Report

a. A brief description of notable obstacles you overcame:

The isWall function requires the input to be a valid grid. For example, in the second function, if c or r is larger than the size of the maze set by the function setgrid, isWall function will result in undefined behavior. This often causes the program to crash. However, the maxsteps takes the robot to a grid out of the allowed range, we want the program to return the maximum steps the robot can take before falling off the edge rather than crash entirely. To achieve this goal, I wrote the “if condition” as at first

if (!isWall (r + i + 1, c) && (r + i + 1) <= getRows())

This line is wrong, because the behavior in of isWall function is undefined and causes the program to crash. Then I made the following change

if ((r + i + 1) <= getRows() && !isWall (r + i + 1, c))

, which allows the program to compile.

b. A description of the design of your program.

Is Route Well Formed:

If the first character is a digit, return false

Otherwise

Visit every character of the string

Characters that are neither digits nor letters, return false

Invalid direction characters, return false

Movement more than two digits, return false

Otherwise return true

Navigate Segment:

If the input is invalid, return -1

Otherwise

Convert all directional characters into lowercase

If north

Move north and return maximum number of steps

If south

Move south and return maximum number of steps

If west

Move west and return maximum number of steps

If east

Move east and return maximum number of steps

If the robot does not reach invalid grid, return the input number of steps (maxsteps)

Navigate Route:

Check whether the inputs are valid

If valid

Repeatedly

Find the first letter of in the string

Check how many digits are after the letter

Covert digits to steps

If robot does not run into a wall

Add number of steps to overall steps

Otherwise

End the program(return 3)

Move robot according to the new starting point based on instruction

Check whether the robot ends up at designated end point

If it does not

End the program (return 1)

If it does (the route is complete)

End the program (return 0)

If invalid

End the program (return 2)

c.

int main ()

{

setSize(3,4);

setWall(1,4);

setWall(2,2);

setWall(3,2);

int len;

len = -999

}

Test “isRouteWellFormed” Function

Input expected output

3sn False first character can’t be digit

elx False invalid directional letter

N144 False movement larger than 2 digits

w2+n3 False invalid character

N2eE01n0e2e1 True

W42 True

All of the tests above are passed

Test “navigateSegment” Function

Input Expected output

(3, 1, 'N', 2) 2 robot does not run into wall or fall off the edge

(5, 1, 'N', 2) -1 invalid grid

(3, 1, 'h', 2) -1 invalid letter for direction

(3, 1, 'N', -3) -1 maxSteps is negative

(3, 1, 'N', 3) 2 robot falls off the edge

(3, 1, 'E', 2) 0 robot runs into the wall

All of the tests above are passed

Test “navigateRoute” Function

Input Expected output (navigateRoute) (len)

Complete route

(3,1, 3,4, "N2eE01n0s2e1", len) 0 7

Invalid character

(3,1, 3,4, "elx", len) 2 -999

Starting point is a wall

(3,2, 3,4, "N2eE01n0s2e1", len) 2 -999

Starting point out of range

(5,5, 3,4, "N2eE01n0s2e1", len) 2 -999

End point is a wall

(3,1, 1,4, "N2eE01n0s2e1", len) 2 -999

End point out of range

(3,1, 6,6, "N2eE01n0s2e1", len) 2 -999

Robot does not end up at designated end point

(3,1, 1,1, "N2eE01n0s2e1", len) 1 7

Robot runs into a wall

(3,1, 3,4, "N2e2s1w1", len) 3 5

Robot falls off the edge

(3,1, 3,4, "N2e2s1e3", len) 3 6

All tests are past.