This project has taught me a lot about working with generics and the why they’re important. Generics allow you to create powerful containers, like a Stack or Queue. I had some issues at first dealing with how I’d implement a queue. Finally, I decided to just shift every item to the left when an item is removed. I had issues understanding the implementation of the postfix to infix. I had to do a lot of research about how exactly it worked. The algorithms helped me program it, but I wanted to understand how exactly it worked. While you explained it in class, I also figured out why the algorithm works in the first place. It really bothered me that there was no required exception called invalidCharacterException, because it’s not really a format error, it’s that the program doesn’t know what to do with these characters, such as letters or other operands such as modulo (%). I made the invalidCharacterException class and made it thrown in one of the Notation utility methods, but ended up catching it at the same time because I didn’t want to mess up the instructor tests by adding some unnecessary stuff to the method’s signature. However, I do believe something like that is necessary for usability. When I first made the design, I thought that I needed to include a getElement() method, and while it would have been nice to have, it wasn’t really necessary for this implementation. However, I still think that it would be a good idea to have the getElement method, but it is never specified if a queue or stack is ordered or not, but I assume it is because you deal with the front and back of the stack or queue. Overall, I learned a great amount about generics and thought about the different ways I could use the data structures, NotationStack and NotationQueue.