PPAR Tutorial 2 **Master 1**

OpenMP

OpenMP

We will start working today with OpenMP, and we will work on the exercices below during two tutorials (4 hours in total). You have therefore a 2-week time period to write, test and send the codes you're going to develop. The deadline for sending the codes is next February 5^{th}, at midnight. The codes should be organized in separated folders (one for each exercise) and included in a tarred and zipped archive. The archive should be sent to the address:

`cedric.tedeschi@inria.fr`

Please include "[PPAR OpenMP]" at the beginning of your email subject.

The following five exercises are independent, sorted in increasing order of difficulty, and conceived for working on different aspects of OpenMP. All codes should be written in C programming language: when compiling, include the `-fopenmp` option in the compilation line.

Please note that your final grade for PPAR will slightly depend upon the codes you will develop and send. Prefer quality to quantity: it is not mandatory for you to send the codes for *all* exercises as you may find some of them difficult.

Exercise 1: Hello world

Write a code where every thread prints `Hello World` in parallel on the screen. Each thread should also print its reference number (id) in the pool of threads. One of these threads should also print the total number of threads that are working in parallel.

Exercise 2: Matrix Multiplication

Write a code implementing the multiplication, in parallel, of two squared integer matrices of size $n \times n$. The value of n is supposed to be given to your program through an input argument, and the real matrices should be randomly generated.

Exercise 3: Eratosthene's Sieve

Eratosthene's Sieve is a general and simple method for the identification of prime numbers. Consider an array of Boolean variables b_i (for instance by using the stdbool library), where $b_i = true$ marks i as prime, while $b_i = false$ marks it as non-prime. Starting from an array where every element is true, Eratosthene's Sieve's main idea is to iteratively mark as non-prime all multiples of each prime, starting with the multiples of 2. Recall that no number can have as a divisor a number greater than its square root.

Write a parallel code implementing Eratosthene's Sieve for computing the first prime numbers up to N. Finally, use the formula

$$\sum_{i=1}^{N} b_i$$

for computing the total number of found prime numbers.

Exercise 4: Char count

Write a parallel code for counting the total number of characters in a table of words. Your code should implement the three following steps:

1. compute the recurrence of each letter in the text;
2. compute the number of vowels and consonants from the result of step 1;
3. compute the total number of letters.

Exercice 5: The philosophers

The dining philosophers problem is a classical example of synchronization issue in concurrent algorithms. Silent philosophers sit at a round table with a delicious dish. Forks are placed between each pair of adjacent philosophers. Each philosopher should alternately eat and think. However, every philosopher can eat only when he has both left and right forks. The problem consists of designing a general behaviour for each philosopher that prevent them from starving.

We will consider the solution proposed by Dijkstra in 1965. An order will be assigned to the forks. Every philosopher will be allowed to take firstly the lower-numbered fork, and then the higher-numbered one, exception made for one philosopher only, which will adopt the inverse behaviour. This general behaviour avoids the situation where all philosophers wait forever for the others to release the forks.

Write a code simulating Dijkstra's solution to this problem. Each philosopher should be assigned to a different thread.