

1. Ejercicio 1

```
useer@useer-VirtualBox:~/Desktop/hdt1$ ./ejercicio1 4
Hello from thread 0 of 4 !
Hello from thread 2 of 4 !
Hello from thread 3 of 4 !
Hello from thread 1 of 4 !
useer@useer-VirtualBox:~/Desktop/hdt1$
```

¿Por qué al ejecutar su código los mensajes no están desplegados en orden?

Cuando se ejecutan threads en paralelo, no hay garantía de que se ejecuten en un orden específico. Los threads pueden competir por recursos de la CPU y ejecutarse en momentos diferentes.

2. Ejercicio 2

```
useer@useer-VirtualBox:~/Desktop/hdt1$ ./hbd_omp 10
Feliz cumpleaños número 10!
Saludos del hilo 4
Feliz cumpleaños número 10!
Feliz cumpleaños número 10!
Saludos del hilo 2
Saludos del hilo 0
useer@useer-VirtualBox:~/Desktop/hdt1$
```

3. Ejercicio 3

```
useer@useer-VirtualBox:~/Documents/GitHub/HDT1_Paralela$ ./riemann 3 7 10000000
Con n = 10000000 trapezoides, nuestra aproximacion de la integral de 3.000000 a 7.000000:
Para x^2: 105.3333333333
Para 2x^3: 1159.9999999998
Para sin(x): -1.7438947509
useer@useer-VirtualBox:~/Documents/GitHub/HDT1_Paralela$
```

4. Ejercicio 4

```
useer@useer-VirtualBox:~/Documents/GitHub/HDT1_Paralela$ ./riemann2 3 7 4
Con n = 10000000 trapezoides, utilizando 4 threads, nuestra aproximacion de la integral de 3.000000
0 a 7.000000:
Resultado total: 105.3333333333
useer@useer-VirtualBox:~/Documents/GitHub/HDT1_Paralela$
```

¿Por qué es necesario el uso de la directiva #pragma omp critical?

Al usar #pragma omp critical alrededor de la operación de suma en la sección paralela, se asegura que solo un thread pueda ejecutar esa operación a la vez. Esto garantiza que los cálculos locales de cada thread se sumen de manera ordenada y segura a la variable total_sum, evitando posibles race conditions y que así se produzca el resultado correcto.

5. Ejercicio 5

```
useer@useer-VirtualBox:~/Documents/GitHub/HDT1_Paralela$ ./riemann3 3 7 4
Con n = 1000000 trapezoides, nuestra aproximacion de la integral de 3.000000 a 7.000000 es 105.333
3333333
useer@useer-VirtualBox:~/Documents/GitHub/HDT1_Paralela$
```

¿Qué diferencia hay entre usar una variable global para añadir los resultados a un arreglo?

Cuando se utiliza una variable global se necesita sincronizar adecuadamente el acceso de los hilos a esa variable, mientras que al almacenar los resultados locales en un arreglo evita la necesidad de esa sincronización, pues cada hilo tiene su propio espacio en el arreglo para almacenar el resultado local. Esto elimina problemas con race conditions.