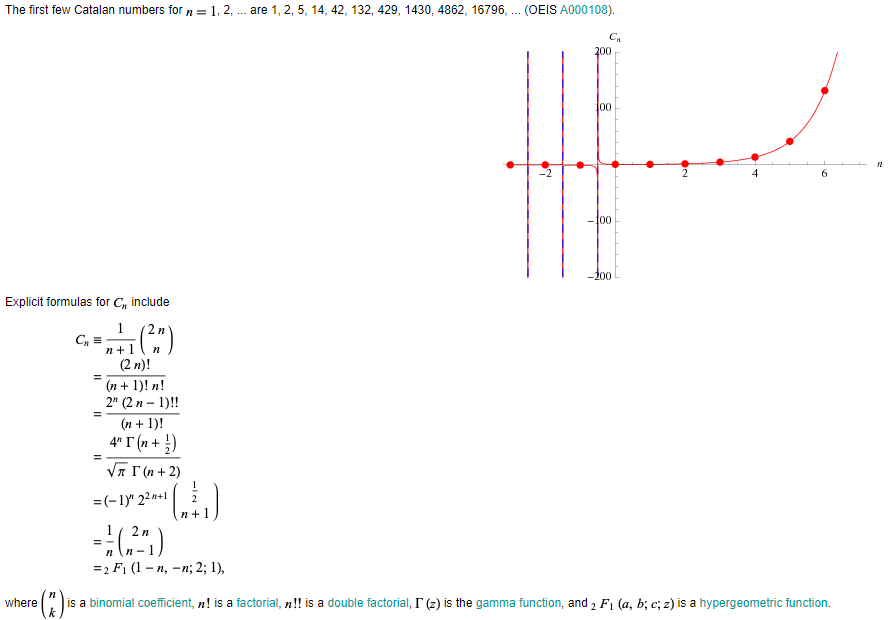
**Review 2**

**Introduction/Motivation:**

I have been working with Catalan Numbers since last semester and is interesting to see how many applications you can use them. The Catalan Numbers are a sequence of positive integers that appear in many counting problems in combinatorics. They count certain types of lattice paths, permutations, binary trees, and many other combinatorial objects. They satisfy a fundamental recurrence relation, and have a closed-form formula in terms of binomial coefficients.



**Progress: Compare CPU vs GPU**

# CPU Implementation

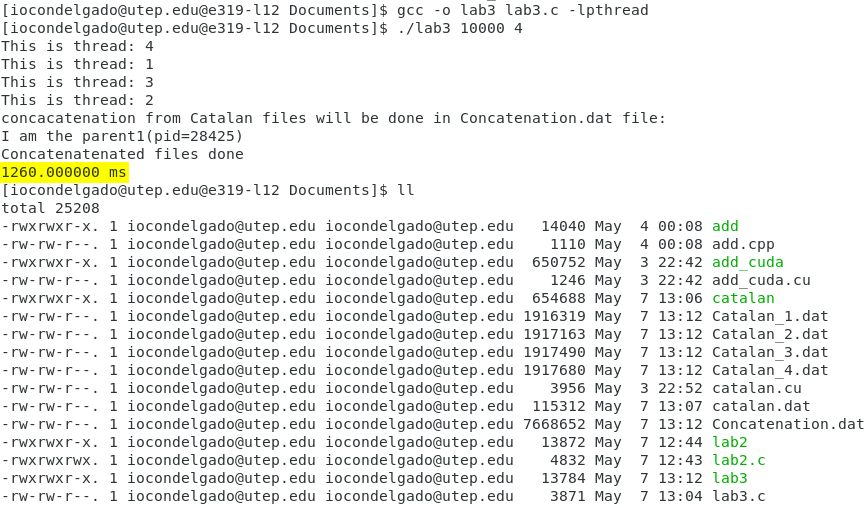
I wrote a multi-threaded program that computes the Catalan number sequence and writes those numbers in fixed point format to a file called “catalan\_\*.dat”. We will generate as many catalan.dat files as we desire and then concatenate them into a file called “Concatenation.dat”. My program will take two command line arguments: the first specifies the number of Catalan numbers to generate and the second specifies the number of threads to use to compute the Catalan number sequence. We can assume a maximum of four threads. For example you can run my program with ten thousand Catalan numbers in 4 threads. In UNIX you can compile as gcc –o catalan catalan.cu –lpthread and run as “./catalan 10000 4”.

# Parallel Implementation

I have worked in the parallel implementation but before I have found ways to allocate memory dynamically. I checked the performance of CPU vs GPU implementation by changing thread allocation to blocks. Moreover, I am comparing both CPU execution vs GPU execution by measuring the time of execution. The most basic way to measure the time was by setting up a timer to measure how long the application executes on the CPU vs GPU. There may be other ways, but it’ll be somewhat more difficult.

