**Programming Project Report**

Name: Riley Nadwodny

Date: 3/23/2020

**Academic Integrity Statement:** I pledge that I have neither given nor received unauthorized help on this programming assignment.

**Problem Statement:**

The goals of this programming assignment were to help us understand the function of queues and stacks, as well as the implementation of queues and stacks. In particular, we had to create a circular queue for this project. The program input for this project was how many times to run operations on the stack and the queue, and the program outputs are the data that is stored in the stack and the queue.

**Design:**

This program contains five files: a main.cpp, a stack.h and queue.h header filers, and a stack.cpp and queue.cpp files that contain the function definitions for the stack and queue header files. For both the stack and queue classes, a vector was used in order for the stack and queue to be able to be more dynamic instead of static. The stack class contains methods to push and pop values off the stack, as well as checking if the stack is full or empty. The queue class contains methods to insert and remove data from the queue, keeping track of the front and back of the queue in case the data wraps around the vector.

**Implementation:**

To implement this program, I began by implementing the header file for the stack and creating skeleton functions for the stack class. After coding and testing each function in the stack class, the main program was coded. The main program contains a variable to take in the amount of operations to perform desired by the user. Sample code was provided by John Gauch on the CSCE 2014 website under “source code” and was adapted accordingly to fit the project. After finishing the stack implementation, the queue header was created with the skeleton code provided. The .cpp for the queue was filled out with skeleton code, with the simpler functions such as isFull and isEmpty being filled out first. Lastly, the remove and insert methods were implemented, and then the same for loop was implemented into the main program for the stack and the queue in order to run the same number of operations on both data structures. The stack was designed and implemented in a couple of hours, while the implementation and design of the queue took at least 10-12 hours of testing and tweaking in order to get it about right.

**Testing:**

The program was tested using inputs of 100, 1000, 10000, 100000, and 1000000. There were no special cases tested with this program. The stack implementation worked as intended, and the queue mostly worked. There are a few exceptions in the queue where some data is lost, but it only happens once or twice at the most per run; otherwise, both function as intended and as expected. These data structures took countless hours of testing and tweaking to make sure that they work correctly, and by the end of it, they were mostly successful. All sample output is included in the typescript.txt file included in the “Project4-Final.zip” folder uploaded to Blackboard.

|  |  |  |
| --- | --- | --- |
|  | **Stack Runtime** | **Queue Runtime** |
| 1,000 | 0 s | .015 s |
| 10,000 | .468 s | .124 s |
| 100,000 | 32.749 s | 2.906 s |

**Conclusions:**

Overall, this assignment was a relative success. The stack works as intended, and the circular queue works mostly as intended, given the circumstances surrounding this project, such as classes suddenly being moved online due to COVID-19, Professor Gauch not receiving emails properly with his UARK email address, and online lessons being posted instead of having live lectures in person or live lectures online using Blackboard Collaborate. Next time, I would start earlier on the queue implementation to have more time to ask questions and figure out the slight errors going on with the programming. The project took the full two weeks of the allotted time from start to finish.