**Module 5 Option 1 Program Analysis**

James Strauser

Colorado State University Global

CSC 450

Jul 2, 2023

For this program analysis I am supposed to discuss performance issues with concurrency, vulnerabilities exhibited with use of strings, and security of the data types exhibited. I will be doing these in no particular order, starting with the discussion of strings… There is no use of strings in this program. That by itself is mostly due to me being minimalist in the coding. I could format the output or put a printou statement that says when the program is finished, but neither of those are particularly necessary so I did not use them. Even those scenarios however do not introduce any vulnerabilities, vulnerabilities with strings really only exist when the user has access to the system but that is not part of the program so strings are very secure for this program. Next is the security of the data types exhibited, there are two maybe three. First is the int used as a counter. It is a global variable so it is not very secure but it would be difficult to make it exhibit undefined behavior as the values of 20 and 0 are very far away from overflow values. Next is maybe whether or not threads count as a data type, I don’t think they do. Next is the mutex which only adds security, which I thought I did not even need when first making the program but in actuality there is a case where the program gets stuck in a loop at counting up and down from 19 and 18. This was because the loop in thread one has a sleep after it counted but before the end of the loop so in that time the second thread would decrement the counter and the first thread would never end as intended. This also happened when the mutex was unlocked before the sleep command but simply adjusting the timings fixed that. That is also one of the main performance issues one can encounter with concurrency. I think it would count as a form of a deadlock even though it is not technically the same. There are also the risks with threads of simply not having enough of them, though in this case just two threads is usually allowable but if there was enough threads, things would lose concurrency. There is also the fact this takes up more memory since multiple things are being used at once but are sharing the same memory space, again not an issue here but it is much more of an issue in larger programs so limiting the memory size of each individual thread is essential.