dsm.fordham.edu/help)) gives more details about submitting a programming assignment. In particular, it will steer you around possible problems. Read it!!

The morbidly curious might want to look at [http://www.roseler-ac.de/wolfram/hello.html](http://www.roseler-ac.de/wolfram/hello.html), which is a collection of HelloWorld programs in hundreds of different computer languages. It also gives you an idea of the variety of programming languages out there.
Here’s the “Hello, world” program. Once again, the _ symbol represents a blank.

```cpp
#include <bjarne/std_lib_facilities.h>

int main()
{
    cout << "Hello, world!\n";

    return 0;
}
```