2.1:

The first few values of the array are 2, 3 and 5 which are the lowest three prime numbers. They are also primes lower than 25. The rest of the values are zeros as the array has not been filled with values by the sieve.c program: the program hasn’t run that far yet. The next output is 2,3,5,0. This occurs because the breakpoint is at the part of the program where the primes array is updated, meaning that it hasn’t been updated yet. Then, when the breakpoint is cleared, 11,13,17,19 and 23 are outputted.

2.2:

There are a few problems in this program, the most visible that the entire number is being multiplied by 0 (n is decremented to zero). To fix this one should replace the while loop with a for loop which would prevent n from decrementing to 0 and ensure that the result is properly multiplied by n.

2.3:

There seems to be an arithmetic exception on line 11, caused by division by 0.

2.4:

The first modification was the removal of the ampersand from the front of the parameter of scanf. This caused a segmentation fault. To identify this, a breakpoint was set at the line where the segmentation fault occurred. When the value of the variable was outputted, it showed a memory address which indicated the bug that was present.

The other modification was a replacement of the equality comparison “==” with an assignment statement that made the if statements permanently true. This was extremely difficult to detect by printing variables as the variables were themselves correct, the logic used to compare them was just incorrect. Eventually, the code was simply checked manually (or visually) for bugs and the error presented itself.

2.5

There are numerous debugging tools available for linux. For example, one may use gdb enhancers, memory-corruption detectors, profilers, and memory-leak detectors. Additionally, there are both open-source and paid tools. An example of the latter would be Affinic 0.5.3 which is a commercial GUI wrapper for GDB which costs $49. On the other hand, KDevelop 4.3 is an open-source Ide that is fairly simple but still apparently effective.

Valgrind is an example of a free debugging tool for Linux. It started as a free memory debugging tool but is now a generic framework for additional tools. When running Valgrind, it will clearly indicate if the number of calls to free and malloc is different which would indicate if there is a member leak due to a call to malloc in main. Additionally, the memcheck tool can be used to determine if invalid pointers have been used. Finally, Valgrind can determine if uninitialized variables have been used. These are all ways of detecting potential memory leaks and they are made much easier to detect by Valgrind

Conclusion

This lab accomplished the task of introducing the process of debugging programs in addition to introducing us to a series of tools used to debug programs. The exercises that we encountered were all fairly simple which meant that they were generally easy. Although the debugger was a bit intimidating, once the basic functionality was worked out the rest of the lab went smoothly. One interesting occurrence was the relative uselessness of printing out variables for the second bug that was introduced to our program. Further understanding and familiarity with the debugging tools available will most likely eliminate this problem. All in all, we found this lab to be extremely informative due to the introduction to debuggers and the valuable practice that we gained in their use.