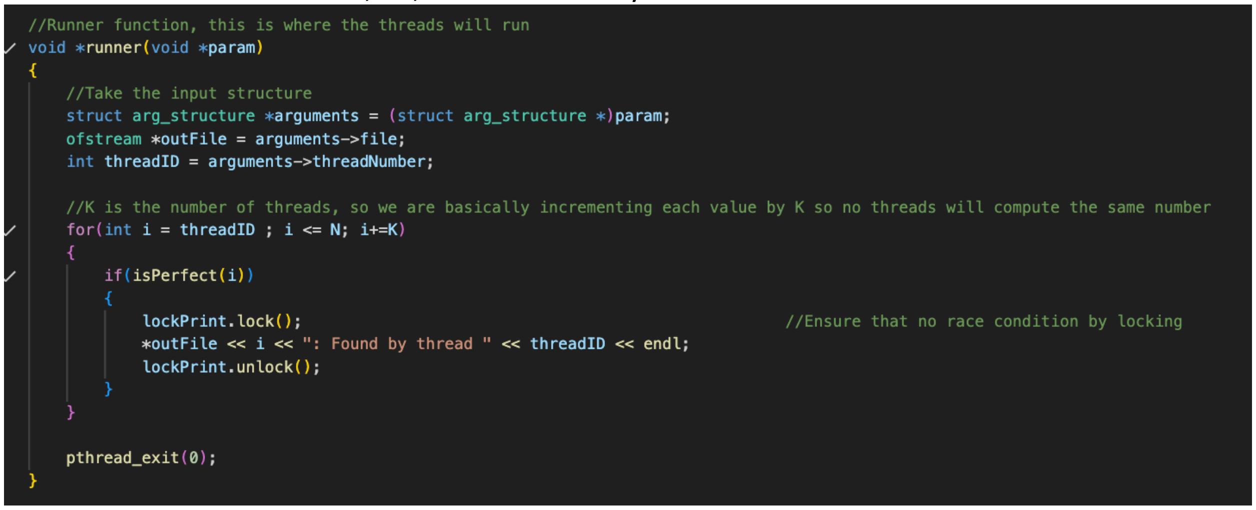
Operating System 2
Programming Exam
Question 1

Name : Ahmik Virani
Roll Number : ES22BTECH11001

<u>Low Level Design</u>

Using the input file I took N and K as input and made K threads.
The threads would run on the runner function. In the runner function each thread calls the isPerfect function which checks if a number is perfect or not.

The logic is :
I start each thread from its thread ID. That is, say thread ID = 1, then it would be assigned 1, 1 + K, 1 + 2K, 1 + 3K and so on. There would be no collision because there are K threads, the first thread checks numbers 1, 1 + K , 1 + 2K, 1+3K and so on, second checks 2, 2 + K, 2 + 2K and so on. The last thread checks K, 2K, 3K so on. Clearly no collisions

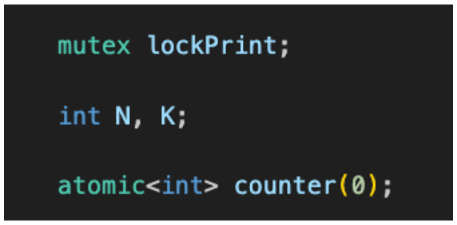


```
//Runner function, this is where the threads will run
void *runner(void *param)
{
    //Take the input structure
    struct arg_structure *arguments = (struct arg_structure *)param;
    ofstream *outFile = arguments->file;
    int threadID = arguments->threadNumber;

    //K is the number of threads, so we are basically incrementing each value by K so no threads will compute the same number
    for(int i = threadID ; i <= N; i+=K)
    {
        if(isPerfect(i))
        {
            lockPrint.lock();                                       //Ensure that no race condition by locking
            *outFile << i << ": Found by thread " << threadID << endl;
            lockPrint.unlock();
        }
    }

    pthread_exit(0);
}
```

The complications that arose were, initially threads were printing output at the same time(race condition) causing inconsistencies in the results, so I added mutex locks to prevent that. Also to avoid race conditions for the global counter, I made it atomic.



```
mutex lockPrint;

int N, K;

atomic<int> counter(0);
```

Performance

Time vs Size, N:

Size(10 ^ n)	Time(micro seconds)
3	465
4	21087
5	1636487
6	165439279

Trend : As the size increases, the time also increases, quite trivially because it takes more time to calculate higher number of numbers

Time vs Number of Threads, K:
(slight change, doing for N = 10 ^ 5)

Number of threads	Time(micro seconds)
1	8034678
2	4076608
4	2124187
8	1633251
16	1554257

Trend : As the number of threads increases, the time decreases, this is because as number of threads increase, the number of numbers checked by each thread decreases, so it becomes faster due to increase in parallelization.