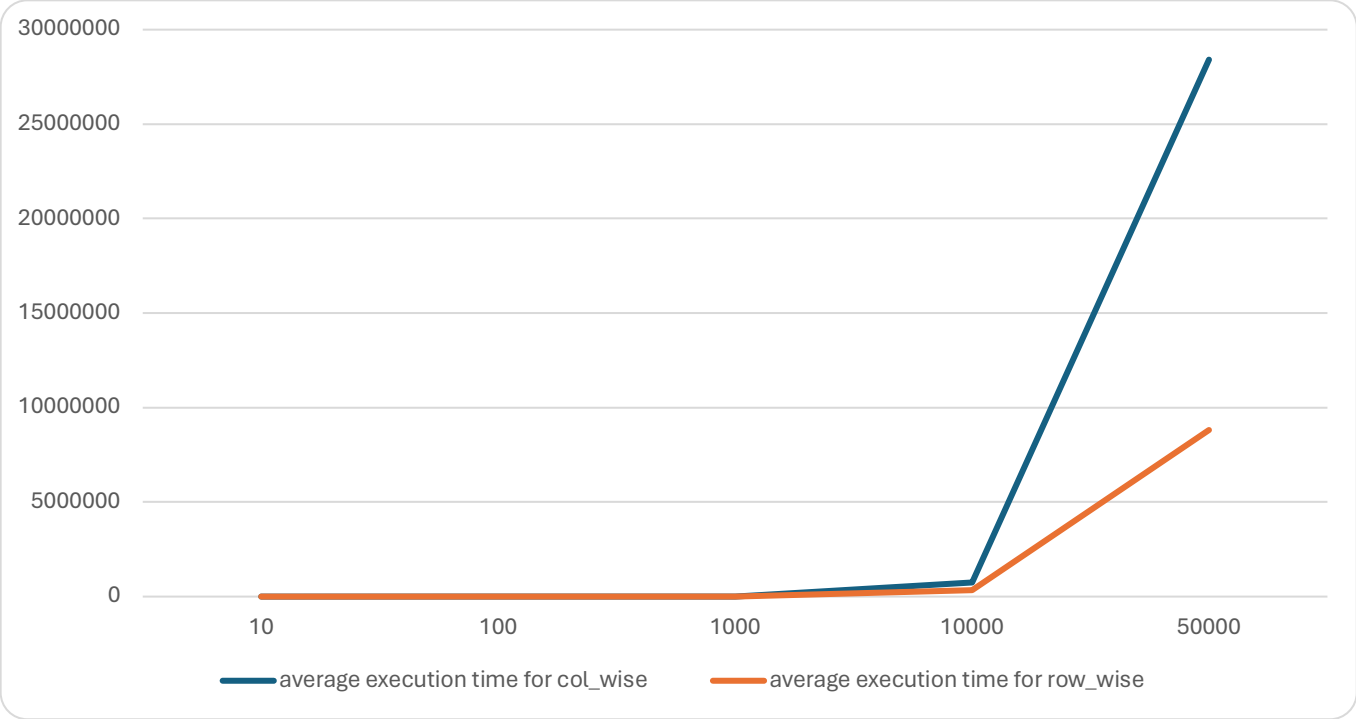
Part 1

# of rows	average execution time for col_wise (microseconds)	average execution time for row_wise (microseconds)
10	1.4 microseconds	1.3 microseconds
100	47.5 microseconds	44.2 microseconds
1000	3594.1 microseconds	3506.7 microseconds
10000	742003.4 microseconds	352758 microseconds
50000	28412959.8 microseconds	8814975.4 microseconds



1. Spatial locality is the concept where items with nearby addresses are referenced closer together in time. Furthermore, within a program if a certain items memory address is accessed, then it is highly likely that the program will also be accessing an item which has an address that is nearby to the first items address. Spatial locality is important in terms of time efficiency and performance.
2. Based on my knowledge of spatial locality, the for loop in row_wise.c would perform faster. Spatial locality is the concept where nearby addresses of addresses are accessed prior to items of memory addresses in a farther location. In this case, the for loop in row_wise.c would perform faster than the col_wise.c because of spatial locality and the fact that the memory addresses of items are closer which is why the row_wise performs faster in terms of time.
3. Based on the plot, we can see that the execution time increases as the array size increases but the execution time for col_wise.c increases at a faster rate than the execution time for row_wise.c and so the row_wise.c for loop is faster. The reason

why row_wise.c is faster than col_wise.c is due to spatial locality which is the cause for faster execution time in row_wise.c compared to col_wise.c.

4. Yes, the answer I gave for 2 is the same answer I gave for 3 which is that the for loop in row_wise.c is faster than the for loop in col_wise.c. The reason why the for loop in row_wise.c is faster than the for loop in col_wise.c is because of spatial locality and accessing the items in row major order would lead to a faster execution time rather than accessing the elements in column major order.
5. A reason why both program behaves the way it does, in terms of execution time, is because spatial locality. As mentioned above, spatial locality is where items with closer memory addresses are accessed sooner. In terms of row_wise.c, we can see that the execution time is faster because the items are accessed in row major order which means that the items right next to each other are accessed first because they have closer addresses in memory.
6. I believe we executed each implementation 10 times in order to get a precise and accurate value to reduce chances of unexpected values and/or outliers. Based on my knowledge in statistics from previous statistics courses, is it good practice to execute a certain task a certain number of times (depending on the task) in order to get a good idea of the execution value and in order to make conclusions.

Part 2

1. The “lscpu” command is meant for providing a list of information regarding to the CPUs in the machine in the format of category name on the left and information output on the right. After executing the lscpu command in the terminal the L1d and L1i cache size is 768 KiB, the L2 cache size is 12 MiB, the L3 cache size is 192 MiB.
2. In order the find the size of the ram we would use the “free -m” command in the terminal. After typing and executing the command we see that the size of the RAM is 515373.
3. Yes, both the RAM and cache sizes have an effect on the execution times. Higher amounts of RAM can have an increase in performance. Larger cache sizes can lead to increase in performance.

Part 3

# of rows	col_wise execution **cache hits**	col_wise execution **cache misses**	row_wise execution **cache hits**	row_wise execution **cache misses**
10	14,019	4,462	13,911	4,501
100	15,235	4,447	15,651	4,468
1000	1,631,922	8,920	239,233	6,040
10000	400,439,705	117,394,726	23,136,277	125,951
50000	7,921,559,831	5,260,804,143	559,447,084	3,123,252

1. In order to get the cache hits and misses I would first compile the col_wise.c and row_wise.c files using "gcc -g -o cw col_wise.c" and "gcc -g -o rw row_wise.c" then execute both executable files using "perf stat -e cache-references,cache-misses ./cw" and "perf stat -e cache-references,cache-misses ./rw" to then display the cache hits and misses.
2. Cache hits indicate that the CPU accesses data that is already stored in cache memory. Cache hits are good because it shows that the data was found in the cache, and it wasn't required to get the data from the RAM which would be slower. Overall, it leads to faster data access. Cache misses indicates that the data was not found in the cache. Due to this occurring, it leads to longer access time and increase in execution time for the program.
3. Spatial locality impacts cache hits and misses by forming how memory is accessed in a program. Programs showing strong spatial locality typically have improved cache performance since they access data stored in close proximity in memory. Due to this, the cache hits increase, and the cache misses decrease. Furthermore, improving the spatial locality is important to improve cache hits.