Name: Soham Thaker SID: 011-748-159 Email: sdthaker1@myseneca.ca

First, I learnt how to create modules and how they help us with writing programs efficiently. A module consists of an implementation and a header file. Implementation file consists of logic, for example function definitions where the logic of the functionality is written and how the function will behave when it is called by the caller. A module also contains a header file, where the function declarations, class declarations, among other declarations are present which will expose the identifiers to the client module for use. Both of these files make up a module and a translation unit is compiled at the time of compilation to then pass it on to the linker which links the preprocessor directives and translation units to create a complete executable file.

The aspect of creating a module is that each module focuses on solving one problem out of many problems to be solved. Multiple modules when compiled together will solve the problem given to the programmers. It hides as much detail as possible from the client file related to the logic of how a particular piece of code is written. The client file only knows what to pass when a particular function is called and where to store/display the value. Modules also help in code readability since each module tries to focus on one problem, and thus updating the code in any module makes it easy because chances of breaking code written in the other modules are less.

Second, I learnt the use of namespaces and why they are necessary in a program. If let us say there are multiple developers working on project, chances are very high that developers will use same identifiers for multiple variables in a project and this will lead to naming conflicts at the time of assembly. To avoid such conflicts, we use namespaces where the variables, functions, struct, classes among entities are defined and declared within a scope. Furthermore, to use the entities defined and declared into a client code for example, we write the statement, “using namespace identifier;”.

Third, I learnt why we use header guard and what is its functionality. Header guards protect the contents of a header file, from being included more than once. When header guards are in place, once a header file has been declared, the guards will protect the file from being included again, hence protecting against multiple definition errors since C++ has a one definition rule where for example, a definition of a struct or a function or any other entity may only appear once within its scope or a translation unit.