# Laboratorio 05

a) private: Una variable declarada como private en una región paralela significa que cada hilo tendrá su propia copia de esa variable.

b) shared: Una variable shared es accesible y compartida por todos los hilos en una región paralela. Esto significa que, si un hilo modifica el valor de una variable shared, esa modificación es visible para todos los demás hilos.

c) firstprivate: Es similar a private, sin embargo, las variables firstprivate son inicializadas con el valor que tenían antes de entrar en la región paralela. Esto permite que cada hilo tenga su propia copia de la variable, pero que esta copia esté inicialmente configurada con el valor de la variable original.

d) barrier: Es un punto de sincronización en una región paralela. Todos los hilos deben llegar a la barrier antes de que cualquier hilo pueda continuar.

e) critical: Es una sección de código que debe ser ejecutada por un solo hilo a la vez.

f) atomic: Una operación atomic es una forma de asegurar que una operación específica sobre una variable compartida se realice de manera indivisible. Esto significa que cuando un hilo está ejecutando la operación atomic, ningún otro hilo puede interferir hasta que la operación haya terminado.

1. **(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.
2. Define N como una constante grande, por ejemplo, N = 1000000.
3. Usa omp\_get\_wtime() para medir los tiempos de ejecución.
4. **(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.
5. **(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.
   1. Usa la cláusula shared para gestionar el acceso a la variable1 dentro del ciclo.
   2. Usa la cláusula private para gestionar el acceso a la variable2 dentro del ciclo.
   3. Prueba con ambas cláusulas y explica las diferencias observadas en los resultados.
6. **(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.
7. **REFLEXIÓN DE LABORATORIO: se habilitará en una actividad independiente.**