Notable obstacles:

The hardest part of the project was understanding what the specs were asking for. I initially had to read the project 3 spec several times in order to get an idea for what I had to do. I spent a majority of my time doing the project in the office hours, so I was able to clarify any of my questions about which loops would be preferable to use and about different implementations of achieving what I wanted the project to do. Another great difficulty was that I continually got my code checked because I was unconfident about it. I had to also learn how to do step-by-step debugging. This project took an enormous amount of time and dedication, especially because I had no previous experience in computer science. I attended office hours for 4 days, which amounted to around 20 hours.

Description of the design of my program (Pseudocode):

Function to test if path is well formed;

…

repeatedly:

check first character

if digit,

check second character

if digit,

check third digit

if digit,

false

if letter,

check for north, south, east, west

if letter,

check for north, south, east, west

if letter,

false

…

Function traverse segment

…

check start position

check steps is less than max steps,

introduce new function to update location

make sure (r,c) is on grid

make sure number of steps is valid

check letter is lower/upper case N,S,E,W

ensure max steps is greater than or equal to 0

…

Function to change char to int

Introduce function to parse segment

String segment

If character is digit

Assign it to char c

Make the char into an int

Repeat

Assign the letter to direction

Return true

Function traverse path

…

Check for valid starting point and well-formed path

Initialize digit, dir, nsteps

Repeatedly:

Create empty string (segment)

If character is a digit

Add it to the segment and go to next character

If character is digit

Add it to segment and add the next letter to segment

If character is letter

Add it to segment

Parse the segment

If traverse segment is equal to n steps

Add the digits to nsteps

If not, add the number of valid steps to nsteps

If starting point ends at the ending point,

Return 0

…

Test cases:

if(isPathWellFormed("2N1e01E0n2s1e"))

True – I tested this to confirm with the example test given on the website.

if(isPathWellFormed("N1e01E0n2s1e"))

False – I tested this because the first character in the string is a letter, not a number.

if(isPathWellFormed("22N1e01E0n2s1e"))

True – I tested this because the first two characters are both digits, which is synthetically valid.

if(isPathWellFormed("229N1e01E0n2s1e"))

False – I tested this because the first three characters are digits, which is not valid.

if(isPathWellFormed("2N1eee01E0n2s1e"))

False – I tested this because there are 3 letters in a row; there cannot be more than 1 letter consecutively.

if(isPathWellFormed("2N1e01E000n2s1e"))

False – I tested this because there are 3 digits in a row, which is not valid because the maximum allowed is 2 consecutive digits.

if (traverseSegment(3, 1, 'N', 2) == 2)

True – I tested this to confirm with the example test given on the website.

if (traverseSegment(3, 1, 'N', 4) == 2)

True – I tested this because I could only go 2 spaces north so if I was asked to go 4 steps north, it would return the max steps.

if (traverseSegment(1, 3, 'E', 3) == 0)

True – There’s a wall east of (1,3) so it cannot move any spaces to the east, which means it will move 0 spaces.

if (traverseSegment(3, 3, 'S', 2) == 0)

True – I tested this because anywhere south of (3,3) on a grid of (3,4) would be out of the grid so it cannot go any spaces to the south.

len = -999; // so we can detect whether traversePath sets len

if (traversePath(3,1, 3,4, "1ex", len) == 3 && len == -999)

True – I ran this because it was one of the tests on the site that would show if traversePath was working correctly. My program was able to return that the path was invalid.

len = -999; // so we can detect whether traversePath sets len

assert(traversePath(3,1, 3,4, "2N1e01E0n2s1e", len) == 0 && len == 7);

True – I ran this because there were situations in which there were 2 digits such as “01” and I had to make sure that my code was still making this a segment when splitting the path.