Nombre y Apellido: ..... N° Legajo: .....

Primer Parcial de Programación Imperativa

22/09/2025

- **<u>Condición mínima de aprobación: Sumar 5 (cinco) puntos.</u>**
- **Los ejercicios que no se ajusten estrictamente al enunciado, <u>no serán aceptados</u>.**
- **No usar variables globales ni static.**
- **No es necesario escribir los #include**
- **Escribir en esta hoja Nombre, Apellido y Legajo**

Ejercicio 1 (4 puntos)

Se dice que **<u>un vector C se forma a partir de dos vectores A y B</u>** si **todos sus elementos están presentes en A o en B**. Los elementos pueden estar repetidos en C, y no importa si aparecen menos veces en A o en B, esto es, si un elemento está 3 veces en C no es necesario que también aparezca 3 veces, con que aparezca una vez en A o en B es suficiente.

Se dice que **<u>una matriz M1 se forma a partir de dos matrices M2 y M3</u>** si las tres matrices tienen la misma cantidad de filas y **cada fila f de M1 se forma a partir de la fila f de M2 y la fila f de M3** según la definición anterior.

Implementar la función **formada** que retorne 1 si una matriz M1 está formada por las otras dos (M2 y M3) y 0 en caso contrario. Basarse en el siguiente ejemplo de testeo para determinar la cantidad y el orden en que recibe los parámetros:

```
int main() {
    int M1[3][2] = { {1, 5}, {7, 8}, {0, 2} };
    int M2[3][3] = { {1, 3, 5}, {7, 0, 8}, {2, 4, 6} };
    int M3[3][1] = { {0}, {8}, {0} };

    // Se obtiene 1 pues cada fila f de M1 se forma a partir
    // de la fila f de M2 y la fila f de M3
    // Los parámetros de formada son:
    // filas, columnas M1, M1, columnas M2, M2, columnas M3, M3
    assert(formada(3, 2, M1, 3, M2, 1, M3) == 1);

    int M4[3][2] = { {1, 9}, {7, 8}, {0, 2} };
    // Se obtiene 0 pues la fila 1 de M4 no se forma a partir
    // de la fila 1 de M2 y la fila 1 de M3
    // dado que el 9 de la fila 1 de M4 no está
    // en la fila 1 de M2 ni en la fila 1 de M3
    assert(formada(3, 2, M4, 3, M2, 1, M3) == 0);

    int M5[3][3] = { {1, 5, 1}, {7, 9, 8}, {0, 2, 2} };
    int M6[3][2] = { {1, 9},    {7, 9},    {0, 2} };
    int M7[3][1] = { {5},       {8},       {9} };
    assert(formada(3, 3, M5, 2, M6, 1, M7) == 1);

    return 0;
}
```

Nombre y Apellido: ..... N° Legajo: .....

Ejercicio 2 (4 puntos)

Implementar la función void **eraseTop** que recibe:

- string s
- char c
- unsigned int top

La función debe **eliminar de s las primeras top apariciones de c**. El carácter se debe eliminar tanto si aparece en mayúsculas como en minúsculas.

Ejemplos:

s	c	top	s queda
You play me like dun dun dun	'n'	5	You play me like du du du
You play me like dun dun dun	'n'	2	You play me like du du dun
You play me like dun dun dun	'n'	0	*s no cambia*
You play me like dun dun dun	'x'	3	*s no cambia*
You play me like dun dun dun	'y'	1	ou play me like dun dun dun
You play me like dun dun dun	'Y'	2	ou pla me like dun dun dun
You play me like dun dun dun	' '	4	Youplaymelikedun dun dun

Ejercicio 3 (2 puntos)

Implementar la función **cambio** que reciba una cantidad de dinero en pesos (un número entero mayor o igual a cero) y retorne la **cantidad mínima** de billetes de $100, cuántos de $20, cuántos de $10 y cuántas monedas de $1 se necesitan para formar esa cantidad.

Por ejemplo si recibe 577 tiene que retornar que son 5 billetes de $100, 3 billetes de $20, 1 billete de $10 y 7 monedas de $1.

Si recibe 1580 tiene que retornar que son 15 billetes de $100, 4 billetes de $20, 0 billetes de 10 y 0 monedas de $1

La función debe ser void y retornar los cuatro valores esperados **por parámetros de salida**. ***No se aceptarán soluciones que utilicen vectores o que impriman el resultado en pantalla (printf, putchar).***