Tema D

Ejercicio 1

Considerar la siguiente asignación múltiple:

```
var x, y, z : Int;
{Pre: x = X, y = Y, z = Z, Y ≠ 0, z > 0}
x, y, z := y * z, x mod z, x / y
{Post: x = Y * Z, y = X mod 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 multiplica_multiplos(int a[], int tam, int k);
```

que dado un arreglo `a[]` con `tam` elementos devuelve el producto de los valores de `a[]` que son múltiplos de `k (en caso de no haber múltiplos de k, devolver 1)`. Por ejemplo:

a[]	tam	k	resultado
[3, -5, **2, 4,** 7]	5	2	**8**
[**3,** -5, 1, **9,** 7]	5	3	**27**
[**3, -5, 1, 9, 7**]	5	1	**-945**
[3, -5, 1, 9, 7]	5	11	**1**

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

Ejercicio 3

Hacer un programa que, dado un arreglo `a[]` y su tamaño `tam` obtenga el máximo elemento par y el mínimo elemento impar del arreglo `a[]`. Para ello programar la siguiente función:

```
struct paridad_t maxmin_paridad(int a[], int tam);
```

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

```
struct paridad_t {
    int maximo_par;
    int minimo_impar;
}
```

La función toma un arreglo `a[]` y su tamaño `tam` devolviendo una estructura con dos enteros que contiene el máximo elemento par (`maximo_par`) y otro con el mínimo elemento impar (`minimo_impar`) del arreglo `a[]`. Si en el arreglo `a[]` no hubiese elementos pares, en `maximo_par` debe devolverse el neutro de la operación *máximo* para el tipo `int` (usar `<limits.h>`). De manera análoga, si no hay elementos impares, el valor devuelto en el componente `minimo_impar` debe ser el neutro de la operación *mínimo* para el tipo `int`.

La función `maxmin_paridad` 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.