Ryan Hoffman
CS202 Project 9 Documentation
Lab Section: 1107
4/21/2019

Compile Command: /make all -> ./proj9

The purpose of this project was to be able to make and use queues that were array based as well as node based. Dynamic memory, linked lists, classes, and operator overloads were also included. We had to create an ArrayQueue class and a NodeQueue class which both performed similar functions. The ArrayQueue class didn't need some functions to be implemented because ArrayQueue didn't use dynamic memory allocation. The main functions in this program were the push and pop functions that are used to add items to the end of the queue or remove them from the beginning of the queue. The front data member in the que points to the first element and the back pointer points to the last element in the queue.

My design was to be as logical as possible when figuring out how to write the code for the constructors and the push/pop functions. In the push function I used the m_back=(m_back+1)%ARRAY_MAX to push a value at the end of the queue and for the pop function I used the m_front=(m_front+1)%ARRAY_MAX to remove an element from the beginning of the queue. For the NodeQueue class, I needed to use memory allocation, so I had to design a destructor for that class. The algorithm I used for this class was to dynamically allocate a new Node and then set m_back->m_next to that new element and then set m_back to the new element. For the pop function, I created a temp pointer that would hold the value of m_front. I would then set m_front to the to the next value and then delete the temp pointer.

I didn't have any problems regarding the ArrayQueue class but I did have a few problems with the NodeQueue class. I had a lot of trouble trying to get the copy constructor and the equals operator overload to work. I kept getting a core dump error and a large debug error. This project was definitely a lot easier than the last one mostly because I had more practice with how nodes and the lists work. Applying linked lists to node queues wasn't nearly as bad. For the copy constructor and equals operator overload I fixed the error by setting a curr pointer to the rhs front pointer and then until curr was NULL I pushed curr's m_data into the node queue object being called and then moved curr to the next value in rhs' queue.

If I had more time I would do a better job at implementing the other class functions into different functions like the constructors. Instead of using the push and pop functions in the constructor I created the constructors first and then the push and pop functions later. I realized that my constructors used the same code as the push and pop functions. One thing I noticed in this project is that the isFull() function for the node class couldn't really be implemented because the node queue is dynamically allocated meaning technically there is no max size, as long as you have enough memory.