# Simulacro parcial 2 - Algoritmos I Taller

---

Aclaraciones

- Este simulacro es una instancia de aprendizaje no evaluativa.
- Tendrán una hora para resolver el simulacro. Luego, resolveremos el simulacro al frente.
- Para aprovechar esta instancia, recomendamos que hagan los ejercicios de manera individual y concentrada

---

## Ejercicio 1

Dada la siguiente estructura:

```
struct info_t {
    int op_sum; // Guarda la suma
    int op_mul; // Guarda la multiplicación
    int op_div; // Guarda la división entera
    int op_dist; // Guarda el valor absoluto de la resta
};
```

Programar la función:

```
struct info_t operar(int x, int y)
```

que dados dos números enteros x e y, calcula las operaciones de suma, multiplicación, división entera y el valor absoluto de la resta entre x e y (en ese orden). Toda esa información es devuelta por la función en una estructura de tipo `struct info_t`.

Por ejemplo:

| a | b | resultado<br>variable res |
|---|---|---|
| 13 | 7 | res.op_sum = 20<br>res.op_mul = 91<br>res.op_div = 1<br>res.op_dist = 6 |
| 10 | 5 | res.op_sum = 15<br>res.op_mul = 50<br>res.op_div = 2 |

| | | |
|---|---|---|
| | | `res.op_dist = 5` |
| 5 | 6 | No cumple la precondición del programa, por lo que no se ejecuta la función y salta un assert. |
| 5 | 0 | No cumple la precondición del programa, por lo que no se ejecuta la función y salta un assert. |

Cabe aclarar que la función `operar` no debe mostrar ningún mensaje por pantalla ni pedir valores al usuario.

En la función `main` se debe solicitar al usuario ingresar dos valores enteros `a` y `b`. Asegurarse mediante la función `assert()` que ambas variables tengan valores mayores a cero y que `a > b`.

Finalmente desde la función `main` se debe llamar a la función `operar` y mostrar el resultado por pantalla.

**NOTA**: Poner como comentario al menos un ejemplo de ejecución, con los parámetros de entrada y la salida de tu programa (podés hacer un copiar y pegar de la consola).

## Ejercicio 2

Dado el siguiente sinónimo de tipos:

```
typedef struct {
    int cant_v;
    int cant_pares;
    int cant_imp;
    bool hay_8;
} datos;
```

Programar la función:

```
datos llenar_estructura(int a[], int tam, int v);
```

que toma un arreglo de enteros, su longitud y un entero v, y devuelve una estructura datos con los siguientes datos: cantidad de veces que encontró a v en el arreglo, cantidad de pares, cantidad de impares del arreglo y un booleano que determina si hay al menos un 8 en el arreglo o no lo hay.

Por ejemplo:

| a | tam | v | resultado<br>variable res |
|---|---|---|---|
| [-1,7,5,10] | 4 | 3 | res.cant_v = 0<br>res.cant_pares = 1<br>res.cant_imp = 3<br>res.hay_8 = false |
| [3,7,8,2,8] | 5 | 3 | res.cant_v = 1<br>res.cant_pares = 3<br>res.cant_imp = 2<br>res.hay_8 = true |
| [3,3,3,0,25] | 5 | 3 | res.cant_v = 3<br>res.cant_pares = 1<br>res.cant_imp = 4<br>res.hay_8 = false |

Cabe aclarar que la función `llenar_estructura` 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`** y un valor entero `v`. Definir a **`N`** como una constante, **el usuario no debe elegir el tamaño del arreglo**.

Finalmente desde la función `main` se debe llamar a la función `llenar_estructura` y mostrar el resultado por pantalla.

**NOTA**: Poner como comentario al menos un ejemplo de ejecución, con los parámetros de entrada y la salida de tu programa (podés hacer un copiar y pegar de la consola).