Tema E

Ejercicio 1

Considerar la siguiente asignación múltiple:

```
var x, y : Int;
var z : Bool;
{Pre: x = X, y = Y, z = Z, X > 0}
x, y, z := x*x + y*y, x + y, x > y
{Post: x = X*X + Y*Y, y = X + Y, 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` deben obtenerse del usuario usando la función `pedirEntero()` definida en el *Proyecto 3*
- El valor inicial de `z` debe obtenerse del usuario usando la función `pedirBooleano()` definida en el *Proyecto 3*
- Los valores finales de `x, y` deben mostrarse por pantalla usando la función `imprimeEntero()` definida en el *Proyecto 3*.
- El valor final de `z` debe mostrarse por pantalla usando la función `imprimeBooleano()` definida en el *Proyecto 3*.

Ejercicio 2

Programar la función:

```
int suma_cuadrados_pares(int a[], int tam);
```

que dado un arreglo `a[]` con `tam` elementos, devuelve la suma de los valores de `a[]` al cuadrado (multiplicados por sí mismos) de aquellos elementos que son pares. Por ejemplo:

a[]	tam	resultado
[3, -1, **2**, **4**]	4	**20**
[3, -5, 1, 9, 7]	5	**0**
[**0**, **6**, 1, **10**, 7]	5	**136**

Cabe aclarar que `suma_cuadrados_pares` 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 finalmente mostrar el resultado de la función `suma_cuadrados_pares`.

Ejercicio 3

Hacer un programa que, dado un arreglo `a[]` y su tamaño `tam` obtenga el mínimo elemento del arreglo `a[]` que es múltiplo de `k y el indice en el que está`. Para ello programar la siguiente función:

```
struct minmul_t multiplo_minimo(int a[], int tam, int k);
```

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

```
struct minmul_t {
    int minimo;
    int indice;
}
```

La función toma un arreglo `a[]` y su tamaño `tam` devolviendo una estructura con dos enteros que contienen el mínimo elemento que es múltiplo de `k` (`minimo`) y otro entero que se corresponde al índice del mínimo encontrado (`indice`). Si en el arreglo `a[]` no hubiese elementos múltiplos de `k`, en `minimo` debe devolverse el neutro de la operación *mínimo* para el tipo `int` (usar `<limits.h>`) y en `indice` debe devolverse `-1`

La función `multiplo_minimo` 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 pedir el valor `k` (verificar con `assert` que `k≠0`). Finalmente mostrar el resultado de la función `multiplo_minimo`

Ejercicio 4*

Hacer un programa que dado un arreglo de compras de productos calcule el precio total a pagar y la cantidad de kilogramos a llevar. Para ello programar la siguiente función:

```
struct total_t calcular_montos(struct producto_t a[], int tam);
```

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

```
struct producto_t {
    int precio;
    int peso_en_kilos;
};
```

y la estructura `struct total_t` se define como:

```
struct total_t {
    int precio_total;
    int peso_total;
}
```

La función toma un arreglo `a[]` con `tam` elementos de tipo `struct producto_t` y devuelve una estructura con dos números que respectivamente indican el precio a pagar y la cantidad de kilogramos de productos que hay en `a[]`. La función `calcular_montos` 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 elementos de tipo `struct producto_t` de longitud `N` (definir a `N` como una constante, el usuario no debe elegir el tamaño del arreglo). Para ello solicitar por cada elemento del arreglo un valor entero y luego otro valor entero. Se puede modificar la función `pedirArreglo()` para facilitar la entrada de datos. Luego se debe mostrar el resultado de la función `calcular_montos` por pantalla.