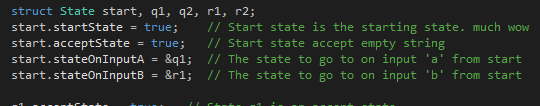
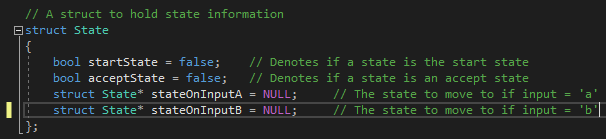
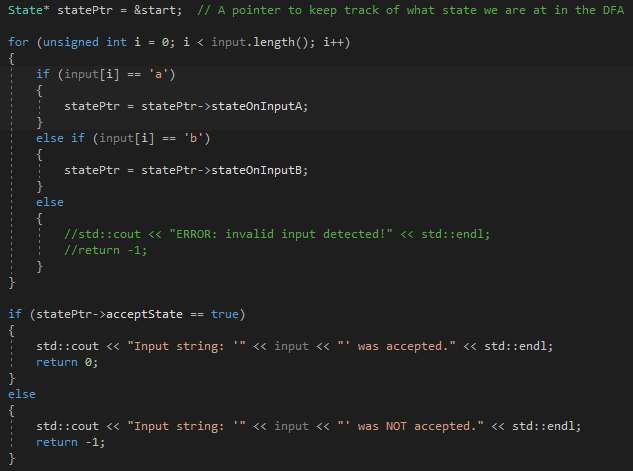
Tyler Bartlett

CS361 Lab 4

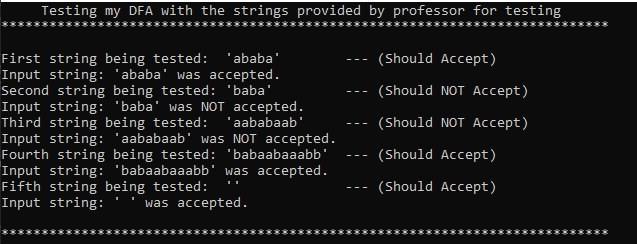
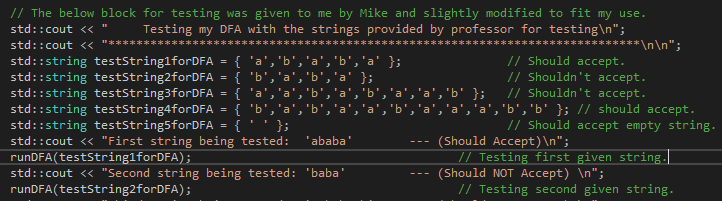
This Lab started out easy for me when I did the DFA. I made a “state” object, then made many instances of the states, and populated the fields with the given instructions from the lab handout. Below is a screen shot of the state object and an example of how each state was setup, but with correct values for that state.



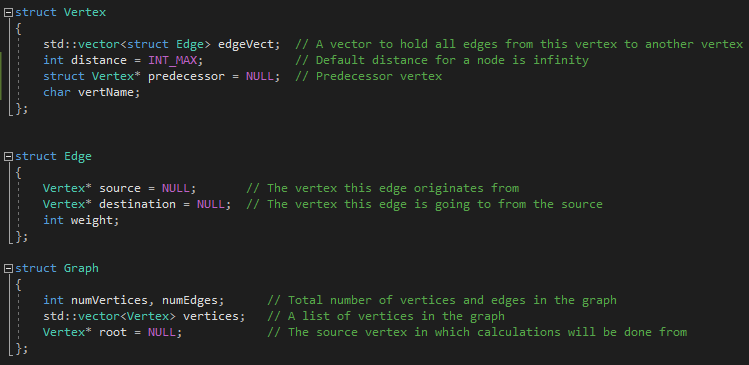
I then checked the input from the strings and moved a pointer through the DFA from the start state to the correct state on its given input. With the DFA, I assumed input was checked prior to strings being sent to the DFA for analysis.



Mike Dietrich had provided me with his strings build and function calls to test them on the DFA which I used as a time save to not have to type it all out. Below is an example of that and the outputs of each string to test from the lab handout.

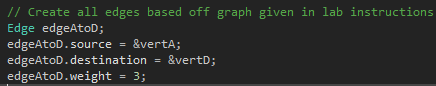


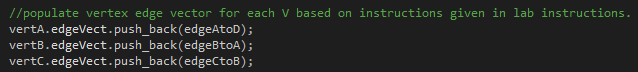
Next up was the task of the Bellman Ford Algorithm, which quite frankly proved more troublesome than anticipated. I decided to make an edge object to represent each edge, a vertex object to do the same, and a graph to hold the information as seen in the below screen shots. I had decided that giving each vertex an array of edges would be good to track this information and how I had it done in my head, seemed to make the most sense. Jacob McLeod and I worked on writing the Bellman Ford algorithm together but hit a brick wall. Vertex K proved to be the bane of our existence because its array of edges is zero which completely broke the logic we had written which was based off the edge array at each vertex. Due to time constraints I could not improve much further upon our algorithm, but I got it to work correctly for each vertex other than Vertex K. with our relax function, it was not originally working with temp pointers so I brought it back into the same loop and broke it down by the objects pointers next vertices and what not. I have a comment in the code about an automagic fix to make declarations work because they for some reason didn’t work in the “relax portion” of the for loop. The automagic fix finds the needed distance and predecessor information buried in the objects edge vector data. I don’t know why it needed this crazy band aid but it worked, kind of. Except for vertex K.



The below four screen shots are examples of building the objects.

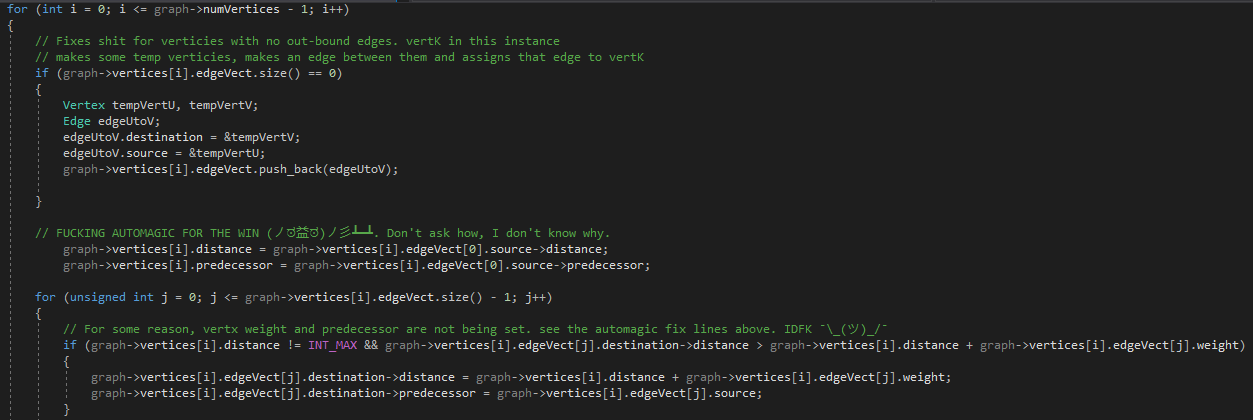








The next below screen shot is a portion of the bellman ford which mostly worked. I put a few different band aids in the loops to get around the issue caused by the automagic not magically working, then working with more automagic and to get around vertex K.



And the last screen shot here is the output of the Bellman Ford algorithm, except with vertex K not displaying correctly. The correct values for K should be distance 2 and Predecessor vertH based of my hand trace of the algorithm.

