Apellido, Nombre:
Padrón:

Fundamentos de Programación

Exámen final - 08/07/2025

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

Bart tuvo un exámen multiple-choice, para el cual tenía que completar una matriz en la que cada fila es la pregunta y cada columna es una posible respuesta. Como no estudió, marcó las respuestas de forma aleatoria.

Ahora tiene las respuestas que eran correctas, y quiere saber cuántos puntos se sacó.
Cada opción suma puntos sólo si Bart la marcó y era correcta.
Una opción resta 1 punto si la marcó y era incorrecta.

Se pide:
Dada una matriz booleana que representa las respuestas que marcó o no Bart, y un vector de respuestas correctas con el puntaje que le corresponde, **hacer una función recursiva** que recorra la matriz y calcule la cantidad de puntos que sacó Bart.

```
typedef struct respuesta {
    int respuesta;
    int puntaje;
} respuesta_t;
```

Aclaración

- Los índices de las preguntas empiezan en 0. Además, cada posición del vector de respuesta_t corresponde a la pregunta.
- Se puede asumir que Bart solo marcó una sola respuesta en cada pregunta.

Parte teórica

1. ¿Qué elementos debe tener una función recursiva?
2. ¿Cuando es conveniente usar recursividad? Mencione algunos ejemplos de problemas en los que es mejor su uso.

Ejercicio 2

Parte práctica

Bart es muy fan de la Formula 1 por lo que cada vez que se corre una carrera, se anota en su agenda como fue el top 10 para poder actualizar los puntos para el campeonato de pilotos.

Se pide

Dado un archivo csv `puntos_carrera.csv` que contiene el top 10 de la última carrera, se pide que actualice el archivo csv `campeonato_pilotos.csv` con los puntos que obtuvieron los pilotos y los ordene de forma descendente en caso de ser necesario.

Aclaraciones

- Los puntos corresponden según el lugar que sacaron. Es decir, el primer lugar obtiene 10 puntos, el segundo 9, y asi sucesivamente hasta llegar al puesto 10 con 1 punto.
- Ambos archivos (el de puntos y el de campeonato) siempre tienen los mismos 10 pilotos.

Ejemplo

Recibiendo el archivo:

puntos_carrera.csv

```
Leclerc;1
Antonelli;2
Verstappen;3
Piastri;4
Norris;5
Russell;6
Colapinto;7
Hamilton;8
Gasly;9
Albon;10
```

Y teniendo:

campeonato_pilotos.csv

```
Norris;118
Piastri;116
Verstappen;92
Leclerc;56
Russell;45
Antonelli;43
Colapinto;32
Gasly;32
Hamilton;28
Albon;17
```

El archivo actualizado deberia quedar:

```
Norris;124
Piastri;123
Verstappen;100
Leclerc;66
Antonelli;52
Russell;50
Colapinto;36
Gasly;34
Hamilton;31
Albon;18
```

Parte teórica

a. ¿Por qué le recomendas a Bart que guarde esos datos en archivos? ¿Qué pasaría si los guarda en vectores?
b. Dar las ventajas y desventajas del uso de vectores y de archivos.

Ejercicio 3 (SOLO PARA 2c2024/1c2025)

Parte práctica

Lisa tiene una vitrina colgante en donde guarda, exhibe y protege del polvo a sus trofeos.
La vitrina se abre únicamente de uno de sus extremos por lo que para acceder a un trofeo determinado, primero debe sacar todos los trofeos que se encuentren entre el extremo y ese que quiere acceder.

Se pide

Dado el TDA *vitrina* crear una función que reciba la vitrina cargada, quite, pula, y vuelva a guardar cada trofeo respetando el orden inicial.

```
struct trofeo;
typedef struct trofeo_t trofeo_t;

struct vitrina;
typedef struct vitrina vitrina_t;


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

// Pre: - La vitrina fue previamente creada con `vitrina_crear`.
//      - El trofeo ya está completamente inicializado.
// Post: Agrega un trofeo **al final** de la vitrina.
void guardar_trofeo(vitrina_t *vitrina, trofeo_t trofeo);

// Pre: - La vitrina fue previamente creada con `vitrina_crear`.
// Post: - Quita el trofeo **del principio** de la vitrina devolviendo true si había
// Además devuelve el trofeo quitado mediante la referencia `trofeo_quitado`.
bool quitar_trofeo(vitrina_t *vitrina, trofeo_t *trofeo_quitado);

// Pre: -
// Post: Pule el trofeo recibido por referencia.
void pulir_trofeo(trofeo_t* trofeo);

// Pre: La vitrina recibida fue previamente creado con `vitrina_crear`.
// Post: Libera la memoria que se reservó en el heap para la vitrina.
void vitrina_destruir(vitrina_t *vitrina);
```

Parte teórica

Explicar las operaciones de únion, mezcla e intersección. ¿En qué se diferencian entre sí? ¿Qué condiciones se tienen que cumplir para poder realizar estas operaciones?

Parte teórica

Explicar las operaciones de únion, mezcla e intersección. ¿En qué se diferencian entre sí? ¿Qué condiciones se tienen que cumplir para poder realizar estas operaciones?