Laboratorio 01: Registros de Longitud Fija y Variable

Prof. Heider Sanchez

P1 (10 pts):

Dada la siguiente estructura del registro, en donde se encuentran dos atributos no textuales: `ciclo` y `costo de mensualidad`.

```
Registro Alumno:
    Atributos:
        codigo (Cadena[5])        # Tamaño: 5 caracteres
        nombre (Cadena[11])       # Tamaño: 11 caracteres
        apellidos (Cadena[20])    # Tamaño: 20 caracteres
        carrera (Cadena[15])      # Tamaño: 15 caracteres
        ciclo (Entero)            # Entero para el ciclo
        mensualidad (Decimal)     # Decimal para la mensualidad
```

Requisitos:

Se le pide encapsular las siguientes operaciones de manipulación de archivo binario con dos estrategias de eliminación en una clase llamada `FixedRecord`:

1. **El constructor** debe recibir el nombre del archivo y el modo de eliminación:
   - **MOVE THE LAST**: mueve el último registro a la posición del registro eliminado.
   - **FREE LIST**: mantiene una lista de espacios libres para ser usados en nuevas inserciones.
   - Puede separar la implementación en dos clases, una para cada modo de eliminación.
2. Implementar las siguientes funciones:
   - `load()`: devuelve todos los registros válidos del archivo.
   - `add(record)`: agrega un nuevo registro al archivo O(1). Debe considerar los espacios libres.
   - `readRecord(pos)`: obtiene el registro de la posición "pos" O(1). Debe validar si el registro ha sido eliminado.
   - `remove(pos)`: elimina el registro de la posición "pos" O(1).
3. Realizar las pruebas funcionales de cada método (`P1.py`).

```
# CODE HERE
```

P2 (10 pts):

Se debe implementar un programa para leer y escribir registros de longitud variable en un archivo binario usando el tamaño del dato como separador.

```
Registro Matricula:
    Atributos:
        codigo (Cadena*)        # Tamaño variable
        ciclo (Entero)          # Entero para el ciclo
        mensualidad (Decimal)   # Decimal para la mensualidad
        observaciones (Cadena*) # Tamaño variable
```

Requisitos:

1. Manejar un archivo adicional (**metadata**) para indicar la posición inicial de cada registro.
   - Evaluar si es necesario guardar también el tamaño del registro.
2. Implementar:
   - `load()`: devuelve todos los registros del archivo.
   - `add(record)`: agrega un nuevo registro al archivo O(1).
   - `readRecord(pos)`: obtiene el registro de la posición "pos" O(1).
   - `remove(pos)`: elimina el registro de la posición "pos" (proponer una estrategia eficiente).
3. Realizar las pruebas funcionales de cada método (`P2.cpp`).

```
# CODE HERE
```

Entregable:

- El ejercicio P1 resolver con Python (modulo struct).
- El ejercicio P2 resolver en C++ 17.
- Los dos ejercicios deben contener pruebas de funcionalidad (deben compilar y correr).
- El programa debe permitir la lectura de los datos en cualquier momento.
- Subir al Canvas todo el código fuente, incluidos los archivos de datos utilizados en su programa.