

<b>Comenzado el</b>	viernes, 19 de mayo de 2023, 07:53
<b>Estado</b>	Finalizado
<b>Finalizado en</b>	viernes, 19 de mayo de 2023, 16:41
<b>Tiempo empleado</b>	8 horas 47 minutos
<b>Calificación</b>	Sin calificar aún

Pregunta **1**

Finalizado

Puntúa como 5,00

Implemente un programa que ejecute 10 hilos que impriman un mensaje identificando al hilo, luego esperen un tiempo aleatorio entre 1 y 5 segundos y luego impriman un mensaje indicando que terminaron (identificando al hilo)

```
import threading
import time
import random

def hilo(id):
    print(f"Hilo {id} iniciando")
    time.sleep(random.randint(1,5))
    print(f"Hilo {id} terminando")

for i in range(10):
    t = threading.Thread(target=hilo, args=(i,))
    t.start()

print("Hilos iniciados")
```



Pregunta **2**

Finalizado

Puntúa como 5,00

Modifique el programa anterior de modo que pueda medir e imprimir el tiempo total que tomo ejecutarse cada hilo (en milisegundos)

```
import threading
import time
import random

def hilo(id):
    print(f"Hilo {id} iniciando")
    start = time.time()
    time.sleep(random.randint(1,5))
    end = time.time()
    print(f"Hilo {id} terminando, tiempo: {end-start}")

for i in range(10):
    t = threading.Thread(target=hilo, args=(i,))
    t.start()

print("Hilos iniciados")
```



Pregunta **3**

Finalizado

Puntúa como 5,00

Implemente un programa que tenga dos hilos A y B, los dos con acceso a una variable X (global) inicializa la variable en un valor entero aleatorio (entre 1 y 100).

El hilo A decrementa X en 1 hasta llegar a 0 intercalando un retardo aleatorio entre 0 y 1 segundo entre cada decremento de X.

El hilo B hará iteraciones cada un tiempo aleatorio entre 1 y 4 segundos, imprimiendo el valor de X en cada iteración hasta que X sea 0.

Tanto A como B deberán imprimir mensajes al arrancar y al terminar, identificando al hilo. El hilo A deberá también indicar el valor inicial de X en el mensaje de arranque o final.

Pregunta: Hay condiciones de carrera? Como las evitaría?



```
import threading
import random
import time

x = random.randint(1,100)

def hilo_A():
    global x
    print(f"hilo_A iniciando identificado como {threading.current_thread().name} valor inicial de x: {x}")
    while x > 0:
        x -= 1
        print(f"hilo_A el valor de x es: {x}")
        time.sleep(random.randint(0,1))
    print(f"hilo_A terminando identificado como {threading.current_thread().name} valor final de x: {x}")

def hilo_B():
    global x
    print(f"hilo_B iniciando identificado como {threading.current_thread().name}")
    while x > 0:
        print(f"hilo_B el valor de x es: {x}")
        time.sleep(random.randint(1,4))
    print(f"hilo_B terminando identificado como {threading.current_thread().name}")

h1 = threading.Thread(target=hilo_A, name="Hilo A")
h2 = threading.Thread(target=hilo_B, name="Hilo B")

h1.start()
h2.start()

# No hay condición de carrera porque hilo_B no modifica el valor de la variable x
```



Pregunta **4**

Finalizado

Puntúa como 5,00

Modificar el programa anterior para que se ejecuten 2 hilos A y un hilo B. Identificar (con comentarios) las zonas críticas y colocar los objetos necesarios para evitar condiciones de carrera.

```
import threading
import random
import time

x = random.randint(1,100)
lock = threading.Lock()

def hilo_A():
    global x
    print(f"hilo_A iniciando identificado como {threading.current_thread().name} valor inicial de x: {x}")
    while x > 0:
        lock.acquire() # acá se bloquea el acceso a la variable x
        x -= 1
        print(f"hilo_A identificado como {threading.current_thread().name} el valor actualizado de x es: {x}")
        lock.release() # acá se libera el acceso a la variable x
        time.sleep(random.randint(0,1))
    print(f"hilo_A terminando identificado como {threading.current_thread().name} valor final de x: {x}")

def hilo_B():
    global x
    print(f"hilo_B iniciando identificado como {threading.current_thread().name}")
    while x > 0:
        lock.acquire()
        print(f"hilo_B identificado como {threading.current_thread().name} el valor de x es: {x}")
        lock.release()
        time.sleep(random.randint(1,4))
    print(f"hilo_B terminando identificado como {threading.current_thread().name}")

hA1 = threading.Thread(target=hilo_A)
hA2 = threading.Thread(target=hilo_A)
hB1 = threading.Thread(target=hilo_B)

hA1.start()
hA2.start()
hB1.start()
```



Pregunta **5**

Finalizado

Puntúa como 5,00

Implemente un programa que pueda lanzar 10 hilos tipo A y 2 hilos tipo B, todos con acceso a una variable global X inicializada en 0.

Los Hilos A incrementan el valor de X hasta 1000000.

Los Hilos B imprime el valor de X cada 2 segundos.

Colocar líneas de comentario en el código, identificando las zonas críticas y los objetos utilizados para evitar condiciones de carrera.

```
import threading
import random
import time

x = 0
lock = threading.Lock()

def hilo_A():
    global x
    print(f"hilo_A iniciando identificado como {threading.current_thread().name} valor inicial de x: {x}")
    while x < 1000000:
        lock.acquire() # acá se bloquea el acceso a la variable x
        if (x < 1000000):
            x += 1
        lock.release() # acá se libera el acceso a la variable x
    print(f"hilo_A terminando identificado como {threading.current_thread().name} valor final de x: {x}")

def hilo_B():
    global x
    print(f"hilo_B iniciando identificado como {threading.current_thread().name}")
    while x < 1000000:
        lock.acquire() # acá se bloquea el acceso a la variable x para poder leerla
        print(f"hilo_B identificado como {threading.current_thread().name} el valor de x es: {x}")
        lock.release() # acá se libera el acceso a la variable x
        time.sleep(2)
    print(f"hilo_B terminando identificado como {threading.current_thread().name}")

for i in range(10):
    hA = threading.Thread(target=hilo_A)
    hA.start()

for i in range(2):
    hB = threading.Thread(target=hilo_B)
    hB.start()
```

