Parcial 2 - Algoritmos I Taller: Tema C

Ejercicio 1

Considerar la siguiente asignación múltiple:

```
var i, j, k : Int;
{Pre: i = I, j = J,k = K ,k > 0, i > j}
i, j, k := j + i, j + k ,k + i
{Pos: i = J + I, j = J + K, k = K + I }
```

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

- Se deben verificar las pre y post condiciones usando la función `assert()`.
- Los valores iniciales de `i`, `j` y `k` deben ser ingresados por el usuario
- Los valores finales de `i`, `j` y `k` deben mostrarse por pantalla.

Ejercicio 2

Programar la función:

```
int suma_divisibles(int a[], int tam, int d);
```

que dado un arreglo `a[]` con `tam` elementos devuelve la suma de los valores de `a[]` que son divisibles por el entero `d`. Por ejemplo:

a[]	tam	d	resultado	Comentario
[3, **-6**, 1, **8**, 7]	5	2	2	Se suman sólo los elementos -6 y 8 ya que son los únicos divisibles por 2 .
[**3**, -5, 1, **9**, 7]	5	3	12	Se suman sólo los elementos 3 y 9 ya que son los únicos divisibles por 3.
[3, -5, 1, 9, 7]	5	4	0	dado que no hay elementos divisibles por 4.

Cabe aclarar que `suma_divisibles` 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 entero `d` y finalmente mostrar el resultado de la función `suma_divisibles`.

Ejercicio 3

Hacer un programa que calculé máximo, mínimo y suma sobre los números pares de un arreglo. Para ello programar la siguiente función

```
struct s_paridad_t stats_paridad(int a[], int tam);
```

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

```
struct s_paridad_t {
    int max_par;
    int min_par;
    int sum_par;
};
```

La función toma un arreglo `a[]` y su tamaño `tam`, y devuelve una estructura con tres enteros que respectivamente indican: el elemento más grande de los pares (`max_par`), el elemento más pequeño de los pares (`min_par`) y la suma de los elementos pares (`sum_par`) en `a[]`. La función `stats_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 personas calcule la altura media, la mínima y la máxima. Para ello programar la siguiente función

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

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

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

y la estructura `struct paridad_t` se define como:

```
struct paridad_t {
    float altura_media;
    float altura_minima;
    float altura_maxima;
}
```

La función toma un arreglo `a[]` con `tam` elementos de tipo `struct persona_t` y devuelve una estructura con tres números que respectivamente indican la altura promedio, la altura mínima y la altura máxima de las personas que hay en `a[]`. La función `calcular_estadistica` 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 persona_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 un valor flotante (usar `%f`). Se puede modificar la función `pedirArreglo()` para facilitar la entrada de datos. Luego se debe mostrar el resultado de la función `calcular_estadistica` por pantalla.