Report for Project 3

2. a

There were many obstacles I faced while doing this project. The hardest part was figuring out how to break down the large specification into small parts so I could make sure that all parts of the program functioned as specified. In addition, I learned that, at a certain point, it becomes difficult to follow your own code so it is helpful to add comments as you code. Furthermore, it is also necessary to try to make the framework of a function as independent as possible so that if it does not work correctly, the problem is isolated to a smaller area of code. Finally, when testing a function that calls other functions, those other functions should be thoroughly checked first.

b.

bool isRouteWellFormed

set a digit counter to zero

if the given route length is less than or equal to two and the first character is a digit

return false

for each character in the string:

if the next character is a direction

set the digit counter to zero

if the digit counter is greater than or equal to three

return false

else  
 if the character is a digit

increment the digit counter by one

if the digit counter is greater than or equal to three

return false

else

return false

int navigateSegment

set possible steps to zero

if the initial position is a wall or is outside the grid

return negative one

if the given direction character is a valid direction

if the valid direction is north

while the next spot along the direction is not a wall or going off the grid

increment possible steps by one

if the possible number of steps is less than the given steps to take

return the possible steps

else

return the given steps to take

do the same thing for the other three directions

return negative one

int navigateRoute

set start row to the given start row

set start column to the given start column

set temporary nSteps to zero

if the given starting position or the ending position or the route is invalid

return two

set a to zero

while a is less than the total number of characters in the given route

set theoretical steps to zero

set new Index to zero

if the next two characters are both digits

set theoretical steps to that number

increment new Index to the next segment

if only the next digit is a digit

set theoretical steps to that number

increment new Index to the next segment

if only the direction

set theoretical steps to one

increment new Index to the next segment

set the possible steps to the return value of navigate Segment using the above information

if the steps possible is lower than the theoretical steps

return three

based on whether the direction is north, east, south, or west

change the start row and start column based on the direction and possible steps

set a to the new Index

if the current row and column position matches the end row and end column

return 0

else

return 1

c.

All tests are assuming that the grid looks like the one shown as an example in the specification

bool isRouteWellFormed(string route)

Valid case: n12swE3s (should return true)

Number in the beginning: 6ns2W3 (should return false)

Three or more digits in a row: e012WsE2 (should return false)

Only numbers: 124535346 (should return false)

Any characters besides digits and a valid direction: n3#sw+12 (should return false)

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

Invalid row: (25, 1, ‘n’, 5) (should return -1)

Invalid column: (1, 25, ‘n’, 5) (should return -1)

Invalid direction character: (1, 1, ‘q’, 5) (should return -1)

Invalid maxSteps: (1, 1, ‘n’, -6) (should return -1)

Trying to move off of grid: (1, 1, ‘n’, 3) (should return 0)

Trying to move through a wall: (2, 4, ‘e’, 3) (should return 1)

Not being blocked by anything: (1, 1, ‘s’, 2) (should return 2)

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

assume totalSteps has been previously initialized

Invalid start row: (0, 1, 1, 3, “e2”, totalSteps) (should return 2)

Invalid start column: (1, 0, 1, 3, “e2”, totalSteps) (should return 2)

Invalid end row: (1, 1, 0, 3, “e2”, totalSteps) (should return 2)

Invalid end column: (1, 1, 1, 0, “e2”, totalSteps) (should return 2)

Invalid route: (1, 1, 1, 3, “q2”, totalSteps) (should return 2)

Check if reached end: (1, 1, 3, 1, “s2”, totalSteps) (should return 0 with totalSteps = 2)

Check if didn’t reach end: (1, 1, 3, 1, “s2”, totalSteps) (should return 1 with totalSteps = 1)

Going west into a wall: (2, 3, 2, “w”, totalSteps) (should return 3 with totalSteps = 0)

Going east into a wall: (3, 1, 3, 2, “e”, totalSteps) (should return 3 with totalSteps = 0)

Going south into a wall: (1, 2, 2, 2, “s”, totalSteps) (should return 3 with totalSteps = 0)

Going north into a wall: (2, 4, 1, 4, “n”, totalSteps) (should return 3 with totalSteps = 0)