

Recursividad y Debugging

Ejercicio 1

En la Planta Nuclear, Smithers está revisando los tableros eléctricos para ver cuáles necesitan mantenimiento urgente.

Se tiene el siguiente struct:



```
typedef struct tablero {
    char sector[MAX_SECTOR];
    int temperatura;
    int luces_rojas;
    bool hace_ruido;
} tablero_t;
```

Un tablero necesita mantenimiento urgente si cumple al menos una de estas condiciones:

- tiene más de 3 luces rojas,
- o su temperatura es mayor a 80,
- o hace ruido y además tiene al menos 1 luz roja.

Dado un vector de tableros, realizar un procedimiento recursivo que guarde en un vector de strings los nombres de los sectores cuyos tableros necesiten mantenimiento urgente.

Ejercicio 2

Moe quiere analizar las consumiciones de la noche en la taberna para saber cómo le fue.

Para eso armó un programa que procesa un vector de consumiciones, pero aunque el código compila, los resultados son incorrectos y, en algunos casos, el programa se termina abruptamente.

Se tiene la siguiente estructura:

```
typedef struct consumicion {
    char cliente[MAX_NOMBRE];
    int cantidad;
```

```
float precio_unitario;
bool pago;
int descuento;
} consumicion_t;
```

Cada consumición representa una compra hecha por un cliente.

Se desea obtener:

1. La recaudación total de las consumiciones pagadas,
2. La cantidad de consumiciones pagadas,
3. El promedio recaudado por consumición pagada,
4. El nombre del cliente que realizó la consumición pagada de mayor importe final.

El siguiente código compila pero no funciona correctamente.

Usar gdb para detectar y corregir los errores.

Ejercicio 3 - Parcial

Todos saben que Los Borbotones son un éxito y que cuesta mucho conseguir entrada para ir a sus shows.

Es por eso que Bart, aprovechando la cercanía con la banda, quiere hacer un mercado negro de entradas. Aún así, le pusieron un par de condiciones para la venta:

- El sector debe ser **platea**
- **y** las entradas que venda tienen que estar entre \$80.000 y \$120.000 (ambos inclusive).
- **y** el asiento debe ser **par**
- **y** estar desocupado.

```
typedef struct asiento {
    bool esta_ocupado;
    int numero_asiento;
} asiento_t;

typedef struct sector {
    char nombre_sector[MAX_LETRAS];
    asiento_t asientos[MAX_ASIENTOS];
    int cantidad_asientos;
    int precio;
} sector_t;
```

Crear una función que recorra una matriz de sectores que representa el predio y devuelva cuántas entradas puede vender Bart.

Aclaración: Puede haber más de un sector que sea platea.