Algoritmos y Estructuras de Datos II

Laboratorio 2025 - 2do Parcial

Tema A: Comisiones 1 y 2 - 1er turno - DNI par

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 A

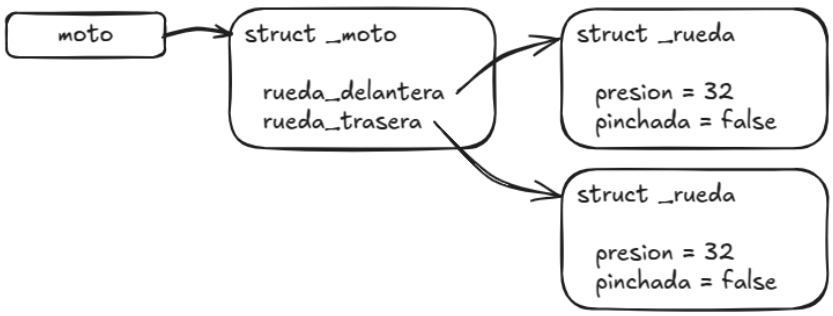
Formulario de entrega: https://forms.gle/ayn8fr7pawjsfg2L9

Ejercicio 1: Gomería de motos

Archivos entregables: `moto.c`

En este ejercicio se define una estructura de datos para representar una moto. Una moto tiene **dos ruedas**, y cada rueda tiene un número que indica la **presión**, y un booleano que indica si está **pinchada** o no. Una moto en condiciones correctas tiene ambas ruedas con **presión óptima** y sin pinchar.

En `moto.h` se define el tipo `moto_t` y se declaran algunas funciones que se implementan en `moto.c`. El tipo `moto_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 `reparar_moto()`, especificada de la siguiente manera:

```
/**
 * @brief Repara las ruedas de la moto
 *
 * Si una rueda no está pinchada, se le debe poner la presión óptima
 * SIN CAMBIARLA.
 * Si una rueda está pinchada, se debe:
 *   1. Descartar la rueda pinchada (liberar memoria)
 *   2. Colocar una rueda nueva con la presión correcta (alojar memoria)
 */
void reparar_moto(moto_t moto);
```

Compilar y testear con:

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

Ejercicio 2: Función `remove_elem` para el TAD Lista

Archivos entregables: `remove.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 eliminar todas las
 * ocurrencias de `e` en `l`
 *
 */
list remove_elem(list l, elem e);
```

Compilar y testear con:

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