Tema B

Hacer primero Ej 1 y Ej 2, el Ej 3 es solo para sacar B+, es decir se considerará si todo lo demás esta perfecto.
Para compilar un archivo .c escribir en la terminal en la carpeta donde esta el archivo: $> gcc -Wall -Wextra -std=c99 miarchivo.c -o miprograma
Para ejecutar escribir:
$> ./miprograma

Ejercicio 1

Considerar la siguiente asignación múltiple:

```
var i, j, k : Int;
{Pre: i = I, j = J, k = K, I > 0, J > 0, K > 0}
i, j, k := i * j, j * k, k * i
{Post: i = I * J, j = J * K, k = K * I}
```

Escribir un programa en lenguaje C equivalente usando asignaciones simples teniendo en cuenta que:

- Se deben verificar las pre y post condiciones usando la función `assert()`.
- Los valores iniciales de `i, j, k` deben ser ingresados por el usuario
- Los valores finales de `i, j, k` deben mostrarse por pantalla.

Ejercicio 2

Programar las siguientes funciones:

a)

```
void llena_con_notas(int a[], int tam);
```

que dado un arreglo **`a[]`** de tamaño **`tam`**, `pide al usuario ingresar los valores (int) del arreglo.`

b)

```
bool hay_mas_de_3_aprobados(int a[], int tam);
```

que dado un arreglo `a[]` de tamaño `tam`, devuelve `true` sólo si en el arreglo hay por lo menos tres notas mayor o igual a 6 y menores o iguales a 10. Por ejemplo:

a[]	tam	resultado Comentario
[3,6,5,8,10]	5	**true** Ya que hay más de 3 notas mayores o iguales a 6 y menores o iguales a 10 en el arreglo.
[1,6,3,7]	4	**false** Ya que sólo hay 2 notas mayores o iguales a 6 y menores o iguales a 10 en el arreglo.
[9,77,5,66]	4	**false** Ya que hay 1 sola nota mayor o igual a 6 y menor o igual a 10 en el arreglo.
[7,44,8,9,5]	5	**true** Ya que hay 3 notas mayores o iguales a 6 y menores o iguales a 10 en el arreglo.

Cabe aclarar que la función `hay_mas_de_3_aprobados` no debe mostrar ningún mensaje por pantalla ni pedir valores al usuario.

c)

En la función `main` se debe:

- declarar un arreglo de longitud `N`. Definir a `N` como una constante, **el usuario no debe elegir el tamaño del arreglo**. Recordar que las constantes se definen al principio del archivo usando ***#define***
- Verificar con assert que N sea mayor estricto que 0.
- Llamar a la función llena_con_notas
- Llamar a la función hay_mas_de_3_aprobados
- Mostrar el resultado de hay_mas_de_3_aprobados por pantalla.
- Dejar un par de ejemplos de ejecución

Ejercicio 3*

Hacer un programa que:

-que cuente la cantidad de Aprobados que hay en el arreglo del Ejercicio 2, es decir la cantidad de notas que son mayores o iguales a 6 y menores o iguales a 10.

- que cuente la cantidad de Promocionados que hay en el arreglo Ejercicio 2, es decir la

cantidad de notas que son mayores o iguales a 8 y menores o iguales a 10

- y que devuelva True si encuentro por lo menos 1 nota con la condición de Promocionado.

Para ello programar la siguiente función

```
s_resultado resultados(int a[], int tam);
```

donde la estructura **`struct`** `s_resultado` se define de la siguiente manera:

```
typedef struct {
    int cuantos_aprobados;
    int cuantos_promocionados;
    bool hay_promocionados;
} s_resultado;
```

La función toma un arreglo `a[]` y su tamaño `tam`, y devuelve una estructura con dos enteros y un booleano que respectivamente indican: la cantidad de “Aprobados” (`cuantos_aprobados`), la cantidad de “Promocionados” (`cuantos_promocionados`) y si hubo por lo menos un promocionado (`hay_promocionados`) en el arreglo a[ ]. La función **`resultados`** debe implementarse con un único ciclo y **no debe mostrar mensajes** por pantalla **ni pedir valores al usuario**.

En la función `main` declarar un arreglo de longitud `N` (definir a `N` como una constante, el usuario no debe elegir el tamaño del arreglo), llenarlo con la función **llena_con_notas**, llamar a la función **resultados** y luego mostrar el resultado de la función por pantalla (los tres valores).