Nombre y Apellido: Legajo:

Examen Parcial de Programación 2

Tiempo mínimo para el examen: 1 hora reloj.
Tiempo máximo para el examen: 2 horas reloj.

El examen se compone los siguientes ejercicios. Se solicita documentar bien su código (por ej., cuando deba definir funciones/métodos indique claramente los tipos de los parámetros, el tipo del valor de retorno y explique brevemente qué hace la función/método).

1. Implemente una función recursiva llamada `contar_ocurrencias_rec` que recibe como parámetros una lista y un valor y que cuente cuántas veces aparece ese valor específico en la lista. Escriba, además, una función iterativa `contar_ocurrencias_it` equivalente.

2. Cuente cuantas veces se llama `print` en la siguiente función. ¿Puede determinar el orden de complejidad temporal de este algoritmo?

```python
def f(n: int) -> None:
    for i in range(n):
        print(f"Elemento {i}")
        for j in range(1, n + 1):
            print(f"Sub-elemento {j} en la iteración {i}")
    for i in range(3,0,-1):
        print(i)
    print("Salimos de la función!")
```

3. **Sistema de Gestión de Eventos**

   Implemente un sistema de gestión de eventos utilizando programación orientada a objetos. Para ello, implemente las clases que se especifican a continuación:

   1. **Implementación de una clase Evento:**
      Implemente una clase `Evento` que tenga los siguientes atributos:

      - `codigo_evento` (`str`): el código de identificación único del evento.
      - `nombre` (`str`): el nombre del evento.
      - `fecha` (`str`): la fecha del evento (en formato "YYYY-MM-DD").
      - `hora` (`str`): la hora de inicio del evento (en formato "HH:MM").
      - `capacidad` (`int`): el número máximo de asistentes permitidos.

      La clase debe incluir un método `__init__` (con los argumentos que considere necesarios) y `__str__` que devuelva una representación en cadena del evento.

   2. **Implementación de la clase ReservaEvento:**
      Implemente una clase `ReservaEvento` que tenga los siguientes atributos:

      - `dni_cliente` (`str`): el dni del cliente que realiza la reserva.
      - `nombre_cliente` (`str`): el nombre del cliente que realiza la reserva.
      - `evento` (`Evento`): el evento para el que se realiza la reserva.
      - `numero_asistentes` (`int`): la cantidad de personas que asistirán.

      La clase debe incluir un método `__init__` (con los argumentos que considere necesarios) y `__str__` que devuelva una representación en cadena de la reserva, incluyendo la información del evento.

3. **Implementación de la clase SistemaEventos:**
   Implemente una clase `SistemaEventos` que gestione múltiples eventos y que tenga los siguientes atributos:

   - `eventos` (`dict[str,Evento`): un diccionario con todos los eventos gestionados en el sistema. La clave de cada elemento del diccionario será el el código del evento y su valor asociado será el objeto correspondiente de tipo Evento.
   - `reservas` (`dict[str,list[Reserva]]`): diccionario que guarda una lista de todas las reservas a un evento particular. La clave de cada elemento del diccionario será el código del evento y su valor asociado será una lista de objetos de tipo Reserva.

   Esta clase debe tener los siguientes métodos:

   - `__init__()`: Crea un sistema de eventos vacío (sin eventos ni reservas)
   - `agregar_evento`: Recibe como parámetro un objeto de tipo Evento y lo agrega al diccionario `eventos`, no devuelve nada.
   - `eliminar_evento`: Recibe como parámetro el código de un evento y elimina el evento correspondiente del sistema, si no existe dicho evento no hace nada. No devuelve nada.
   - `mostrar_eventos`: Muestra por pantalla todos los eventos disponibles en el sistema.
   - `devolver_capacidad_restante`: Recibe como parámetro el código de un evento y devuelve la capacidad restante de un cierto evento (o sea, la capacidad todavía no reservada).
   - `crear_reserva`: Recibe como parámetros el dni del cliente que hace la reserva, el nombre de dicho cliente, el código del evento al cual corresponde la reserva y cantidad de lugares que se reservan, permite a un cliente realizar una reserva para un evento específico, validando que exista el evento y que haya suficiente capacidad disponible. En caso que no exista el evento o que no haya suficiente capacidad restante para la nueva reserva, no debe tomarse la misma. No devuelve nada.
   - `eliminar_reserva`: Permite eliminar una reserva de un cliente para un evento específico (si no existe tal reserva no hace nada). Recibe como parámetros el dni del cliente y el código del evento asociados a la reserva. No devuelve nada

```
# Ejemplo de uso
sistema = SistemaEventos()

# Creando eventos
evento1 = Evento("E001", "Concierto de Rock", "2023-10-20", "19:00", 120, 100)
evento2 = Evento("E002", "Teatro Musical", "2023-10-21", "20:00", 90, 50)

sistema.agregar_evento(evento1)
sistema.agregar_evento(evento2)

# Mostrando eventos
sistema.mostrar_eventos()

# Creando reservas
sistema.crear_reserva("12345678", "Juan Perez", "E001", 4)
sistema.crear_reserva("87654321", "Maria Lopez", "E002", 3)

# Devolviendo capacidad restante
print("Capacidad restante para el evento E001:",
       sistema.devolver_capacidad_restante("E001"))

# Eliminando una reserva
sistema.eliminar_reserva("12345678", "E001")# Mostrando reservas después de eliminar
                                            # sistema.mostrar_reservas("E001")
```