1. Describan la situación que modelarán:

   Una sala de urgencias tiene a su disposición 10 doctores. Las personas que llegan a la sala son atendidas de acuerdo al nivel de urgencia de su caso, siendo:

   a. Poco urgente (1)
   b. Urgente (2)
   c. Muy urgente (3)
   d. Super urgente (4)

   En caso de que haya algún doctor libre, el paciente ingresará de manera normal. Si ya no hay doctores disponibles, los pacientes serán puestos en una lista de espera hasta que un paciente que estaba siendo atendido, haya terminado y "libere" al médico que lo atendía y el primer paciente que llegó lo pueda "ocupar". Si un paciente de nivel de urgencia máxima (4) llega, este será llevado a urgencias atendido con prioridad, y dado que es una urgencia debe de tomar más tiempo su consulta, no puede ser simplemente un tiempo "aleatorio".

2. ¿Dónde pueden verse las consecuencias nocivas de la concurrencia? ¿Qué eventos pueden ocurrir que queramos controlar?

   A la hora de que un servicio tenga que operar de manera paralela pero no de forma independiente entre sus procesos, como en un sistema de reservaciones, donde cada usuario puede hacer una reservación pero esto implica que una persona que acceda después al mismo sistema debe ver reflejadas las acciones del usuario anterior, y esto puede ser peligroso cuando dos o más usuarios solicitan una misma acción por parte del sistema al mismo tiempo como pedir el mismo asiento en una sala de conciertos al mismo tiempo, sólo por dar un ejemplo más práctico e ilustrativo.

3. ¿Hay eventos concurrentes para los cuales el ordenamiento relativo *no* resulta importante?

   Podrían haberlos, como cuando varios jugadores acceden a una misma sala de juegos, no es tan relevante el orden en el que estos aparezcan, sino que es relevante cuando la sala se ha llenado, por ejemplo, o como cuando todos los jugadores inscritos en la partida han llegado.

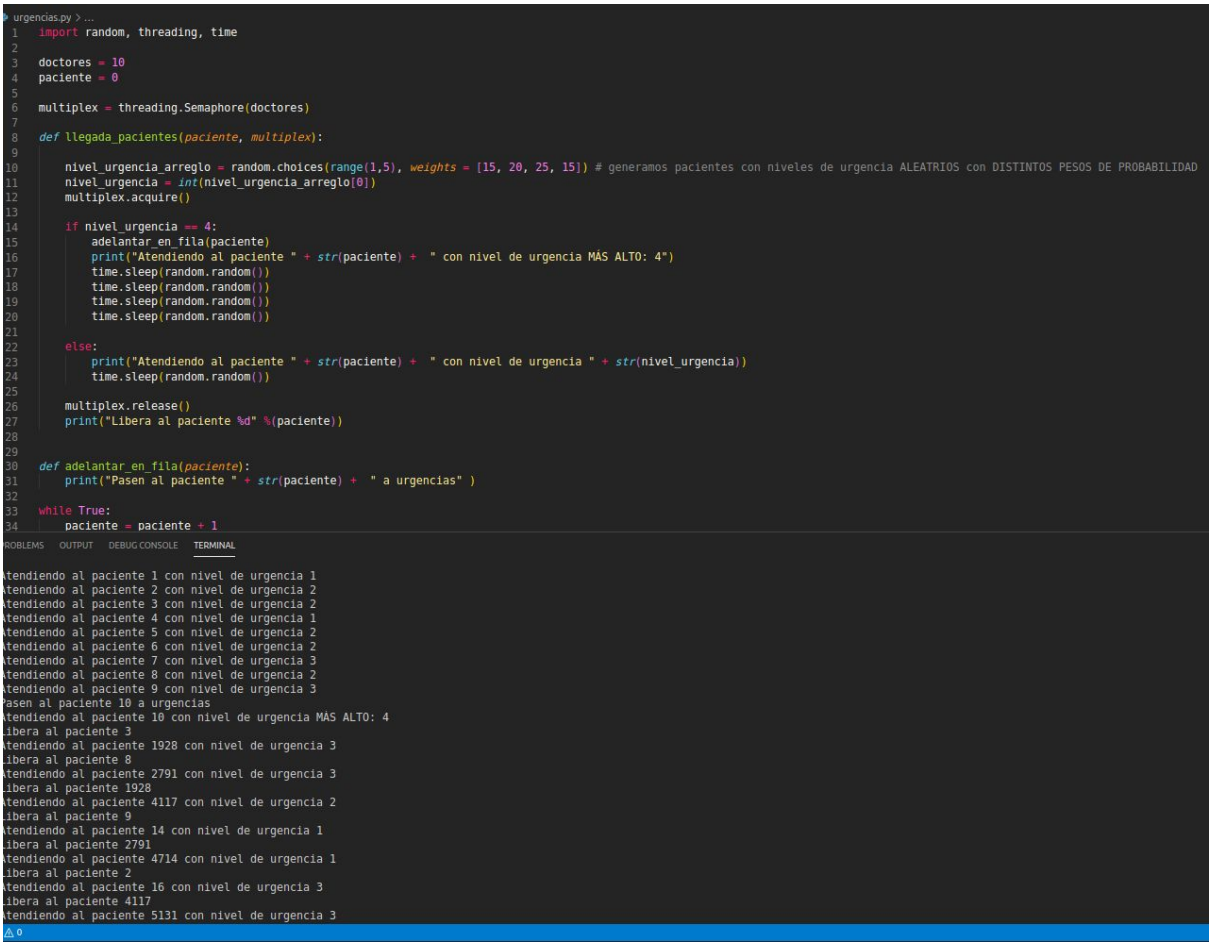
4. Descripción de los mecanismos de sincronización empleados

   Usé el método de sincronización denominado "multiplex", que consiste en reconocer una cantidad límite de procesos donde si este es alcanzado, el sistema deja de atender procesos sino hasta que un proceso es liberado, esto sucede así uno a uno, y en caso de que no se alcance este límite, no hay problema, el sistema debería y debe operar normalmente.

