Apellido, Nombre:
Padrón:

Fundamentos de Programación

Exámen final - 22/07/2025

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 Sr. Burns chantajea al Hombre Pie para que dé un pastelazo a cada enemigo que le debe dinero, amenazando con revelar su identidad.
La particularidad en esta misión es que el Hombre Pie conoce al siguiente objetivo a partir del actual.

Un enemigo puede representarse con el siguiente registro:

```
typedef struct direccion {
    int fil;
    int col;
} direccion_t;

typedef struct enemigo {
    char nombre[MAX_NOMBRE];
    int dinero_debia;
    bool se_lo_merecia;
    direccion_t proximo_objetivo;
} enemigo_t;
```

El Hombre Pie no siente culpa al darle pastelazos a enemigos que:

- Se lo merecían.
- Y debían más de 10 veces el valor de su sueldo o tienen un nombre demasiado largo (75 caracteres o más).

Se pide

Dada una matriz de `enemigo_t`, crear una función ***recursiva*** que simule el recorrido del Hombre Pie y calcule el total de dinero adeudado por los enemigos de los cuales no sintió culpa.

Aclaración

- El Hombre Pie comienza su recorrido en la dirección (0,0).
- Se termina de recorrer cuando el próximo objetivo tiene dirección (-1,-1)

Parte teórica

Mencionar y explicar al menos 3 funciones de la biblioteca `strings.h` ¿Qué pasa si una cadena no termina en '\0'?

Ejercicio 2

Parte práctica

Todos sabemos que Patty y Selma tienen una adicción a los cigarrillos y que además, son muy competitivas entre ellas.
Es por esto, que Selma quiere ver quien de sus amigos le prestó cigarrillos a su hermana y comparar con los amigos que le prestaron a ella. Específicamente quiere saber quienes son los amigos que le dieron más a su hermana que a ella.

Se pide

Teniendo los archivos `prestadores_selma.csv` y `prestadores_patty.csv`, devolver en un archivo `traicioneros.csv` quienes traicionaron a Selma junto con la diferencia de cigarrillos que le dieron a su hermana y a ella. Además, generar otro archivo `lista_negra.csv` con los nombres de quiénes le dieron más de 10 cigarrillos a su hermana para cortar relación para siempre.

Aclaraciones

- Ambos archivos de prestadores están ordenados por nombre y se espera que el archivo de traicioneros también lo esté.
- Si una persona le dió igual cantidad o más cigarrillos a Selma, **no** se debe agregar a los archivos.

Ejemplo

prestadores_patty.csv

```
bob;5
flanders;5
hombre topo;15
homero;3
krusty;12
serpiente;17
```

prestadores_selma.csv

```
barney;11
bob;7
hombre topo;15
homero;2
marge;4
serpiente;9
```

Deberian resultar:

traicioneros.csv

```
flanders;5
homero;1
krusty;12
serpiente;8
```

lista_negra.csv

```
krusty
serpiente
```

Parte teórica

1. Explicar las diferencias entre una visión caja negra y una caja blanca. ¿Cuáles son las ventajas de cada una?
2. Dados los siguientes conceptos, mencionar si las mismas nos proveen una visión de tipo caja negra o blanca del programa (justificar):
   a. El `.h` de una biblioteca.
   b. El `.c` de una biblioteca.
   c. Un TDA.

Ejercicio 3 (SOLO PARA 2c2024/1c2025)

Parte práctica

Lisa quiere hacer una fila virtual para comprar entradas para ir a ver a Imagine Dragons. Gracias a que sabe que los usuarios tienen un ID único y a que sabe programar, quiere hackear el sistema para entrar antes y conseguir buen lugar.

El problema es que la página tiene un detector de fraude y, si lo hace muy evidente, pueden prohibir su entrada de por vida al sitio. Por lo que va a tener que hacer un truco mas artesanal con los IDs de los usuarios.

Se pide

1. Dada una fila virtual existente y sabiendo que los IDs son aleatorios, crear una nueva fila virtual que inserte al principio a los usuarios que tengan ID que terminen en '5'. Una vez que se termine de ingresar a estos, se debe ingresar a Lisa con el ID `"111"` y luego continuar con los demás restantes en la fila existente.

```
#define MAX_NOMBRE 1000
#define MAX_ID 100

struct fila_virtual;
typedef struct fila_virtual fila_virtual_t;

typedef struct usuario{
    char nombre[MAX_NOMBRE];
    char id[MAX_ID];
} usuario_t;


// Pre: -
// Post: Crea un TDA fila_virtual en el heap y devuelve un puntero al
// mismo. Devuelve NULL si no lo pudo crear.
fila_virtual_t *fila_virtual_crear();

// Pre: - La fila_virtual fue previamente creada con `fila_virtual_crear`.
//      - El usuario ya está completamente inicializado.
// Post: Agrega un usuario **al final** de la fila_virtual.
void agregar_usuario(fila_virtual_t *fila_virtual, usuario_t usuario);

// Pre: La fila_virtual fue previamente creada con `fila_virtual_crear`.
// Post: Devuelve true si habia usuario para eliminar, false en caso
// contrario. En caso de haber usuario, elimina el **primero**
// y lo devuelve por la referencia `usuario_eliminado`.
bool eliminar_usuario(fila_virtual_t *fila_virtual, usuario_t* usuario_eliminado);

// Pre: La fila virtual recibida fue previamente creada con `fila_virtual_crear`.
// Post: Libera la memoria que se reservó en el heap para fila_virtual.
void fila_virtual_destruir(fila_virtual_t *fila_virtual);
```

Parte teórica

a. Explica cómo funcionan malloc y free. ¿Qué es un memory leak?

b. ¿Por qué es importante utilizar buenas practicas a lo largo de todo el código? Mencionar tres buenas prácticas para mantener legibilidad y calidad de codigo, dando un ejemplo donde no se cumpla y un ejemplo donde sí para cada una de ellas.