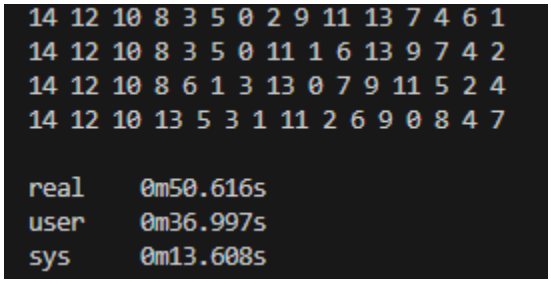
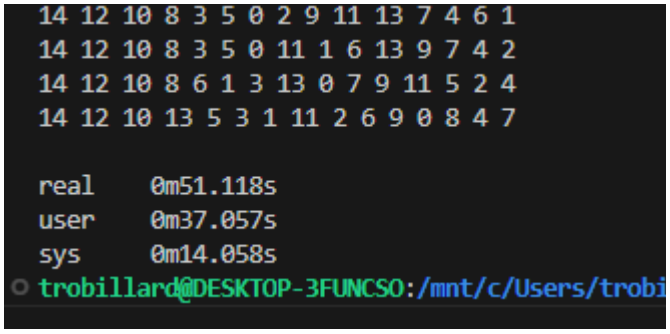
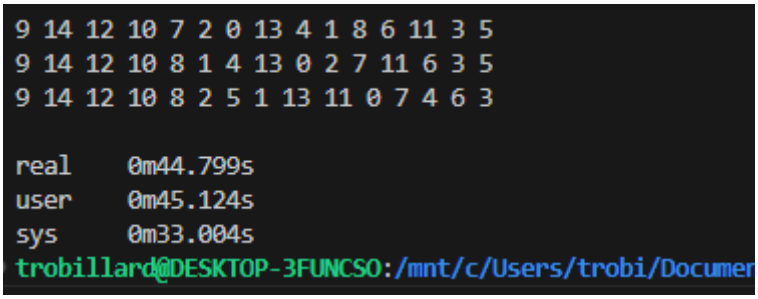
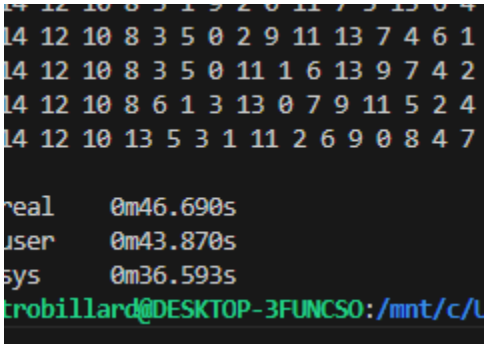
N Queens (for N = 15)						
	Sequential	1 thread	2 threads	3 threads	4 threads	OpenMP
Time (s)	50.492	50.616	51.118	44.799	46.690	25.744
N Queens (for N = 15) Without Printing						
	Sequential	1 thread	2 threads	3 threads	4 threads	OpenMP
Time (s)	33.209	33.756	24.177	16.551	13.207	15.390

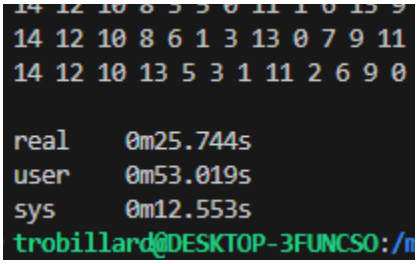
The recorded times for each additional thread was not reduced very much when using pthreads, and I had a feeling that the mutex lock surrounding the print section was severely affecting runtime, as the OpenMP variant was more than twice as fast than the 4 threaded program using pthreads. Also, I noticed that 4 threads was actually slower than using 3 threads, which didn't make a whole lot of sense. I assumed that the printing was the main cause of this again, so I ran a set of additional tests without printing the results. These second group of times show that this original assumption was correct. It seems that OpenMP does a very good job managing critical sections in a timely manner, as opposed to manually controlling them with the mutex.








14 12 10 8 3 5 0 2 9 11 13 7 4 6 1
14 12 10 8 3 5 0 11 1 6 13 9 7 4 2
14 12 10 8 6 1 3 13 0 7 9 11 5 2 4
14 12 10 13 5 3 1 11 2 6 9 0 8 4 7
real 0m46.690s
user 0m43.870s
sys 0m36.593s
trobillard@DESKTOP-3FUNCSO:/mnt/c/U


14 12 10 8 6 1 3 13 0 7 9 11
14 12 10 13 5 3 1 11 2 6 9 0
real 0m25.744s
user 0m53.019s
sys 0m12.553s
trobillard@DESKTOP-3FUNCSO:/m