CS3006 Parallel and Distributed Computing
Assignment 3

Question#1

You are required to program the following pseudocodes for finding a given number is prime or not with the given configurations in the openmp environment.

Algorithm 1

```
        // Assume the number is prime
        boolean isPrime = true; // Is the current number prime?

        // Test if number is prime
        for (int divisor = 2; divisor <= number / 2; divisor++) {
          if (number % divisor == 0) { // If true, number is not
prime
            isPrime = false; // Set isPrime to false
            break; // Exit the for loop
          }
        }
```

Algorithm 2

```
1 procedure PRIMES(n):
2 let A be an array of length n
3 set all but the first element of A to TRUE
4 for i from 2 to √n
5     begin
6         if A[i] is TRUE
7         then set all multiples of i up to n to FALSE
8     end
```

Algorithm 3

```
for p ← 2 to n do A[p] ← p
for p ← 2 to ⌊√n⌋ do     //see note before pseudocode
    if A[p] ≠ 0          //p hasn't been eliminated on previous passes
        j ← p * p
        while j ≤ n do
            A[j] ← 0     //mark element as eliminated
            j ← j + p
//copy the remaining elements of A to array L of the primes
i ← 0
for p ← 2 to n do
    if A[p] ≠ 0
        L[i] ← A[p]
        i ← i + 1
return L
```

You have a find the largest 10 digit prime number by executing the *openmp* programs with following configurations and plotting the graphs for performance analysis. You will be using *long long* data type for this purpose.

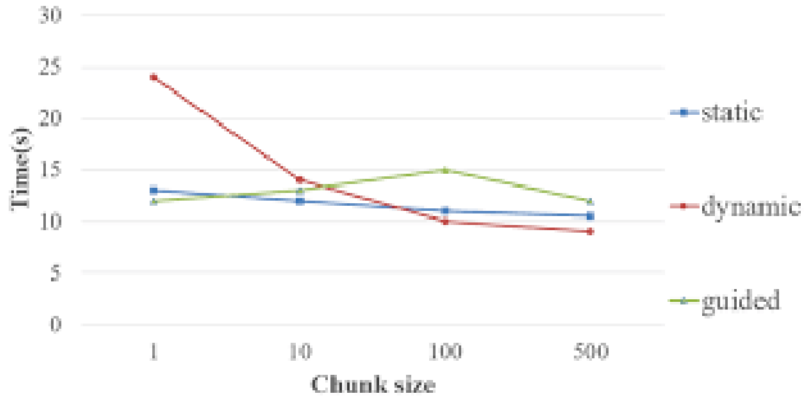
	Number of Threads	Scheduling Mode	Chunk Size
1	lastDigitofRollNumber+9	Static	lastDigitofRollNumber
2	lastDigitofRollNumber+9	Static	lastDigitofRollNumber+1
3	lastDigitofRollNumber+9	Static	lastDigitofRollNumber+2
4	lastDigitofRollNumber+9	Static	lastDigitofRollNumber+3
5	lastDigitofRollNumber+9	Static	lastDigitofRollNumber+4
6	lastDigitofRollNumber+9	Dynamic	lastDigitofRollNumber+1
7	lastDigitofRollNumber+9	Dynamic	lastDigitofRollNumber+1
8	lastDigitofRollNumber+9	Dynamic	lastDigitofRollNumber+2
9	lastDigitofRollNumber+9	Dynamic	lastDigitofRollNumber+3
10	lastDigitofRollNumber+9	Dynamic	lastDigitofRollNumber+4
11	lastDigitofRollNumber+9	Guided	lastDigitofRollNumber
12	lastDigitofRollNumber+9	Guided	lastDigitofRollNumber+1
13	lastDigitofRollNumber+9	Guided	lastDigitofRollNumber+2
14	lastDigitofRollNumber+9	Guided	lastDigitofRollNumber+3
15	lastDigitofRollNumber+9	Guided	lastDigitofRollNumber+4

You will also submit the specifications of the machine used for this purpose as the performance will be varying with the hardware specifications.

Processor model	
Number of cores	

Level 1 cache	
Level 2 cache	
Clock frequency	

A sample of the plot is given below; however you will be providing 3 different plots for each algorithm, i.e., static, dynamic and guided containing multiple lines for number of threads.



Question#2

You MUST use OpenMP and/or MPI to complete the tasks. Do not use pthreads (for multi-threading).

The process ranked 0 (let's call it master and not to be confused with the machine named master) initializes and distributes a large list of numbers amongst other processes (let's call them slaves). The list size and distribution depend on the number of slave processes and each one is assigned an equal share of the numbers to be searched.

The master process then waits for the slave processes to search and report back. Each slave process is doing the similar tasks. First, it is comparing each number in the chunk assigned to the number being searched and secondly at the same time, the slave process is waiting for the abort message from the master process (in case any other process finds the number). For the first case, if the process finds the number, it breaks the search and sends a message to the master that it has found the number. The master process is waiting for exactly this message, it then notifies all the other processes to abort the search.

The important thing here is that a slave processes should abort the search as soon as the abort message is received from the master process. For example, if a process is busy searching 100k numbers and have say searched 5 as yet, and if it receives the abort message it should abort searching immediately rather than the complete the search within 100k numbers.

A sample execution flow is as follows: Master:

The number to search is 39

Process 0 has input data: 1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39

Process 2 has local data: 21 23 25 27 29 31 33 35 37 39

Process 1 has local data: 1 3 5 7 9 11 13 15 17 19

Process 2: I have found the number :-)

Master: Process 2 has found the number!

Master: Informing all processes to abort! Process 1: Aborting search!

Question#3

a)

Implement a serial version of function 'count_occurrence'. Note that this function takes two input arguments: text_string and pattern. The purpose is to find out many times pattern appears in text_string. Some concrete examples are as follows:

count_occurrence ("ATTTGCGCAGACCTAAGCA", "GCA") will return 2

count_occurrence ("AAABCDEFGAAGEREAANMT", "AA") will return 4

count_occurrence ("ABCDEFGHIJKLMNOPQRSTUVWXYZ", "BTTT") will return 0

b)

Implement an OpenMP parallelization version of the above code.