Elena Escalas

CS 31 Project 3

1. I initially tried doing navigateRoute without modifying the route string to make it easier on myself, and I got the position values very confused, so eventually I just decided to modify route into newRoute so that I would only have lower case letters to work with instead of numbers. The newRoute I created simply had everything as a letter, so for example if there were ‘e2’ in route, it would be converted into ee and added to newRoute using for-loops

Another one of my big obstacles was I kept getting “isWall invalid value: (0,3)” in my navigateSegment portion. It took me a while to realize that this was because I was decreasing the row by one, and I had made sure r was in the grid by saying that it was less than getRows(), but in actuality it also needed to be greater than zero. When I would input a value for r it wouldn’t exit the loop if r went below 1 because the only restriction on r was that it had to be less than the number of rows, not greater than zero.

I also made the silly mistake many times of setting my stepsMoved to nsteps AFTER I returned a value. I found many cases where I had changed a variable after returning a value, and then the integer would not set because I would return a value and exit the function before being able to set the value to the other integer.

1. bool isRouteWellFormed (string route)

check every character in the string:

if the first character is a digit:

return false;

if the character in this position is not a digit

and is not a letter:

return false;

if character is a letter:

set letter as lowercase letters

if letter is n, s, w, or e:

continue to next character;

else return false;

else if character is a digit:

set digit1 to that digit;

set digit2 to the next character;

if digit1 is negative:

return false;

else if digit2 is a digit:

if digit2 is negative:

return false;

else continue on to next position;

else continue on to the next position;

if every position has been checked and false has not been returned:

return true;

int navigateSegment (int r, int c, char dir, int maxsteps)

set nSteps to -1

if the row and column are not within the grid

return -1;

else if the row and column are at a wall

return -1;

else while row and column are within the grid limits

if row and column are a wall

break out of the loop;

else increase nSteps by 1

if the lowercase direction input indicates north:

move the point being tested up one by subtracting 1 from r;

if the lowercase direction input indicates south:

move the point being tested down one by adding 1 to r;

if the lowercase direction input indicates east:

move the point being tested right by adding 1 to c;

if the lowercase direction input indicates west:

move the point being tested left by subtracting 1 from c;

if nSteps is greater than or equal to the maxSteps:

return maxSteps;

else return nSteps;

**To test whether or not positions were valid I created another function:**

bool clearPosition(int row, int column)

if the row or column is greater than the grid size allows:

return false;

if the row or column is less than the grid size allows:

return false:

if the row and column position is a wall:

return false;

else return true;

navigateRoute(int sr, int sc, int er, int ec, string route, int& nsteps)

if route is not well formed or the **start/end positions aren’t valid**:

set nsteps to zero and return 2;

set k equal to zero;

create the string newRoute;

while k is not equal to the size of the route string:

if the first character is a digit:

set nsteps to zero and return 2;

if the character at position k is a letter and the next character is a zero:

continue to the next character (basically skip this letter)

if the character at position k is a letter:

add the lowercase of that letter to newRoute and move onto next position (increase k by one);

if the character at position k is a digit:

set number equal to the integer version of the character;

if the character before the digit is a letter:

if this number is zero:

continue onto next character;

print out the letter before the digit the digit number of times then move onto the next letter;

if the character before the digit is a number:

convert the two numbers into a two digit number;

print out the letter before the digit the digit number of times then move onto the next letter;