Apellido, Nombre:
Padrón:

Fundamentos de Programación

Exámen final - 09/12/2025

Condición de aprobación: Tener 2 ejercicios **Bien**.

Aclaraciones:

- Un ejercicio práctico se considera **Bien** cuando además de cumplir con lo pedido, aplica las buenas prácticas profesadas por la cátedra.
- En un ejercicio la parte práctica equivale al 75% de la calificación de ese punto mientras que la parte teórica equivale al 25%.

Ejercicio 1

Parte práctica

El señor Burns se despertó con ganas de hacer despidos random por sector. Cada empleado se puede definir con el siguiente struct:

```
typedef struct empleado {
    bool despedido;
    char nombre[MAX_NOMBRE];
    char tareas_a_cargo[MAX_TAREAS][MAX_DESCRIPCION];
    int cantidad_tareas;
} empleado_t;
```

Se pide

Dada una matriz que representa la planta nuclear y donde cada fila es un sector, crear un procedimiento que recorra **recursivamente** la matriz y eche a un empleado por sector teniendo en cuenta que:

- Sólo puede haber un empleado despedido por sector, y ya puede haber un sector con un empleado despedido.
- No se puede echar a los empleados que hagan tareas que **incluyan** la palabra "seguridad", "riesgo" o "crítico".

Aclaración

- Las tareas están en minúscula.
- Si ya hay un despedido en el sector, se puede asumir que es el único.

Parte teórica

Dado el siguiente struct menú:

```
typedef struct plato {
    char ingredientes[MAX_INGREDIENTES][MAX_NOMBRE];
    int cantidad_ingredientes;
    bool es_vegetariano;
} plato_t;

typedef struct menu {
    plato_t platos[MAX_PLATOS];
    int cantidad_platos;
    char color[MAX_COLOR];
} menu_t;
```

Tener en cuenta que el menú está correctamente inicializado con 3 platos también inicializados correctamente, con más de un ingrediente cada uno.

Decir para cada punto si se está haciendo un acceso correcto o incorrecto. De ser correcto, definir tipo de dato al que se está accediendo:

a. `menu.platos.ingredientes[0]`
b. `&(menu.platos[0].cantidad_ingredientes)`
c. `menu.platos[menu.cantidad_platos].ingredientes[0]`
d. `menu.platos[0].ingredientes[0]`
e. `&(menu.platos[cantidad_platos - 2])`
f. `menu.platos[1].ingredientes[0][0]`

Ejercicio 2

Parte práctica

Springfield está haciendo un censo para obtener un ranking de las donas favoritas de los ciudadanos, tienen 2 archivos, uno con las selecciones de cada ciudadano y otro con el ranking previo.

Se pide

Implementar una función que reciba **los nombres de los archivos** de ciudadanos y ranking.

Se debe leer del archivo de ciudadanos para actualizar el de ranking, que tiene datos precargados de la gente censada previamente, actualizando la cantidad de puntos de cada uno y ordenandolo descendentemente por puntos.

Ejemplo:

ciudadanos.csv

```
Homero,chocolate
Marge,vainilla
Bart,frutilla
Lisa,frutilla
Moe,frutilla
Apu,arcoiris
```

ranking.csv

```
chocolate,50
frutilla,49
vainilla,30
arcoiris,15
maple,10
```

ranking.csv (actualizado)

```
frutilla,52
chocolate,51
vainilla,31
arcoiris,16
maple,10
```

Aclaraciones

- El archivo de ranking tiene un máximo de 10 gustos. Se puede asumir que el archivo de ranking entra en memoria.
- Los ciudadanos solo eligen gustos que estén en el ranking.

Parte teórica

¿Cómo se representan las cadenas de caracteres (strings) en C? ¿Para qué sirve la biblioteca string.h? Mencione 3 funciones de string.h y explique cómo funcionan.

Ejercicio 3

Parte práctica

Homero está compitiendo en un concurso de comer donas y para ayudarlo a ganar, se quiere ordenar la torre de donas dejando las de sabor frutilla arriba porque no son sus favoritas. Las demás les encantan y le da igual.

Se tiene el siguiente TDA pila_donas que representa la torre de donas:

```
#ifndef PILA_DONAS_H
#define PILA_DONAS_H

#include <stdbool.h>
#include <stddef.h>

#define MAX_SABOR 500

typedef struct dona {
    char* sabor[MAX_SABOR]; // "frutilla", "chocolate", "glaseada"
    bool con_glaseado;
    int tentacion;
} dona_t;

typedef struct pila_donas pila_donas_t;

/*
 * Crea una pila de donas vacía.
 * Devuelve un puntero a la pila creada, o NULL en caso de error.
 */
pila_donas_t* pila_donas_crear();

/*
 * Destruye la pila liberando todos los recursos de la misma.
 */
void pila_donas_destruir(pila_donas_t* pila);

/*
 * Apila una dona. Devuelve true en éxito, false en error (por ejemplo, memoria ins
 */
bool pila_donas_apilar(pila_donas_t* pila, dona_t* dona);

/*
 * Desapila y devuelve la dona del tope. Devuelve NULL si la pila está vacía.
 */
dona_t* pila_donas_desapilar(pila_donas_t* pila);

/*
 * Indica si la pila está vacía.
 */
bool pila_donas_esta_vacia(const pila_donas_t* pila);


#endif /* PILA_DONAS_H */
```

Se pide

Dada una pila de donas desordenadas, crear una función que procese la pila para que ponga arriba a las donas de frutilla, dejando a las demás en el orden en que fueron encontradas.

Parte teórica

¿Qué diferencias existen entre un TDA “lista” enlazada y un vector estático? ¿Para qué problemas conviene usar cada uno?