The first step I did was compile the original sort.c program with the following flags: -Wall -pedantic –ansiA screenshot of a computer

Description automatically generated

To fix this, I created a copy of the original program into a new file called “mod\_sort.c”. Under the mod\_sort.c file, I removed the comments to get rid of the C++ style comments warning. A screenshot of a computer

Description automatically generated with medium confidence

After removing the bugs, I created a function for printing out the array of items to see if the bubble sort function properly sorted the array. The following is my print function:

Text

Description automatically generated

Next, I re-compiled and executed the program to see if the array was sorted. These are the results that I got:

A screenshot of a computer

Description automatically generated

As shown in the pic above, there are no compile errors with the pedantic, Wall, and ansi flags, however, the array is not being properly sorted. The order of the array should be in ascending order. To help me figure out what is happening with the bubble sort function, I re-compiled but added -g flag to indicate that I will use the executable with GDB. Here is my compile command: A screenshot of a computer

Description automatically generated with medium confidence

After running the compiling command above, I then proceeded to using the “Debug” executbale with GDB. I got the following output:

Text

Description automatically generated

As shown above, GDB was not able to detect any bugs such as a memory leak with the program. Next, I tried using Valgrind to see if it could provide any insights on what is wrong with the program. This is my valgrind output:

Text

Description automatically generated

As shown above, Valgrind detected that there is an issue of a conditional jump or move that depends on unitialized values. After researching how to interpret this, I discovered that if I add the following option to my valgrind command:  **--tool=memcheck --track-origins=yes**, Valgrind will tell me exactly where the conditional jump issue is occurring. After running valgrind with those options I got the following output:

Text

Description automatically generated

This output above provided me with more information, specifically it shows me that there is an uninitialized value that is being created by a stack allocation on line 17 of the sort method. Knowing that there is an initialized value being created on the stack, I looked at the sort method (variables on the stack come from local variables of a function):

Text

Description automatically generated

I see that the sort method has a variable **s** that is uninitialized and it seems like it is unnecessary to be in the sort function altogether. Then I removed the **s** variable and left with the following sort method:

Text

Description automatically generated

I recompiled my program and then re-ran the same Valgrind command that I used before. I did not get any errors as shown below:Text

Description automatically generated

Although I did not have any errors as show above, the sort program was still not sorting correctly. After researching bubble sort, I see that the issue is in the for loop conditions. The loop conditions were set to iterate **n** times, but this is incorrect. It should instead be **n -1** times for the first loop. And the second for loop should iterate until **n – i – 1** times. My sort function is now the following:

Text

Description automatically generated

I recompiled my program afterwards and got no errors. I then executed the program and now I have the correct result:

Text

Description automatically generated