5. Lógica de operación
    a. Identificación del estado compartido (variables o estructuras globales)
        i. Cuando más de diez pacientes llegan a la clínica que únicamente cuenta con diez médicos.
    b. Descripción algorítmica del avance de cada hilo/proceso
        i. Cada proceso es atendido con un identificador dado por un contador que irá aumentando conforme más procesos vayan llegando y soliciten ser atendidos.
        ii. Una vez el proceso llegue, este también cuenta con un nivel de urgencia otorgado de manera probabilística gracias a una matriz de pesos.
    c. Descripción de la interacción entre ellos (sea mediante los mecanismos de sincronización o de alguna otra manera)
        i. Estos procesos interactúan entre ellos gracias al semáforo que comunica a los procesos que están siendo atendidos con los que aún no. Este semáforo sirve como una especie de señalización para dejar pasar al siguiente, de una manera muy similar cuando en un consultorio de farmacias “Similares”, el paciente que sale del consultorio le avisa al primero en la fila que es su turno.

6. Descripción del entorno de desarrollo, suficiente para reproducir una ejecución exitosa
    a. ¿Qué lenguaje emplean? ¿Qué versión?
        i. Python3
    b. ¿Qué bibliotecas más allá de las estándar del lenguaje?
        i. Usamos la librerías: random, threading y time
    c. ¿Bajo qué sistema operativo / distribución lo desarrollaron y/o probaron?
        i. Ubuntu

7. Ejemplos o *pantallazos* de una ejecución exitosa



```python
import random, threading, time

doctores = 10
paciente = 0

multiplex = threading.Semaphore(doctores)

def llegada_pacientes(paciente, multiplex):

    nivel_urgencia_arreglo = random.choices(range(1,5), weights = [15, 20, 25, 15]) # generamos pacientes con niveles de urgencia ALEATRIOS con DISTINTOS PESOS DE PROBABILIDAD
    nivel_urgencia = int(nivel_urgencia_arreglo[0])
    multiplex.acquire()

    if nivel_urgencia == 4:
        adelantar_en_fila(paciente)
        print("Atendiendo al paciente " + str(paciente) +  " con nivel de urgencia MÁS ALTO: 4")
        time.sleep(random.random())
        time.sleep(random.random())
        time.sleep(random.random())
        time.sleep(random.random())

    else:
        print("Atendiendo al paciente " + str(paciente) +  " con nivel de urgencia " + str(nivel_urgencia))
        time.sleep(random.random())

    multiplex.release()
    print("Libera al paciente %d" %(paciente))


def adelantar_en_fila(paciente):
    print("Pasen al paciente " + str(paciente) +  " a urgencias" )

while True:
    paciente = paciente + 1
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

Atendiendo al paciente 1 con nivel de urgencia 1
Atendiendo al paciente 2 con nivel de urgencia 2
Atendiendo al paciente 3 con nivel de urgencia 2
Atendiendo al paciente 4 con nivel de urgencia 1
Atendiendo al paciente 5 con nivel de urgencia 2
Atendiendo al paciente 6 con nivel de urgencia 2
Atendiendo al paciente 7 con nivel de urgencia 3
Atendiendo al paciente 8 con nivel de urgencia 2
Atendiendo al paciente 9 con nivel de urgencia 3
Pasen al paciente 10 a urgencias
Atendiendo al paciente 10 con nivel de urgencia MÁS ALTO: 4
Libera al paciente 3
Atendiendo al paciente 1928 con nivel de urgencia 3
Libera al paciente 8
Atendiendo al paciente 2791 con nivel de urgencia 3
Libera al paciente 1928
Atendiendo al paciente 4117 con nivel de urgencia 2
Libera al paciente 9
Atendiendo al paciente 14 con nivel de urgencia 1
Libera al paciente 2791
Atendiendo al paciente 4714 con nivel de urgencia 1
Libera al paciente 2
Atendiendo al paciente 16 con nivel de urgencia 3
Libera al paciente 4117
Atendiendo al paciente 5131 con nivel de urgencia 3
⚠ 0
```

Aquí sucede algo curioso, que a partir del proceso con identificador “10”, los identificadores comienzan a presentarse de manera aparentemente aleatoria, pero en realidad siguen el mismo patrón, por ejemplo, el “paciente” 11 tiene el identificador 1928, el 12 tiene 2791, el 13 es el 4117 pero el 14 es el 14, el 15 vuelve a ser aleatorio pero el 16 vuelve a ser el 16, es decir sí lleva un orden pero desconozco porqué sucede esto.