Tema C

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, z mod y, x / y
{Post: x = Y + Z, y = Z mod 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, 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 indice_maximo_par(int a[], int tam);
```

que dado un arreglo `a[]` con `tam` elementos devuelve el índice más grande de `a[]` que contiene un número par. Por ejemplo:

a[]	tam	resultado
[3, 8, 6, **2**, 5]	5	**3**
[3, 8, 6, 2, **4**]	5	**4**
[**2**, 5, 7]	3	**0**
[3, 5, 7, 11, 9]	5	**-1**

Si en el arreglo `a[]` no hubiese un elemento par la función debe devolver `-1`.

Cabe aclarar que `indice_maximo_par` 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 `indice_maximo_par`.

Ejercicio 3

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

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

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

```
struct paridad_t {
    int maximo_par;
    int maximo_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 máximo elemento impar `(maximo_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 `maximo_impar` debe ser el neutro de la operación *máximo* para el tipo `int`.

La función `maximo_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 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.