Tema A

Ejercicio 1

Considerar la siguiente asignación múltiple:

```
var x, y, z : Int;
{Pre: x = X, y = Y, z = Z, X ≠ 0, Y mod X = 0}
x, y, z := y / x, y + z, x * y
{Post: x = Y / X, y = Y + Z, z = X * Y}
```

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

- Se deben verificar la pre y post condición usando la función `assert()`.
- Los valores iniciales de `x, y, z` deben obtenerse del usuario usando la función `pedirEntero()` definida en el *Proyecto 3*
- Los valores finales de `x, y, z` deben mostrarse por pantalla usando la función `imprimeEntero()` definida en el *Proyecto 3*.

Ejercicio 2

Programar la función:

```
int multiplo_maximo(int a[], int tam, int k);
```

que dado un arreglo `a[]` con `tam` elementos devuelve el elemento más grande de `a[]` que es múltiplo de `k`. Por ejemplo:

a[]	tam	k	resultado
[3, 8, 6, **20**, 5]	5	2	**20**
[3, 8, **6**, 20, 5]	5	3	**6**

Si en el arreglo `a[]` no hubiese un elemento múltiplo de `k` la función debe devolver el neutro de la operación `max` para el tipo `int` (usar `<limits.h>`)

Cabe aclarar que `multiplo_maximo` 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) y luego se debe pedir el valor `k` (verificar con `assert` que `k≠0`) y finalmente mostrar el resultado de la función `multiplo_maximo`.

Ejercicio 3

Hacer un programa que cuente la cantidad de elementos múltiplos de dos, tres y cinco que hay en un arreglo. Para ello programar la siguiente función

```
struct multiplos_t contar_multiplos(int a[], int tam);
```

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

```
struct multiplos_t {
    int n_multiplos_dos;
    int n_multiplos_tres;
    int n_multiplos_cinco;
}
```

La función toma un arreglo `a[]` y su tamaño `tam`, devolviendo una estructura con los tres enteros que respectivamente indican cuántos elementos múltiplos de dos, cuántos múltiplos de tres y cuántos múltiplos de cinco hay en `a[]`. La función `contar_multiplos` 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.

Ejercicio 4*

Hacer un programa que cuente la cantidad de personas que miden más y las que miden menos de una cantidad c que viene dada por un parámetro.

Para ello definimos primero una estructura **`struct`** `persona` que contiene el DNI de una persona y su altura, de la siguiente manera:

```
struct persona {
    int dni;
    float altura;
}
```

Luego se debe programar la función

```
struct alturas_t contar_altos_y_bajos(struct persona a[], int tam, float
alt);
```

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

```
struct alturas_t {
    int n_altos;
    int n_bajos;
}
```

La función toma un arreglo `a[]`, su tamaño `tam` y una altura `alt` para poder comparar. La función devuelve una estructura con los dos enteros que respectivamente indican la cantidad de personas existentes en el arreglo `a[]` que son más altas que `alt`, y en el segundo entero la cantidad de personas que son más bajas.

La función `contar_altos_y_bajos` 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), también se debe pedir la altura para comparar, y por último, se debe mostrar el resultado de la función por pantalla.