Good job! You've almost isolated the problem. Narrowing it down from "somewhere in the program" to a specific function is great.

I view debugging as a problem in finding a needle in a haystack. The needle is the line (or lines) that do not have the intended results, and the haystack is the collect of all lines in the program. My strategy is usually similar to dividing the haystack into several smaller stacks and using a metal detector to find which of the smaller stacks has the needle. Then, take that smaller stack and divide it into even smaller stacks, using the metal detector to again identify the smaller stack that has the needle. If I repeat this process, eventually my division into smaller stacks will be separating single stalks of hay and the needle.

Since you've identified a function that is the problem, you've already done a couple of the "split the haystack up" steps. Now, this function probably has 5-10 lines of code. Each is potentially a hay stalk or a needle. There might be multiple needles.

Now, I would want to see what each line of code does as the function processes its input parameters. One simple way to do this is to add print() statements in function to display the values received by the function, and the values of each variable as it changes. Then, run the function. This is usually accomplished by running the unit tests. You should be able to find the output of the print() statements in the output from the unit tests every time the function is called.

Pick one set of output lines that correspond to the execution of the function when it gives an incorrect result. Look at the output and the function's source code at the same time to observe where each value came from. If possible, identify the first place where a value isn't what it should be. The line that produced that incorrect result is your first needle. Now all you have to do is analyze that line and the other lines in its neighborhood to decide how to get the desired result.

I often find that students feel that this debugging process is too much work, and takes too much time. So, they spend their time staring at the function trying to figure out what's wrong with it, and ultimately get very frustrated. So, don't be one of those students. A major part of programming is verifying code that has been written works correctly, and fixing those things that aren't correct. If you don't develop a process for identifying and fixing errors, you'll be often be frustrated. There are better debugging techniques than print() statements, but they all rely on observing the behavior of code, and you don't have to learn anything new to use print().

Ask me about "True Laziness" sometime.