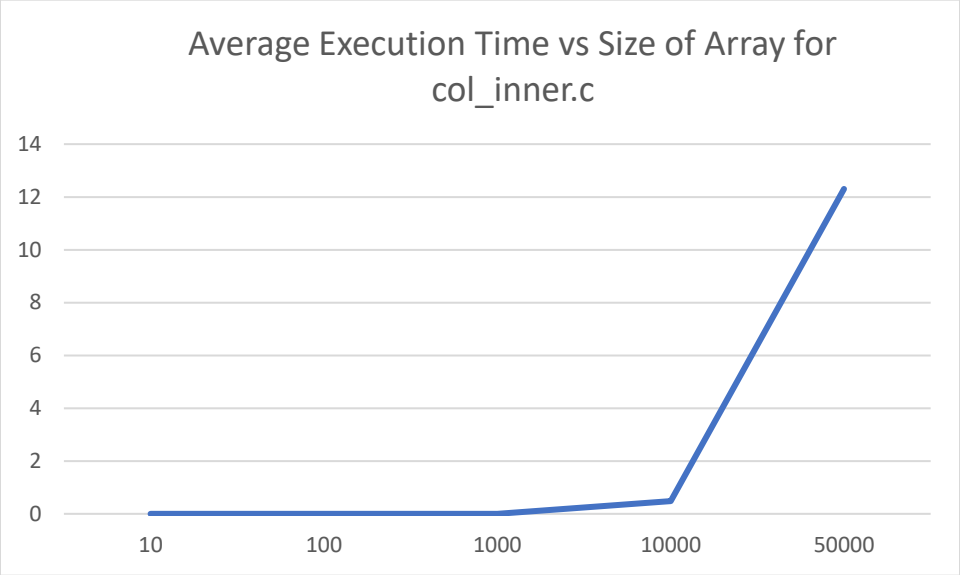
Lab 3

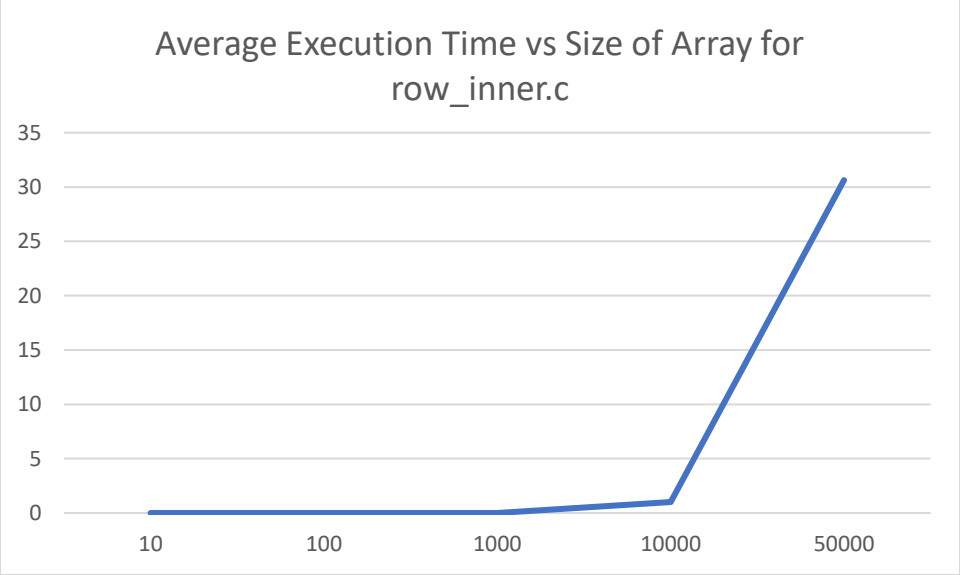
Task 1:

5:

- Based on spatial locality, the nested loop with columns inside, col_inner.c, will perform faster, because 2D arrays are retrieved by row, not column. This is because 2D arrays are stored by one row following another, and column really just acts like an index on each row.

6:





# of rows	Average Execution Time for col_inner.c	Average Execution Time for row_inner.c
10	0	0
100	0	0
1000	0	0
10000	0.483	0.997
50000	12.304	30.629

7:

- Execution time increases when the size of the array the loops are iterating through, like one would expect. Strangely, it does not increase linearly, and in fact, there appears to be no time taken by the function until the array has 10,000 rows and columns, and then it appears to be half a second. And then it grows exponentially from 10,000 to 50,000.
- The prediction was correct, col_inner.c performed better than row_inner.c.
- With arrays of smaller sizes, it may take such a small amount of time that the clock function can not accurately measure the execution time. When trying to figure out this problem, I printed the start and end clocks, both of which read to be 0, so it is most likely because the programs are executed too quickly.
- Each implementation was executed multiple times because the times can vary. The times can vary because most operating systems now are multi-tasking, and the execution time can vary depending on if the program runs into other processes on the computer during its execution, which could affect run time.

Task 2:

- The cache sizes are as follows:
    - L1d cache: 32K
    - L1i cache: 32K
    - L2 cache: 256K
    - L3 cache: 30720K
- The RAM size is 515722.
- Cache sizes influence execution time, because the bigger the cache, the more memory that is immediately accessible by the CPU, which means less time is spent trying to access memory.

Task 3:

To get the cache hits I used:

$perf stat -e cache-references ./executable_name

To get the cache misses I used:

$perf stat -e cache-misses ./executable_name

The cache hits and misses indicate the efficiency of the program. The more misses a program has, the longer it takes because it has to take more time accessing memory.

When a program needs to access data not in the cache, or when its attempt to access data results in a cache miss, it retrieves a spatial locality from the memory, or data that is stored close together. So if a program uses data that is stored close together, like two values next to eachother in an array, the program will have more cache hits, because when retrieving the first element, the program also retrieves the other element from memory.

col_inner		
Size of array	Cache hits	Cache misses
10	3,291	1,818
100	3295	2,196
1000	3,839	2,249
10000	59,653	8,351
50000	7,630,365	72,489

row_inner		
Size of array	Cache hits	Cache misses
10	3,181	2,172
100	3,518	2,260
1000	76,657	7,022
10000	106,341,240	5,171,072
50000	5,353,559,731	483,972,500