## Sample Program #2 (Tree House Walking) Testing Strategy

My first test case in the sample involved 1 pair of points. Presumably, if this works, no other test case with n = 1 is necessary. My first new test case beyond the sample just has 2 pairs of points. I created a corner case where two pairs of points are very, very close to each other, but the pairs are far apart from one another. This way, I knew what the answer should be. (The points I picked were (10000, 10000), (-10000, -10000), (-10000, -9999) and (9999, 10000). In this case, the answer is 2.) To be thorough, I tried 3 different orderings of those points, where the first matches the second, the first matches the third, and the first matches the last one.

Next, I wanted to make some test cases where n = 4. One simple test case would be to put all the points on a line, but space out the ones that should be matched farther from one another, so that a greedy strategy of matching the closest pair of points would fail: (0, 0) (100, 0), (101, 0), (201, 0), (202, 0), (302, 0), (303, 0), and (403, 0). I randomized the order these points were listed in the input. The answer for this case should be 400, since the four consecutive segments on the line each have length 100.

Another nice arrangements of points is in a 2 by 4 rectangle set up. We can make the rectangle really tall or really short, so to speak to create different matchings, visually. I picked the points: (-10000, 0), (-10000, -1), (-5000, 0), (-5000, -1), (0, 0), (0, -1), (5000, 0), and (5000, -1). Again, I randomized the order these points were in the input. Of course, the answer here is 4. For the second arrangement, I picked the points: (3,-10000), (3, 10000), (5, -10000), (5, 10000), (6, -10000), (6, 10000), (9, -10000), and (9, 10000) and rearranged their order. This should have an answer of 10. (Two segments of length 2, two segments of length 3.)

Finally, I created two cases with n = 5 where the answer was not integers. One of the cases has randomly selected points (but just ranging from 0 to 100 for each coordinate) and the other case looks like tracing a mountain. For the former case, a visual inspection was done to approximate the real answer and for the latter case, the mountain was created with a clear correct answer. Also, all of these cases were verified with an unoptimized solution which just tries all permutations of points to verify correctness.

For n = 6, n = 7, and n = 8, similar strategies were used to create a few of the test cases and the rest of the test cases were randomized. Ultimately, once getting to making these cases, there has to be some confidence that the program is indeed trying all possible pairings and is running correctly. (A program in C that generated random cases is posted separately.)