This project shares a lot of similarity with project 8, so I reused a lot of code from project 8, with only a few minor adjustments, particularly incorporating the m_back member to work with them.

The ArrayQueue class was a lot more straightforward than that of the ArrayList we made last project. For one, I didn't have to deal with all the dynamic memory stuff which was nice. Many of the function algorithms were covered in the lecture, so I didn't really come up with anything too unique, and none of the functions ever really went above 10 lines long.

The NodeQueue class was a bit more challenging in that I had to repurpose my code to work with the m_back member. My solution to this was that, when copying over the information from one queue to another (whether it be during the copy constructor or the operator=), I stopped at the element right before the last one, updated the m_back pointer, then copied the last element over. Because the NodeQueue has so much similarity with the NodeList, I could reuse a lot of the functions I had previously made, with little to no changes.

I had to make a couple modifications to some of the header files in order for some of my functions to work. In the Node class, I had to add the NodeQueue class and the NodeQueue operator<< as friends otherwise I would not be able to access the m_next pointers within the node in order to traverse the queue and print out its contents.

As a side note, I did not include any built-in checks to determine whether a queue was empty when using the front and back operations. Usage of those functions should probably be left to the user however. I read that some implementations of front and back throw exceptions if the queue is empty, but I did not include that in my code.