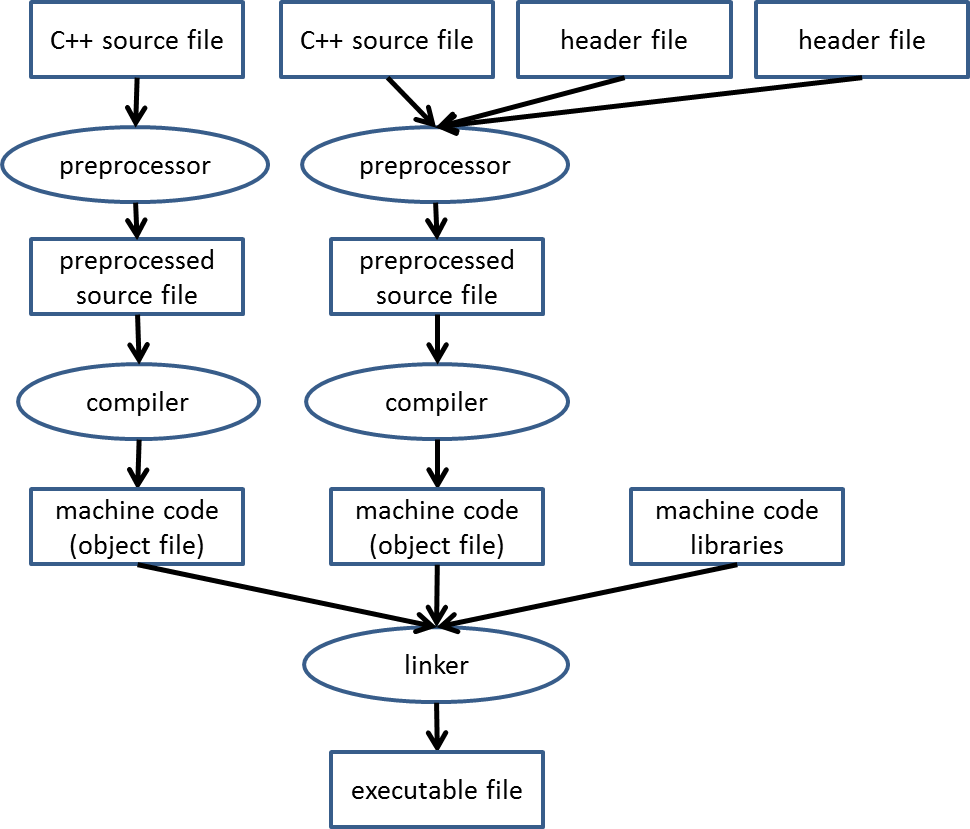
Name: *John Hellrung*

This reflection is to be completed individually, although consultations with TAs and classmates are encouraged as long as they are appropriately acknowledged. This document is intended to help you think about the various concepts covered in Chapter 8 of the textbook we asked you to SKIM. Remember, we are asking you to SKIM through the chapter, hopefully presenting material you have already read in the PDF document in a different way, thereby reinforcing the ideas.

## The Compilation Process

Our programs written in Python are interpreted, meaning that there is a small program called an interpreter that takes each line in the Python module one at a time and converts it to machine code before executing it. As discussed in the text, a C++ program is compiled, which means that the entire source code is analyzed and translated into an executable file. We will consider a broad understanding of the compiling process.

Consider Figure 8.1 in page 272 of the textbook and reproduced with slight changes below, and answer the following questions.



1. What does the "preprocessor" stage do? Specifically, what does it do with the header files?

|  |
| --- |
| The preprocessor stage takes the source code and processes all lines of code in file with the pound sign and transfers them to the compiler. When preprocessor finds a pound sign it copy and classes in the header file so that the source file can excess the file class(es). |

1. Briefly describe the process that the compiler goes through to generate object files from source code, making sure to explain the impact of syntax errors into this process.

|  |
| --- |
| The compiler first step to check for syntax errors and next if your code has syntax the compiler prompts the user and stop running but if you have not syntax errors then the compiler runs. When the compiler runs it converts the source code into machine code (1 or 0 binary). |

1. Briefly describe how the linker takes the various object files and creates an executable file.

|  |
| --- |
| The linker combines the various machine code object files into one executable program that then is compare to machine code libraries in order to see if the machine code libraries supports the machine code object. |

## Input/Output and Variable Declarations

Take a look at the following C++ program:

* 1. #include <iostream>
  2. using namespace std ;
  3. int main ()
  4. {
  5. int x = 3 , Y = 5 , temp;
  6. cout << x << " " << y << endl ;
  7. temp = x ;
  8. x = y ;
  9. y = temp ;
  10. cout << x << " " << y << endl ;
  11. }

What is contained in the iostream header file, and what does the #include directive do with it?

|  |
| --- |
| The header file “iostream” contains C++ standard library and #include copies the classes inside the header file so the in the source code can use the function calls throughout the code. |

Briefly explain what cout and cin are and what they are used to do:

|  |
| --- |
| cout is a function of ostream that allow the code to print format string prompted out on to the screen. Cin is the same way except it take in data as a string. |

Why does the variable temp have random values in it before line 7?

|  |
| --- |
| It because c++ needed to have varable defined because for you can set it to a value. |

## Function Declarations/Definitions

Consider the following program:

1. // this example will not compile
2. int main ()
3. {
4. double a=2.5, b=3.0, c;
5. // the compiler has not yet seen the f function
6. // so it cannot determine if f is called correctly
7. c = f(a, b);
8. }
9. double f(double x , double y)
10. {
11. return x \* x + 2 \* x \* y ;
12. }

What would need to be added to the code above so that the program will compile?

|  |
| --- |
| #include <iostream>  using namespace std ;  These two lines need to be added. |

What is the difference between a *function* *declaration* and *function* *definition*?

|  |
| --- |
| A function declarations tells the compiler about the function name and how to call them. A function definition is the body of the function that supply the directions to execute. |

What are the *formal parameters* of the function f(…)? What are the *actual parameters*?

|  |
| --- |
| Formal parameters: A formal parameters is a parameters like x that stands on a value.  Actual parameters: The Actual parameters is the value used in the function. |

## Looping and Variable Scope

Consider the following code snippet:

1. #include <iostream>
2. using namespace std;
3. int main()
4. {
5. for ( int counter=0 ; counter<10 ; counter++ ) {
6. cout << counter << endl ;
7. }
8. // other code can go here.
9. return 0;
10. }

How many times will the for loop execute?

|  |
| --- |
| 9 times |

Briefly explain why the variable counter does not exist at line 9.

|  |
| --- |
| Because the variable for counter is created in the for loop. |

Insert C++ code to the main() function below for a while loop that is equivalent to the for loop above:

|  |
| --- |
| #include <iostream>  using namespace std;  int main() {  int index = 10;  // replace me with code of the equivalent while loop  While index < 10{  index++;  cout << index << endl;  }  return 0;  } |

## Summary

Give at least 6 important concepts that Python programmers should keep in mind when creating a C++ program. Any explanations must be in your own words, so if you are quoting the text, you must explain why you think the concepts are important.

|  |
| --- |
| 1. You need to assign a type to a variable before assign a variable a values. 2. You need to remember your brackets in c++ when making a function or any a logic statements. 3. Remember that you need a main function. 4. Remember that there have public and private members in c++. It not like in python where you just make a variable self.\_ and it is so. 5. Remember that you can’t just print in C++, you have to cout string and cin to get input from the user. 6. Remember that you have assign a variable to your class as well. This especially important with the type named “void” witch mean that the complier does not search for a return. This mean that if you put a return statement in a void function the complier will not read it. |