* The two main obstacles that I overcame were making sure that the value of *maxSteps* passed to the driveSegment() function was accurate and making sure that the value of *nSteps* was accurate even with the pass by reference to the driveCourse() function. The first problem occurred because I was trying to take a number from a string and treat it as an integer of that value without taking into account its unique character code. So when the integer variable *max* was used in the driveCourse() function, I was getting a value of 51 instead of 3. To solve this, I subtracted the character code for ‘0’ from *max* since the character code of 0-9 is contiguous and increasing. I was alerted to the second problem when my program failed three of the test cases in the g31 compiler. The issue was that I was using the conditional phrase (isalpha(x) == true) instead of just (isalpha()). Xcode did not pick up this error, but g31 did. Once I realized this, the fix was simple and my program passed all six test cases.
* The design of my program is according to the project specifications -- I use three functions, with one of them calling the other two, to systematically decide if a given path is valid for a car to take in a given grid, with certain starting and ending coordinates. The pseudocode for the three main functions follows:

isCourseWellFormed():

1. If *course* is an empty string, then return true.
2. If the first element in *course* is a number, return false.
3. repeatedly:
   1. Visit each character. If the character is not a cardinal direction (N, S, E, W) in upper or lower case or a digit, then return false.
   2. If the element is a number and the next two elements are a number as well, then return false.
4. If we’ve navigated through the string with no problems, return true.

driveSegment():

1. Make sure there is no wall at coordinate position.
2. Make sure coordinate position is on the grid.
3. Make sure direction is a cardinal direction (N, S, E, W) in either upper or lower case.
4. Make sure number of steps to travel is positive
5. Depending on the direction, repeatedly:
   1. visit the grid position in the given direction and up to the given number of steps. increment a counting variable each time each visit is successful.
   2. if a grid position is a wall or off of the grid, return the counting variable.
   3. if we navigate through the number of steps fully, return that number.

driveCourse():

1. make sure course is valid; if it isn’t, return 2.
2. make sure the starting and ending coordinates are not walls and that they’re on the grid. If not, return 2.
3. repeatedly:
   1. visit each element in the string.
   2. if the element is a cardinal direction, check if there are numbers following it. If there are 1 or 2 numbers, store this value to pass to driveSegment() later. If no number, the default value is equal to 1.
   3. Call the driveSegment() function and store the return value. Compare this value to the value passed to the driveSegment() function. If the returned value is less than the passed value, return 3. If not, proceed to update the coordinates of the car based on how far it traveled.
   4. Do this until we have traversed the entire string or we’ve hit the “return 3” statement.
4. If the updated coordinates are equal to the given end coordinates, return zero. If not, return 1.

* A list of the test data that could be used to thoroughly test my program is as follows:

*setSize(3,4); setWall(1,4); setWall(2,2); setWall(3,2);*

*assert(isCourseWellFormed("n2e1")); //valid command, should return true*

*assert(!isCourseWellFormed("e1x")); //invalid, should return false*

*assert(driveSegment(3, 1, 'N', 2) == 2); //asking function to make car go 2 units north. should succeed and return value of 2*

*int len = -999; // so we can detect whether driveCourse sets len*

*assert(driveCourse(3,1, 3,4, "N2eE01n0s2e1", len) == 0 && len == 7); //function should navigate from start to end point successfully. len gets 7 and function should return 0.*

*len = -999;*

*assert(driveCourse(3,1, 3,4, "e1x", len) == 2 && len == -999); invalid course string, should return 2 while leaving len unchanged.*

*len = -999;*

*assert(driveCourse(2,4, 1,1, "w3n1", len) == 3 && len == 1); can only navigate 1 unit west, so return 3 and len gets 1.*

*setSize(2,5); setWall(2,2);*

*assert(driveSegment(2, 5,'w', 4) == 2); //valid segment but car hits obstruction and can only travel two steps. thus, return 2.*

*len = -999;*

*assert(driveCourse(2,5, 1,3, "w1n1", len) == 1 && len == 2); //valid course, but ends at a coordinate that is not the end coordinate. should return 1 and set len to 2.*

*len = -999;*

*assert(driveCourse(2,5, 1,3, "w2n1", len) == 0 && len == 3); // valid course and ends at end coordinate. should return 0 and set len to 3.*

*len = -999;*

*assert(driveCourse(2,5, 1,3, "w4n1", len) == 3 && len == 2); // valid course but obstruction causes car to stop after two steps. should return 3 and set len to 2.*

My program handles all of the above test cases correctly.