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
   significa que cada hilo tendrá su propia copia de la variable. Esto implica que cada hilo trabajará de manera independiente con su copia y los cambios realizados en esta variable no afectarán a la copia de los demás hilos.

   b) Shared
   Una variable shared es compartida entre todos los hilos en un bloque paralelo de OpenMP. Todos los hilos acceden y modifican la misma copia de la variable, por lo que los cambios realizados por un hilo serán visibles para los demás hilos.

   c) Firstprivate
   Combina aspectos de private y shared. Cada hilo recibe su propia copia privada de la variable, pero la diferencia es que esta copia se inicializa con el valor de la variable original antes de entrar en la región paralela.

   d) Barrier
   punto de sincronización en OpenMP donde todos los hilos deben esperar hasta que todos los demás hilos hayan alcanzado ese punto antes de continuar. Esto asegura que todos los hilos hayan completado una cierta parte del código antes de avanzar.

   e) Critical
   define un bloque de código que solo puede ser ejecutado por un hilo a la vez. Es útil para proteger secciones de código que modifican recursos compartidos para evitar condiciones de carrera.

   f) Atomic
   asegura que una operación simple, como una suma o resta, se ejecute sin interrupciones por otros hilos, lo que es esencial para evitar errores en operaciones concurrentes sobre variables compartidas.

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.
    Diferencias Observadas:
        - variable1 (shared): La salida puede variar entre diferentes ejecuciones debido a condiciones de carrera, ya que varios hilos modifican la misma variable al mismo tiempo. Esto puede llevar a resultados no deterministas.

        - variable2 (private): La salida para variable2 es consistente y predecible, ya que cada hilo tiene su propia copia de la variable y no hay interferencia entre ellos.

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.

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