Laboratorio 05

Competencias para desarrollar
Distribuir la carga de trabajo entre hilos utilizando programación en C y OpenMP.

Instrucciones
Esta actividad se realizará individualmente. Al finalizar los períodos de laboratorio o clase, deberá entregar este archivo en formato PDF y los archivos .c en la actividad correspondiente en Canvas.

1. **(18 pts.)** Explica con tus propias palabras los siguientes términos:
   a) private
   Cada hilo que ejecuta un bloque de código paralelo tiene su propia copia independiente de esa variable.

   b) shared
   Una variable es compartida por todos los hilos que ejecutan un bloque de código paralelo.

   c) firstprivate
   Las variables declaradas como firstprivate se inicializan con el valor de la variable original antes de entrar en el bloque paralelo.

   d) barrier
   Es un punto de sincronización en un programa paralelo donde todos los hilos deben detenerse hasta que todos los demás hilos lleguen a ese punto.

   e) critical
   Define un bloque de código que solo puede ser ejecutado por un hilo a la vez.

   f) atomic
   Se utiliza para especificar que una operación simple (como la suma de una variable) debe ser ejecutada de manera atómica, es decir, sin interrupción.

2. **(12 pts.)** Escribe un programa en C que calcule la suma de los primeros N números naturales utilizando un ciclo ***for*** **paralelo**. Utiliza la cláusula **reduction con +** para acumular la suma en una variable compartida.
   a) Define N como una constante grande, por ejemplo, N = 1000000.
   b) Usa omp_get_wtime() para medir los tiempos de ejecución.

3. **(15 pts.)** Escribe un programa en C que ejecute tres funciones diferentes en paralelo usando la **directiva #pragma omp sections**. Cada sección debe ejecutar una función distinta, por ejemplo, una que calcule el factorial de un número, otra que genere la serie de Fibonacci, y otra que encuentre el máximo en un arreglo, operaciones matemáticas no simples. Asegúrate de que cada función sea independiente y no tenga dependencias con las otras.

4. **(15 pts.)** Escribe un programa en C que tenga un ciclo for donde se modifiquen dos variables de manera paralela usando #pragma omp parallel for.
   a. Usa la cláusula shared para gestionar el acceso a la variable1 dentro del ciclo.
   b. Usa la cláusula private para gestionar el acceso a la variable2 dentro del ciclo.
   c. Prueba con ambas cláusulas y explica las diferencias observadas en los resultados.

5. **(30 pts.)** Analiza el código en el programa Ejercicio_5A.c, que contiene un programa secuencial. Indica cuántas veces aparece un valor key en el vector a. Escribe una versión paralela en OpenMP utilizando una descomposición de tareas **recursiva**, en la cual se generen tantas tareas como hilos.

   El valor key aparece mínimo 3 veces, aunque esta cifra puede aumentar, ya que el arreglo se llena de manera aleatoria.

6. **REFLEXIÓN DE LABORATORIO: se habilitará en una actividad independiente.**