The logics part of the function determinSafeDistance and obeyPlan is hard. I used pseudo code and debug mod of gdb to see the variables in a loop. Also, I didn’t think about the case that number of steps can be two digits. I fixed that using some time.

hasCorrectForm

Define counter

Repeatedly

Read characters in the string

if is a correct character

do something to counter

else

return false

if the end is not a string

return false

else

return true

determinSafeDistance

make sure the car is in the grid and the point is not a wall, numb of steps is larger than 0

read direction

if some direction

if maxSteps is larger than valid steps

return valid steps

else return maxSteps

\*do that for four directions

obeyPlan

make sure all conditions from spec are met

define counter and steps

if some direction

repeatedly

read from plan

change direction according to plan

determine safe distance

compare safe distance and steps in plan

if safe

move coordinates

else

return error

\*do that for four directions

setDirection

if dome direction

if left

set directions

if right

set directions

\*do that for four directions

// testing function hasCorrect form

//Testing correct form

assert(hasCorrectForm("2R1r"));

// testing plan with non-letter ending.

assert(!hasCorrectForm("L2"));

// testing plan with non-valid character

assert(!hasCorrectForm("1Rx2L"))

// testing plan with 3 digits of integers

assert(!hasCorrectForm("144R"));

// testing with non-valid character

assert(!hasCorrectForm("1R+2L"));

// testing plan with 3 digits of integers && non letter ending

assert(!hasCorrectForm("N144"));

// testing determineSafeDistance

100110010010001

001001000000100

100100010010010

110000101010010

101010010101010

010101010011001

// N to bound of grid in safe distance

Assert(determineSafeDistance(3,2,’N’,2) == 2)

// when can’t move a step

Assert(determineSafeDistance(1,2,’N’,1) ==0)

// when input distance is larger than safe

Assert(determineSafeDistance(3,2,’N’,5) == 2)

//N to wall in safe distance

Assert(determineSafeDistance(3,3,’N’,1))

//when input distance is larger

Assert(determineSafeDistance(3,3,’N’,3))

// when can’t move a step

Assert(determineSafeDistance(3,3,’N’,1))

// testing obeyPlan

// correct plan

int len = 999;

assert(obeyPlan(2,1,5,15,'N',"R1r1l1r1L3l1R1l1r5l1R2r1l1R3R",len) == 0 && len == 23);

//hitting boundry

assert(obeyPlan(2,1,5,15,'N',"R1r1l1r1L3l1R1l1r5l1R2r1l2R",len) == 3 && len == 20);

// hitting wall

assert(obeyPlan(2,1,5,15,'N',"R1r1l1r1L3l1R1l1r5l1R2r1l1R4r",len) == 3 && len == 23);

// incorrect plan

len = 999;

assert(obeyPlan(2,1,5,15,'N',"R1rxl1r1L3l1R1l1r5l1R2r1l1R3R", len) == 2 && len == 999);

// starting point is wall

assert(obeyPlan(1,1,5,15,'N',"R1r1l1r1L3l1R1l1r5l1R2r1l1R3R",len) == 2 && len == 999);

// ending point is wall

assert(obeyPlan(2,1,6,15,'N',"R1r1l1r1L3l1R1l1r5l1R2r1l1RR",len) == 2 && len == 999);

// not getting to the end point

assert(obeyPlan(2,1,5,15,'N',"R1r1l1r1L3l1R1l1r5l1R2r1l1R2R",len) == 1 && len == 22);