Laboratory #7 – Threads – April 28th 2021

Exercise 1– Write a multi-thread program to search for DNA sequence matching. The program reads the full genome of an organism from a text file. The genome is a string with 4 valid characters: C, G, A, T. The maximum number of characters in a genome is 50000. A second sequence of DNA bases comes from a second file and the program must search if this latter sequence is present in any position of the genome. To increase the computation, this search procedure must be implemented using several parallel threads. Let the user specify the number P of threads.
At the end of the execution the program must print the start and the end positions of the genome subsequences that match with the searched sequence.

As example, look at following files:

Full Genome	Sequence File
CACCATAAAACCGTGTGGGACCTACAGGCAGCAACCTCTGGAGC TACATCACGGATTTTACCCATCCATCCAGGGGGGCGGCGGATTT TTTTTTTTTTTAACGGTATAAAACGGAGAGGGAGACACGTACCC ATAACTCACCACACCCTCTGGAGCTAGAGCAGATTTACGATCGG	TACCCAT

The program must report:

1. 59 – 66
2. 128 – 135

Hints: resort to common fopen and fprintf to deal with files: indeed, both files contains a "string".

Exercise 2– Write a multi-thread program to search if a number N is a prime number by checking if it is divisible by any number in the range [2, N/2]. The program receives in input the number N and the number P of threads to use. The program must create P threads and assign to each of them a subset of possible divisors to test. Each thread checks if N is divisible by any of the assigned divisors. The main program continuously checks the result of the generated threads. As soon as a thread detects that the number is not prime, the main program must stop the executions of all generated threads and exit.
Once the program is ready, try its execution with big numbers and different values of P to understand how the execution time changes. Printing the start and the end time is a good idea to trace the whole execution time.

Hint: before writing the program down, think carefully about the way your algorithm is going to split the data among threads. Then, after the first implementation, ask yourself if the program really stops as soon as one of the threads "detects" that the number is not prime. In order to check it, try with a huge non-prime number and notice how fast it is to end.