## PONTIFICIA UNIVERSIDAD CATOLICA DEL PERU FACULTAD DE CIENCIAS E INGENIERIA

## <u>Laboratorio Nro 2</u> (2021 - 1)

1) (10 puntos) (nombre de archivo fuente: *agua1.py*) Usted ya conoce el siguiente programa, modificado ligeramente para evitar los bucles infinitos y tener algo ejecuciones aleatorias:

```
from concurrent import futures
import threading as th
import trandom
import time

def Hidrogeno():
    for x in range(1,41):
        time.sleep(random.random())
    print("H",end="")

def Oxigeno():
    for y in range(1,21):
        time.sleep(random.random())
    print("O",end="")

def CambiodeLinea():
    for z in range(1,21):
        time.sleep(random.random())
    print()

# No modificar ninguna de las siguientes líneas

with futures.ThreadPoolExecutor(max_workers=3) as executor:
    executor.submit(Hidrogeno)
    executor.submit(Oxigeno)
    executor.submit(CambiodeLinea)
```

La salida que se pueden obtener son variadas, a continuación algunas de ellas:

Su tarea será sincronizar, usando *barriers*, de forma que siempre se imprima HHO

```
alejandro@abdebian:Soluciones$ python3 agual.py
нно
HH0
HH0
нно
HHO
нно
нно
HHO
нно
нно
HHO
нно
нно
HHO
HH0
нно
HHO
нно
alejandro@abdebian:Soluciones$
```

Laboratorio 2 2021-1

**2)** (10 puntos) Este programa es análogo, la diferencia se encuentra en que no son 3 hilos que realizan bucles, sino son 80 hilos en total. Cada hilo imprime su mensaje y termina, no hay bucles. Para la función *Hidrógeno* son 40 hilos, para la función *Oxígeno* son 20 hilos y para la función *CambiodeLínea* son 10 hilos, a continuación el programa:

```
from concurrent import futures
import threading as th
import random
import time
def Hidrogeno():
    time.sleep(random.random())
     print("H",end="")
def Oxigeno():
     time.sleep(random.random())
     # mantener solo un print
print("0",end="")
def CambiodeLinea():
     time.sleep(random.random())
# mantener solo un print
     print()
# No modificar ninguna de las siguientes líneas
with futures.ThreadPoolExecutor(max_workers=40) as executor:
     for _ in range(20):
    executor.submit(Hidrogeno)
          _ in range(10):
executor.submit(Oxigeno)
          _ in range(10):
executor.submit(CambiodeLinea)
```

Las salidas también son variadas, a continuación alguna de ellas:

```
alejandro@abdebian:2021-1$ python3 agua2.py

H
00HH
HHHHOH
0
HHO
H
HHOHHOH
OHH
HOalejandro@abdebian:2021-1$
```

Sincronice usando *Condition Objects*, para que las salidas permitan cualquier combinación de 2 H, 1 O y 1 cambio de línea siempre al final. El programa debe permitir las diferentes combinaciones.

Pando, 07 de mayo de 2021.

Prof: Alejandro T. Bello Ruiz

Laboratorio 2 2021-1