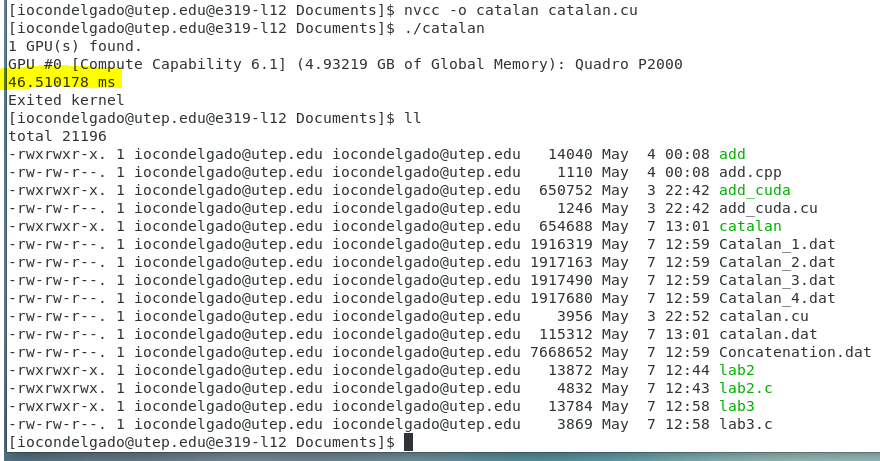
I am running CPU with ten thousand Catalan Numbers in four threads.

CPU Time

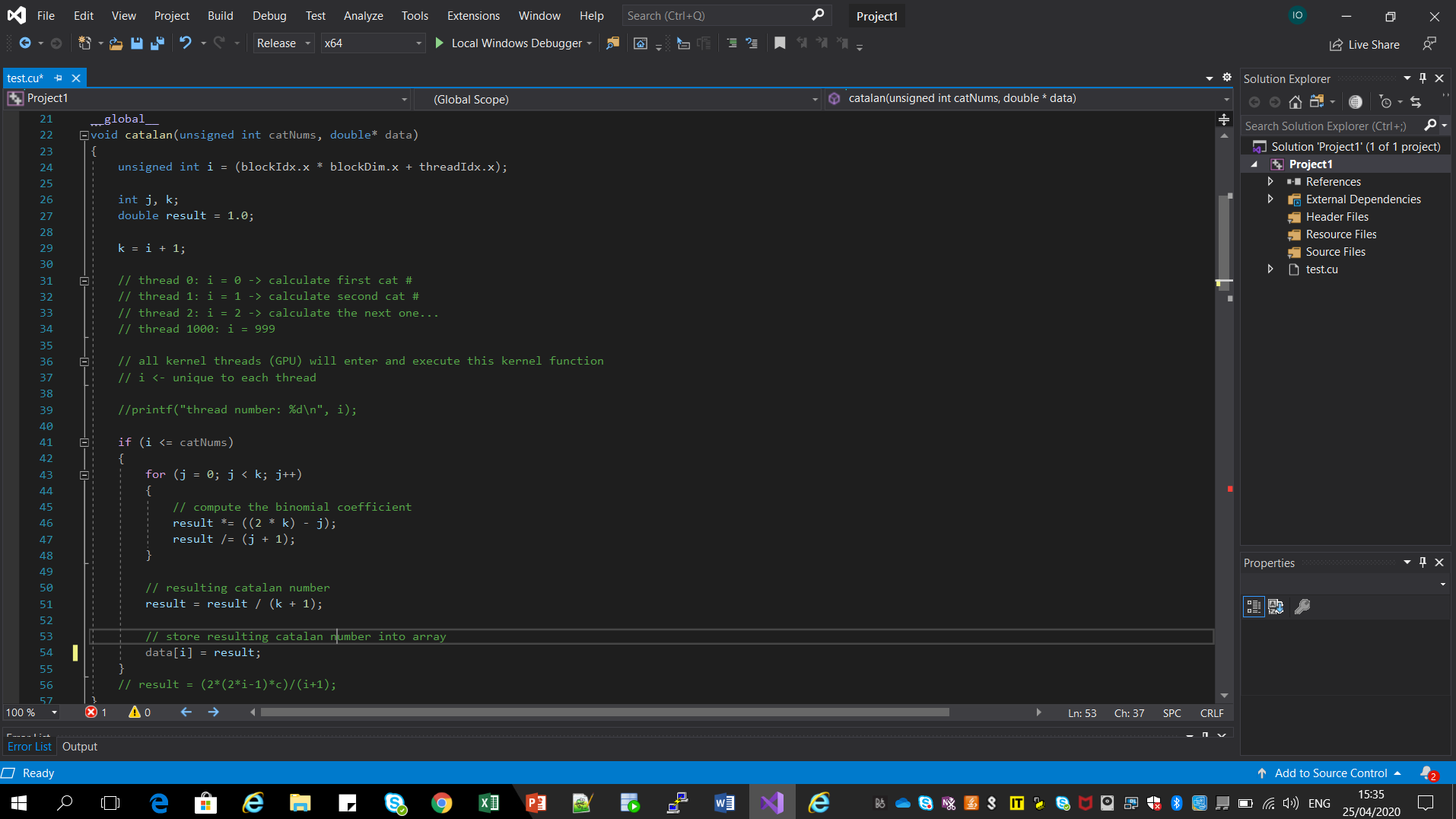


I am also running the GPU with ten thousand Catalan Numbers per thread.

GPU Time



As you can see the time comparison between the CPU and GPU is tremendous. The GPU is much faster than CPU. Even making the Intel CPU a multithreaded program is slower than NVIDIA GPU. Parallel programming in NVIDIA is much faster than Intel CPU. Therefore, I have created this function in CUDA:



## Kernel 1

I have allocated memory on the host, allocate memory on the device, then copy host memory to device, and finally launch the threads into the GPU. I created 1024 thread blocks by creating; if statement that creates a thread block each time that is iterated.



**References:**

<https://mathworld.wolfram.com/CatalanNumber.html>