Seth Cram

CS121

* I began by working off of a pre-established link list and queue class
* I was impressed to learn just how easy it is to link four files in the same directory to one another
* The first main problem I ran into was the dynamical array in general
  + Didn’t know how to create one
  + Didn’t know how to read information into it when created
  + Learned that you can create a pointer to connect you to an array of pointers to allocate memory dynamically in arrays
* My other enqueue class functions couldn’t gain access to this array
  + Had to define the function that created and filled the array within the queue class
  + Solved the scope problem
* Next problem was how to enqueue the values surrounding the current position of a node
  + I kept getting an error when I’d try to compare an array spot that was out of bounds to anything
  + I had to compare characters to surrounding values in order to decide whether to enqueue the character or not
  + Had to go back to where the maze got read in and dynamically allocate enough space for empty borders around the maze
  + Now there’s something to compare, so it worked
  + As a result, I had to alter my ShowMaze function to print values two wider and two higher than it did before
  + I learned that values outside an array aren’t just automatically set to NULL or zero
* Actually traversing down the maze began giving me problems
  + I couldn’t get the current position to change when dequeuing a node
  + Even when passing the position of the enqueueing node to the node class, the passed position variables weren’t in the scope of the queue class functions
    - Therefore I couldn’t access the coordinates
  + Resorted to creating two new global variables to take the coordinates of a node when dequeuing, right before that node was deleted
  + Now I had the position of the current node every time I dequeued a node
    - I could now read in the surrounding values and enqueue the moveable places
* Added in a recursive statement to the bottom of the WalkMaze function allowing me to search for the goal, even after running into a corner with no available move spots
  + I learned that I still don’t understand recursion that well since I wasn’t able to determine when the recursion would stop and run back
    - Or atleast, how to illustrate that
* I thought I finished, but then went to test the code and it failed when the maze couldn’t be solved
  + I wrote a size function to return the size of the list as a member of the list class
  + Then used this in the queue class by creating bool IsEmpty() that incorporated the size function as a member of the queue class
  + Added the IsEmptyFunction in to stop the WalkMaze function from recursing when the maze wasn’t solvable
* A big takeaway I got from this assignment was that many functions make light work of a hard problem
  + As long as scope problems are overcome, of course
* This assignment took me, in all, around 16 hours
  + A large amount of this time was spent thinking about what next step I wanted to take and how I’d accomplish that, or how to overcome a specific problem I had run into
  + Toward the end I realized that it was much more efficient just to bypass my current problem and keep working on the code as if it worked, rather than get frustrated for hours on end over a single issue
    - This helped me complete it much more efficiently, but was rather late