Algoritmos y Estructuras de Datos II

Laboratorio 2025 - 2do Parcial

Tema B: Comisiones 1 y 2 - 2do turno - DNI impar

Requerimientos:

1. **Debe compilar**. Si no compila, no se aprueba el ejercicio.
2. **Debe pasar los tests**. Si no pasa los tests, no se aprueba el ejercicio.
3. **No debe tener memory leaks.** Baja nota.
4. **El código debe ser prolijo y comprensible, indentado y comentado.** Si no, baja nota.

Código kickstart Tema B

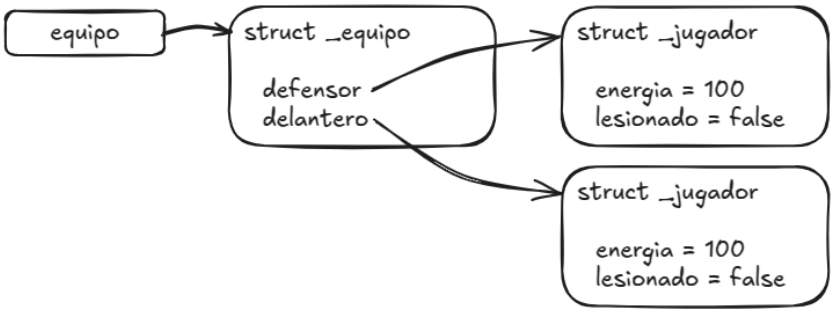
Formulario de entrega: https://forms.gle/67Xg6MfaJ8GfZ3N2A

Ejercicio 1: El médico del equipo

Archivos entregables: `equipo.c`

En este ejercicio se define una estructura de datos para representar un **equipo de dos jugadores**. El equipo tiene **un delantero** y **un defensor**, y cada jugador tiene una cantidad de energía y un booleano que indica si está **lesionado** o no. Un equipo en condiciones correctas tiene ambos jugadores con **energía óptima** y sin lesionar.

En `equipo.h` se define el tipo `equipo_t` y se declaran algunas funciones que se implementan en `equipo.c`. El tipo `equipo_t` se define usando **punteros y estructuras**. En el siguiente diagrama se puede ver un ejemplo de una moto nueva:



Ejercicio 1.1: Implementar la función `curar_equipo()`, especificada de la siguiente manera:

```
/**
 * @brief Cura al equipo
 *
 * Si un jugador no está lesionado, se le debe poner la energía óptima
 * SIN CAMBIARLO.
 * Si un jugador está lesionado, se debe:
 *   1. Sacar el jugador lesionado (liberar memoria)
 *   2. Poner un jugador nuevo con la energía óptima (alojar memoria)
 */
```

Compilar y testear con:

```
$ gcc -Wall -Wextra -std=c99 tests.c equipo.c -o tests
$ ./tests
```

Se incluye un **ejemplo** simple de uso en `ejemplo.c`. Compilar y ejecutar con:

```
$ gcc -Wall -Wextra -std=c99 ejemplo.c equipo.c -o ejemplo
$ ./ejemplo
```

Ejercicio 2: Función `reverse` para el TAD Lista

Archivos entregables: `reverse.c`

En `list.h` se incluye una **especificación del TAD Lista**. En `list.o` se incluye una **implementación pre-compilada** que usa listas enlazadas.

Ejercicio 1.1: Implementar la siguiente operación:

```
/**
 * @brief Devuelve en UNA NUEVA lista el resultado de invertir
 * el orden de los elementos de `l`
 *
 */
list reverse(list l);
```

Compilar y testear con:

```
$ gcc -Wall -Wextra -std=c99 list.o reverse.c tests.c -o tests
$ ./tests
```