Nombre y Apellido: Legajo:

Examen Parcial de Programación 2

Recuperatorio 1er Examen Parcial

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_pacientes_rec` que recibe como parámetros una lista de pacientes (cada paciente es un diccionario con las claves `"Nombre"`,`"DNI"` y `"Síntomas"`) y un síntoma (por ejemplo, "fiebre", "dolor de cabeza", etc) y que cuente cuántos pacientes sufren dicho síntoma (tener en cuenta que un paciente puede tener varios síntomas, el valor asociado a la clave `"Síntomas"` es un `str` con los síntomas separados por `','`). Escriba, además, una función iterativa `contar_pacientes_it` equivalente.

2. Indique el orden de complejidad temporal de este algoritmo. Explique lo más detalladamente posible cómo obtiene esa complejidad.

```python
def registrar_pacientes(n: int) -> list[dict[str,str]]:
    lista_pac = []
    datos = ["Nombre","DNI","Síntomas"]
    for i in range(n):
        print(f"Registrando paciente {i + 1}")
        pac = {}
        for j in range(3):
            pac[datos[i]] = input(f"Ingrese {datos[i]} para paciente {i + 1}:")
            lista_pac.append(pac)
    return lista_pac
```

3. **Sistema de Gestión de Pacientes**

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

   1. **Implementación de una clase Paciente:**

      Implemente una clase `Paciente` que tenga los siguientes atributos:

      - `dni` (`str`): el DNI del paciente.
      - `nombre` (`str`): el nombre del paciente.
      - `edad` (`int`): la edad del paciente.
      - `sintomas` (`str`): los sintomas del paciente (asumiendo que se ingresan separados por `','`).

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

2. **Implementación de la clase Hospital:**

   Implemente una clase `Hospital` que gestione múltiples pacientes y que tenga los siguientes atributos:

   - `pacientes` (`dict[str, Paciente]`): un diccionario que almacena los pacientes, donde la clave es el DNI del paciente.

   Esta clase debe tener los siguientes métodos:

   - `__init__()`: Crea un hospital vacío (sin pacientes).
   - `agregar_paciente`: Recibe como parámetro un objeto de tipo Paciente y lo agrega al diccionario `pacientes`.
   - `eliminar_paciente`: Recibe como parámetro el DNI de un paciente y elimina al paciente correspondiente, si existe.
   - `mostrar_pacientes`: Muestra por pantalla todos los pacientes registrados en el hospital.
   - `contar_pacientes_con_sintoma`: Recibe como parámetro un síntoma y devuelve la cantidad de pacientes que tienen ese síntoma.

```
# Ejemplo de uso
hospital = Hospital()

# Creando pacientes
paciente1 = Paciente("12345678", "Juan Perez", 30, "fiebre, tos")
paciente2 = Paciente("87654321", "Maria Lopez", 25,"dolor de oidos")

hospital.agregar_paciente(paciente1)
hospital.agregar_paciente(paciente2)

# Mostrando pacientes
hospital.mostrar_pacientes()

# Contando pacientes con fiebre
print("Pacientes con fiebre:", hospital.contar_pacientes_con_sintoma("fiebre"))

# Eliminando un paciente
hospital.eliminar_paciente("12345678")
```