How many shortest paths?

In this problem you are supposed to implement a recursive function that determine the number of shortest paths between two points on the Cartesian axis.

For example, if the start point is (1, 0) and the end point is (4, 1), there will be four shortest paths, which are

(1,0)->(1,1)->(2,1)->(3,1)->(4,1)

(1,0)->(2,0)->(2,1)->(3,1)->(4,1)

(1,0)->(2,0)->(3,0)->(3,1)->(4,1)

(1,0)->(2,0)->(3,0)->(4,0)->(4,1)

You are to implement the count\_shortest\_paths function in problem x. The declaration of the function is given below,

int count\_shortest\_paths(int start\_x, int start\_y, int end\_x, int end\_y)

which takes in the starting x and y coordinates, ending x and y coordinates and returns the number of shortest paths between these two points. To make your live easier, you may assume the end point will always locate above the start point.

You are given

paths.c, which contains the count\_shortest\_paths function for you to implement.

paths.h, which contains the function prototype. Do not modify this file.

test\_path.c, which contains a main function for you to test your function. You should add more test cases to ensure your function works for all cases.

Makefile, in order for you to test easily, type in “make” to compile and build to program as you did in the MPs. Do not modify this file. To run your program, type in “./paths”.

Please be noted, you must use recursion to solve this problem. Any other method (such as iteration or calculating using some formula) will not be accepted and you will earn zero point. Only your function will be graded.