/\*

Student: Marina Fiamskaia

Student ID: 025-698-051

email: mfiamska@myseneca.ca

Date: May 21, 2022

Subject: OOP345 NAA

Project: Workshop 1 part 2 reflect

\*\*\*\*I have done all the coding by myself and only copied the code that my professor provided to complete my workshops and assignments.\*\*\*\*

\*/

The main concepts that were covered in the workshop one are: the main function that accepts command line arguments, its proper syntax and use; placing a header guard on class definition from duplication in a translation unit; defining and accessing a variable in a different translation unit; declaring a local variable that resides in a memory throughout the lifetime of the program; upgrading code to accept and managed a user-defined string of any length. It also covered a review of previously learned concepts such as dynamic c-strings and rule of three; namespaces; scope resolution; defining and accessing global variables.

The very first new concept covered in this workshop is receiving program arguments from the command line. In order to do that I have used overloaded int main(int argc, char\*argv[]){} instead of default int main(){}. The form used allows for user defined elements to be passed down to the application using command line interface, such as files to read from. First argument in overloaded int main() represents the total number of element that will be passed to the application and second argument represents an array of elements passed to the application.

Furthermore, I have learned the practical “in a code” meaning of the difference between internal and external linkage and storage duration. The internal linkage concept represented in this workshop by static int counter variable, declared on line 69 of foodorder.cpp module. This variable is visible to only foodorder.cpp module. It is created and initialized only once in the program, it got global lifetime but local scope. The external linkage concept is represented in this workshop by global variables the g\_taxrate and g\_dailydiscount that are used in different scopes in different modules. I have used extern keyword in foodorder.h and defined the variables in foodorder.cpp in order to make them global and accessible in all translation units including w1\_p2.cpp and not to break the one definition rule.

Additionally, I have made slight changes to the FoodOrder module, as per instructions, to be able to accept string of any length as a value to one of the class member variables. To be able to do, I have declared m\_foodDesc member variable as a pointer to a string that will be dynamically populated and initialized it as a nullptr. Another change was I have added three member-methods to the class FoodOrder: the copy constructor, the copy assignment operator, and destructor. The reason is The Rule of Three when dealing with dynamic memory in a class. Also, I have changed my set method so that instead simply copying a string from source to destination, it is first deallocating the memory, then it dynamically allocates new memory and only then it copies the source string to the newly allocated destination memory. And lastly, I have changed the syntax for the read method, so that it will be capable to read string of any length into variable type std::string.

Moreover, I have used head guards in the module foodorder.h in order to make sure it will be declared only once in the translation unit, which follows the one definition rule.