Algoritmos y Estructuras de Datos II

Laboratorio 2025 - 2do Parcial

Tema C: Comisiones 3 y 4 - 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 C

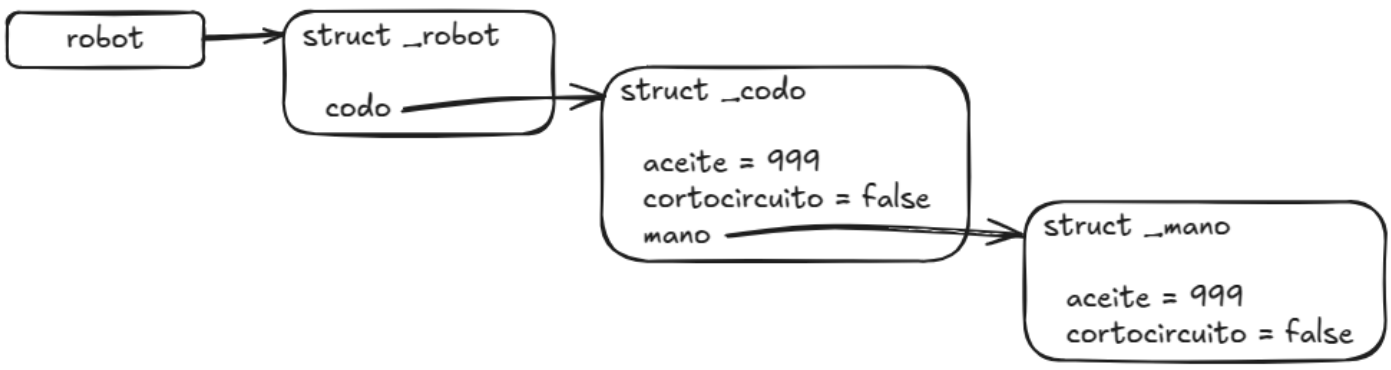
Formulario de entrega: https://forms.gle/K1q5AFT43zWiJVx76

Ejercicio 1: El brazo robótico

Archivos entregables: `robot.c`

En este ejercicio se define una estructura de datos para representar un **brazo robótico.**
El brazo tiene **un codo** y **una mano que sale del codo**. Cada parte tiene una cantidad de aceite y un booleano que indica si está **en cortocircuito** o no. Un robot en condiciones correctas tiene sus partes con **aceite óptimo** y sin cortocircuitos.

En `robot.h` se define el tipo `robot_t` y se declaran algunas funciones que se implementan en `robot.c`. El tipo `robot_t` se define usando **punteros y estructuras**. En el siguiente diagrama se puede ver un ejemplo de **robot nuevo en condiciones correctas**:



IMPORTANTE: Observar que la mano se desprende del codo.

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

```
/**
 * @brief Repara el robot
 *
 * Reemplaza SOLAMENTE las partes quemadas por partes nuevas con aceite óptimo
 * (requiere liberar memoria y alojar memoria nueva)
 *
 * A las partes no quemadas NO LAS REEMPLAZA, sólo les pone el aceite óptimo
 */
```

Compilar y testear con:

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

Ejercicio 2: Función `intercalate` para el TAD Lista

Archivos entregables: `intercalate.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 con el resultado de intercalar el
 * elemento `e` entre todos los elementos de `l`
 *
 */
list intercalate(list l, elem e);
```

Compilar y testear con:

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