Ryan Hoffman
CS 202
Project 10
Lab #: 1107

Compile Instructions: make -> ./proj10 (there will be warnings but no errors)

The purpose of this project was to create a stack using an array-based method as well as a node-based method. Both methods utilized templates to make the stacks usable for multiple data types. Other things included in the project are dynamic memory, classes, pointers, and io stream. We were given the header file of a non-templated array and node based class and we had to implement them using templates. The stack's function is to insert and remove data from the top of the stack, therefore the only node data member in the stack class was a m_top node pointer.

My design was fairly straight forward. I followed the sample template file given in the previous lab to template both classes. To move up and down the stack I used the size_t m_top variable to iterate through the stack. Since it wasn't dynamically allocated I didn't need to implement a destructor. The m_top variable ends up being the index of nothing so in order to get the top variable of the stack we need to use m_top-1. Popping variables from the stack was as simple as just decrementing the m_top variable. When pushing a variable to the top of the queue, I had to assign the m_top index to the value that I was pushing and them increment m_top. Checking if the stack was full was as simple as checking if m_top was equal to the MAX_STACKSIZE const global variable. The node-based stack was more complicated because it involved pointers and dynamic memory allocation. Constructing a node-based queue involves dynamically allocating memory at the m_top node pointer using the node class that we also had to implement and create a template for. Getting the top element of the stack involved using the m_top pointer to call the data() function which returns whatever data member was on the top of the queue. The push function just allocates a new Node data member at m_top and the pop function deletes the current m_top data member. When the stack is empty m_top will be NULL. The test driver (proj10.cpp) tests the default, parameterized, and copy constructors as well as the equal operator overload, destructor (if applicable), push, and pop functions.

I had two problems over the course of this project. The first problem was that the for loop I was using to iterate through the array-based stack wasn't printing all of the data members in the stack. I fixed this problem by modifying the conditions of the for loop. The last problem I had was that my destructor wasn't working, causing a memory error whenever the program was done running and the data members were being deleted. I had to completely redo my deconstructor but eventually it worked.

If I had more time I would implement more of the functions into the constructors and operator overloads. I could have used the clear(), empty(), push(), and pop() functions throughout the program but I didn't use them because I had already implemented the functions that would have used them. In the future I should implement functions like those first so that I don't have to write as much code in the constructors and other functions.