Reading Assignment Guidelines

The Purpose of a Reading Assignment Even though this is a programming course, you will have to complete several reading assignments. The objective of reading assignments is for you to read, understand, and express the meaning of a program that was written by someone else. Reasons why reading code is an important programming skill include:

- Learning the syntax of the language elements in a classroom setting is not enough to learn how these elements are used. You need to see the language in use in order to understand how those elements can be applied in practice.
- Reading someone else's code will help you appreciate how the structure of the program and its documentation affect its readability. You should make sure that when you write code it is as easy as possible to read.
- Most software is developed by more than one programmer, which means that programmers working on the team need to read each others' code.
- In order to understand code you will sometimes need to learn elements of the language that you are not familiar with. No programming course will ever teach you everything that there is to know about a programming language. You should know where and how to look up things that you do not know about. Reading code extends your knowledge of the language.

Structure of the Assignments For each reading assignment you will be given a short C++ program, probably lacking comments. Most likely, you will not be able to understand everything on the first reading: it may contain language elements that are new, or implement a complicated algorithm that needs to be analyzed step by step. These programs will have structure and documentation that may help in reading or that may make it harder to read and understand.

Your Task. Your goal is to go through the program and figure out what it is doing. I would like you to record the steps that you take in understanding the program. There is no single right or wrong way of writing your response.

- I am interested in your reaction to the program:
  - Is it well written? Is it easy to understand what it is doing? or is it confusing?
  - How much of the program were you able to understand on the first reading?
  - Was your understanding on the first reading the same (or at least similar) to your final understanding? or did it change completely?
- I also want to know which language elements were new to you and how you went about learning about them.
  - Which resources did you use to learn about unfamiliar features of the language? List any websites, man pages, books that you used. If you talked to an instructor (including me) or a friend about it, list these people and how they helped.

  - Are there still some elements that you do not understand? or that you think that you might not understand completely?

- Finally, I want you to describe briefly what the program does and how it does it. It shouldn't be a detailed description of every line, but rather a few sentence long summary with an explanation of what algorithms are used.

Grading. Your response will be graded on a four level scale:

zero	nothing was submitted
one	unacceptable; the response does not show that much work was done; it does not address adequately the questions raised in the previous part; it does not show that the writer understands the program;
two	acceptable, but lacking in some ways; the response is mostly complete, but lacking some parts;
three	acceptable; the response addresses adequately the questions raised in the previous part; it shows writers understanding of the program.

Deliverables. Your response can be in any format you like: bullet points, paragraphs, anything that gets your message across. It should be typed and double-spaced. I expect to see one half to a full page, but feel free to write more.

You should hand in a printed version of your response in class on the day that it is due. If for some reason you cannot be in class on that day, you should email a pdf file to me before the class starts. **I will not accept any format other than printed copy or pdf. I will not accept any work past the deadline.**

Hints. There are many ways to understand the program. You can compile it, build it and run it to see what it is doing. But this may not help in understanding how the program's task is accomplished and/or what language elements are used in the program. Studying the program structure and design should give you more insight into how things are done.

You will need to do some research to learn about some language features that are used in the programs. There are many resources to learn about the language. Your textbook might be a good start. Other books and reference materials for C++ can also be used. There are of course many online resources. Some pages that you might find helpful are:

- http://www.cplusplus.com/
- http://www.learncpp.com/
- http://www.sourcepole.com/sources/programming/cpp/cppqref.html
- http://www.google.com/

Finally, the `man` pages on any Linux/Unix system are a good source. Type, for example, `man sqrt` to learn what `sqrt` does and how it is used. Note: If you want to use `man` pages to look up the header files, you need to use the C header files, not C++ header files. For example, type `man math.h`, not `man cmath`.

Example. Here is a very small example of a C++ program and a possible response to it.

```
#include <iostream>
using namespace std;

int main()
{
    int sum = 0;
    for (int i = 0; i < 100; i++)
    {
        sum += i;
    }
    cout << "The answer is " << sum << endl;
    return 0;
}
```

The program computes the sum of numbers from 0 to 99 (I ran it to make sure). It uses a `for` loop to iterate over the numbers that are added. This way the upper and lower limits of the sum can be changed easily for computations of sums of numbers from other ranges.

I saw the line `using namespace std;` in programs that we wrote in 127, but I never really understood why it has to be there. I know that if I remove it, the program will not compile anymore, so it must be important. I found in the textbook on page 28 that this line is necessary to use `cout`, `cin` and `cerr` (I am not sure why we need both `cout` and `cerr` since they both print to the screen). I was still not clear about what `namespace` is, so I looked at http://www.cplusplus.com/doc/tutorial/namespaces/. I think I found more than I was looking for ;). It seems that namespaces group various commands together, but I do not know where they are. Do I need to create the namespace before I can use commands? I hope we can discuss this in class.

I was not sure what += does. According to the book (page 14) it is just a shortcut notation for addition followed by assignment. I do not like to use shortcuts. I don't think I can remember which symbol goes first. I tried to run the program with =+ instead of += and it does something different, but I do not know why. I will probably stay away from the shortcuts.

The program could use some comments to explain what it is doing, even though for a short program like this, it is easy to figure out what happens. The structure of the code makes it easy to read.