a. Obstacles:

* For the first function, bool, it was difficult for me to convert my pseudocode into a logical series of expressions in the actual code
* For the second function, navigateSegment, I had a tough time coming up with the for loop to create the movement in one direction in the warm-up
  + Once able to create it, it was easy to replicate the function and tweak it slightly to suit all four directions
* The third function was the trickiest and it was difficult to create loops that would process the digits after a given direction
  + The links provided on the webpage helped to create the necessary code to process the strings of characters and digits

b. Pseudocode

bool Function: Used to check if a route is a syntactically valid one

Loop to process string

if first char is valid (N/S/E/W/n/s/e/w)

if next char is valid

pass

if next char is digit

if next char is valid

pass

increment int (k) by 1

else if next char is digit

if next char is valid

pass

Increment k by 2 (k = k + 2)

Make sure digitCount is not greater than two

else

fail

else

fail

else

fail

else

fail

int navigateSegment Function: Determine max steps in a given direction

Set conditions for valid position

Four different if functions for each dir (N/S/E/W)

for loops

set int i to either r or c

keep incrementing or decrementing until robot hits a wall

if wall

break

else

increment counter

if count less than max steps

return count

else

return max steps

int navigateRoute Function:

Set conditions for valid start and end positions

Set nsteps to 0

For loop to process route string

if first char is valid

create a new char and set it equal to dir

create a new string

set a new int equal to 1

if next char is digit

while loop to process digit after the char

calls a new function that turns a string of digits to int

initialize steps

call the navigateSegment function

if steps is less than c

return 3 and set nsteps to steps

if N/S/E/W then either add or subtract steps from the start row or column

increment nsteps

if start row and start col are equal to end row and end column

return 0

else

return 1

c. Test Data Tests:

* For bool function:
  + empty string
    - To make sure it will pass if it is an empty string
      * pass
  + 1 segment with no digits
    - To make sure it will pass if it is a single segment with no digit
      * pass
  + 1 segment that is a digit
    - To test what would happen if it is a single segment of only a digit
      * fail
  + Segments with char followed by digits
    - To make sure it will pass if it is a segment with digits after a valid dir, but not more than 2 digit
      * pass
  + Segments with only digits
    - To test what would happen if it is a segment of only digits
      * fail
  + Segments that have 3 consecutive digits
    - To test what would happen if a segment had 3 consecutive digits
      * fail
  + Segments that have multiple dir in a row
    - To make sure any consecutive dir will pass
      * pass
  + Segments that have invalid characters
    - To test what would happen if invalid dir were given
      * fail
* For navigateSegment function:
  + r or c that are outside the grid
    - To check if return value is correct if outside of grid
      * return -1
  + r or c that start at a wall
    - To check if return value is correct if it is at a wall
      * return -1
  + dir that are invalid
    - To check if return value is correct if given an invalid dir
      * return -1
  + if maxSteps < 0
    - To check if return value is correct if maxSteps given is less than 0
      * return -1
  + Normal cases
    - (3, 1, 'N', 2)
      * To see if it will produce correct output in a normal case
        + return maxSteps
    - (2, 3, 'e', 1) etc
      * To see if it will produce correct output in a normal case
        + return maxSteps
* For navigateRoute function:
  + sr or sc that are outside of grid
    - To check if return value is correct if outside of grid
      * return 2
  + er or ec that are outside of grid
    - To check if return value is correct if outside of grid
      * return 2
  + sr or sc that start at a wall
    - To check if return value is correct if starts at a wall
      * return 2
  + er or ec that start at a wall
    - To check if return value is correct if starts at a wall
      * return 2
  + a route that is not well formed
    - To check if return value is correct for an invalid route
      * return 2
  + set of tests in the spec:
  + int len;
  + len = -999; // so we can detect whether navigateRoute sets len assert(navigateRoute(3,1, 3,4, "N2eE01n0s2e1", len) == 0 && len == 7) len = -999; // so we can detect whether navigateRoute sets len assert(navigateRoute(3,1, 3,4, "e1x", len) == 2 && len == -999); len = -999; // so we can detect whether navigateRoute sets len assert(navigateRoute(2,4, 1,1, "w3n1", len) == 3 && len == 1);
    - to check if the function can return correct values and steps in cases that have invalid characters in the routes or different steps in certain dir
      * len gets set to value specified