Calvin Liu

Report

804182525

CS31 TA: Nick Lecture 2 Disc 2C

1. Obstacles –

-My biggest problem for this project was getting the computer to read the characters 1 and 2 as 12. I had no idea that you could change it by referring to the ascii code.

-Another obstacle was solving part 1 of the bool statement. I tried really convoluted stuff and it kept failing certain tests. Instead of looking for specific failures, I was told to just look for what would work and then make everything else, false.

-It also took a long time to figure out how to segment characters into a command that the computer could understand. If given a direction and steps to take, it was hard to make the computer take in those commands and do them and then take in another command. It was also hard to figure out for the bool statement to make a loop where it only checks every segment and notice the difference between a 3-character segment and a 2-character segment.

-Keeping track of variables, copying them, creating new variables and modifying their copies was also hard to keep track of.

1. Pseudocode -

Check each segment of the given movement to see if there are any errors

If the first character of a segment is a number

Check every 2 characters.

Return false if it reads only 1 character at the end

The second character better be one of the directions

If the second character of a segment is a number, read ##L

Check every 3 characters.

Return false if it reads only 1 or 2 characters at the end.

The second character better be one of the directions

If it is correct:

See if your maxSteps is negative

See if you’re starting off the grid

See if you’re starting at a wall

You took 0 Steps

While the amount of steps you take is less than maxSteps, all the time, increase the amount of steps you take by one, and change r and c by 1 depending on the direction.

If you run off the grid, return the number of steps you took

If you run into a wall, return the number of steps you took

If it hits nothing, return the amount of steps you took which is the maxSteps

If you’re starting at a wall, ending at a wall, starting off the grid, ending off the grid, or if your input made no sense syntactically, return 3

Copy the starting position into another thing called current \_\_\_ so you can modify current \_\_\_ once you start to move.

The steps you’re attempting to walk and the total number of steps finished should begin with 0

If the number read in is 2 digits

Read in the number the 2 digits make and put it into the attempted steps to walk, read the direction and move onto next segment

Read in the number 1 digit makes and put it into the attempted steps to walk, read the direction and move onto the next segment

If the steps attempted is equal to the steps possible

then use the direction to change the current \_\_\_ based on the direction and add the total number of steps and store it into nsteps

else, if there is a wall, add the number of steps completed without a wall to the number of steps took until you hit the wall and return 2

if the current row/column that you’re at does not equal the ending position, then you missed the end and return 1.

Otherwise, return 0.

1. Testing
2. assert(isPathWellFormed("2n1e"));
3. assert(isPathWellFormed(“15n8e));
4. assert(!isPathWellFormed("1ex"));
5. assert(!isPathWellFormed("n12"));
6. assert(!isPathWellFormed("123N"));
7. assert(!isPathWellFormed("12e4ss"));
8. assert(!isPathWellFormed("45t30"));
9. assert(!isPathWellFormed("45e2"));
10. assert(!isPathWellFormed("x23n"));
11. assert(!isPathWellFormed("2x13n"));

The first two tests are the normal scenarios and include any sort of combination.

#3 tests if it’s a letter, letter.

#4 tests if the first character is a letter

#5 tests to see if there are 3 digits in a row

#6 tests if two letter are next to each other later in the program

#7 tests if there is a character that does not represent a direction

#8 tests if it is syntactically valid, but not possible

#9 tests to see if the first letter is not a number and it’s not a cardinal direction

#10 tests if there is an invalid character with only 1 digit at first

1. assert(traverseSegment(21, 5, 'N', 2) == 2);
2. assert(traverseSegment(3, 25, 'N', 2) == 2);
3. assert(traverseSegment(3, 1, 'X', 2) == 2);
4. assert(traverseSegment(-21, 1, 'N', 2) == 2);
5. assert(traverseSegment(3, -25, 'N', 2) == 2);
6. assert(traverseSegment(3, 1, 'N', 5) == -2);
7. assert(traverseSegment(3, 1, 'N', -2) == 2);
8. assert(traverseSegment(3, 1, 'N', 5) == 2);

#1 tests to see if the starting row is off the grid

#2 tests to see if the starting column is off the grid

#3 tests to see if the direction is not one of the cardinal letters

#4 tests to see if the starting row is negative

#5 tests to see if the starting column is negative

#6 tests to see if the steps you are taking is equal to a negative number

#7 tests to see if the the maxSteps is negative

#8 tests to see if the number of steps is not equal to maxSteps

All the tests that could be used for returning 3 work such as:

* if the direction starts with a letter
* If there is 3 letters next to each other
* If there is 2 characters next to each other
* If there is a wall at the starting position
* If there is a wall at the ending position
* If it’s not syntactically valid
* If you’re starting off the grid

All the tests that could be used for returning 2 work such as:

* If you run into a wall or off the grid and output the correct number of steps that you took
* Ex) if (traversePath(1,3, 3,1, "2S2w", len) == 2 && len == 2) with a wall at (3,2)

All the tests that could be used for returning 1 work such as:

* Having sr not equal to er or sc not equal to sr means that it missed the end

If it reaches the end then it will equal 0.

Ex) if (traversePath(1,3, 3,1, "2S1e", len) == 1 && len == 3)