Nicholas Ang

CS 202 - 1101

Project 10 Documentation

Purpose: The purpose of this project is to test our ability to create templates and templated classes in regards to creating a stack. The project requires the implementation of an array based stack and a node based stack which involves dynamic memory. Additionally, the project tests our basic c++ skills such as pointers. The project requires us to be able to create test drivers for our classes and to be able to manipulate the data inside of the stacks. The project also requires us to use hpp files instead of standard h files and cpp files.

Design: The project is set up so that the include folder has all the hpp files and the src folder holds nothing. There are three hpp files which are ArrayStack.hpp, NodeStack.hpp, and Node.hpp. They hold the declaration and implementation of the templated classes. The Node file holds the template class for node which has constructors and a method to access the data inside the node. The ArrayStack file holds fifteen methods and members. It has a default constructor which creates a stack with null values, a parameterized constructor that creates a stack with given values, a copy constructor that creates a stack with the same values as another stack, and a destructor that clears out the stack before destroying the object. The assignment operator = does the same thing as the copy constructor but assigns the values of one stack to another. The top function has a normal version and a const version that returns the top of the stack. The push function adds a new element to the top of the stack while the pop function deletes the top most part of the stack. The size function returns the size of the stack. The empty function returns true when the stack is empty while the full function returns true if the stack is full. The clear function

sets the array stack size to 0 which effectively clears the stack. The serialize function outputs the information of the stack from top to bottom. The operator overload for << calls the serialize function and outputs the information to the terminal. The NodeStack is similar to the ArrayStack with some differences and dynamic allocation. The NodeStack class also has fifteen methods and members. It has a default constructor which creates a stack with null values, a parameterized constructor that creates a stack with given values, a copy constructor that creates a stack with the same values as another stack, and a destructor that clears out the stack before destroying the object. The assignment operator first checks for self assignment and then clears the stack. It then copies and assigns the values of the right hand side to the assigned stack. The top function has a normal and a const version which returns the top node of the stack. The push function pushes a new dynamically allocated node and links it as the top of the stack. The pop function dynamically frees the memory and deletes the top most node. The size function iterates through each node and returns the size of the stack. The empty function returns true if the stack has no nodes inside while the full function always returns false as the stack can never be full. The clear function dynamically frees each element in the stack until the stack is empty. The serialize function outputs the data of every node in the stack. The operator overload for << calls the serialize function to output the information to the terminal.

Problems/Challenges: There were some slight troubles with this project. As templates were a fairly new concept, I had to use the samples to reference for my own code. Also setting up an hpp file is somewhat different than normally having a header file and a cpp file. I also had some trouble with the program outputting forever but I figured out where the problem was and properly fixed it. I also had a problem with my pop function in the nodestack class as I did not

properly deallocate the memory. Overall, the project was not too hard and was fairly straightforward in its instructions.

Possible Changes: I would try to clean up the code and make the code simpler. It was annoying to have to type template <typename T> in every single function and for declaring the templates. I would try to learn more about templates because templates were a little confusing. I would also try to look at different types of data structures such as stacks and queues and linked lists to better help me in my coding.