Recuperatorio C - Algoritmos I Taller

Ejercicio 1

Considerar la siguiente asignación múltiple:

```
var x, y, z : Int;
{Pre: x = X, y = Y, z = Z, Y > X, X > 0}
x, y, z := y, z, (x / y) + x mod y
{Pos: x = Y, y = Z, z = X}
```

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 `x, y, z` deben ser ingresados por el usuario
- Los valores finales de `x, y, z` deben mostrarse por pantalla.

Ejercicio 2

Programar la función:

```
bool todos_multiplos(int a[], int tam, int k);
```

que dado un arreglo `a[]` de tamaño `tam`, devuelve `true` sólo si todos los valores de `a[]` son múltiplos del elemento de la posición `k` del arreglo. Por ejemplo:

a[]	tam	k	resultado	Comentario
[3, -5, 1, **9**, 7]	5	3	**false**	Ya que los elementos 3, -5, 1 y 7 no son múltiplos de 9, que es el elemento ubicado en la posición k=3.
[6, -9, **3**, 21, 30]	5	2	**true**	Ya que 6, -9, 3, 21 y 30 son todos múltiplos de 3, que es el elemento ubicado en la posición k=2
[7, 21, 11, 3, **1**]	5	4	**true**	Ya que 7, 21, 11, 3, y 1 son todos múltiplos de 1, que es el elemento ubicado en la posición k=4
[**7**, 21, 11, 3, 1]	5	0	**false**	Ya que, por ejemplo, 11 no es múltiplo de 7, que es el elemento ubicado en la posición k=0

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

En la función `main` se debe solicitar al usuario ingresar un arreglo de longitud `N`. Definir a `N` como una constante, **el usuario no debe elegir el tamaño del arreglo**. Luego se debe pedir el índice `k` y verificar con `assert`:

- Que el valor de `k` es un número mayor o igual a 0 y menor estricto que `N`
- Que el elemento del arreglo en la posición `k` sea distinto de 0.

Finalmente se debe mostrar el resultado de la función `todos_multiplos` en la pantalla.

Ejercicio 3

Hacer un programa que calcule la suma de los elementos pares, la suma de los elementos impares y la suma de la totalidad de elementos de un arreglo. Para ello programar la siguiente función

```
struct s_suma_t sumas(int a[], int tam);
```

donde la estructura `struct s_suma_t` se define de la siguiente manera:

```
struct s_suma_t {
    int suma_pares;
    int suma_impares;
    int suma_total;
};
```

La función toma un arreglo `a[]` y su tamaño `tam`, y devuelve una estructura con tres enteros que respectivamente indican: la suma de elementos pares (`suma_pares`), la suma de elementos impares (`suma_impares`) y la suma de todos los elementos (`suma_total`) de `a[]`. La función `sumas` debe implementarse con un único ciclo y **no debe mostrar mensajes** por pantalla **ni pedir valores al usuario**.

En la función `main` se debe solicitar al usuario ingresar un arreglo de longitud `N` (definir a `N` como una constante, el usuario no debe elegir el tamaño del arreglo) y luego se debe mostrar el resultado de la función por pantalla (los tres valores).