Parcial 2 - Algoritmos I Taller: Tema A

Ejercicio 1

Considere las siguientes afirmaciones y seleccione la respuesta correcta:

a) Respecto a los elementos que puede tener un arreglo en C:
   1) Se pueden tener elementos de distintos tipos
   2) Sólo se puede tener elementos de un tipo, del tipo del que fue declarado el arreglo.
   3) Se pueden tener elementos de cualquier tipo menos Booleanos
   4) Ninguna de las anteriores es correcta.

b) Respecto de los campos que puede tener una estructura en C:
   1) Puede tener campos de distintos tipos.
   2) Debe tener todos los campos del mismo tipo.
   3) Debe tener al menos 1 campo de tipo entero.
   4) Debe tener al menos 2 campos.

c) ¿Cuál es la librería que tenemos que incluir para poder hacer x=INT_MAX?
   1) `stdio.h`
   2) `assert.h`
   3) `limits.h`
   4) `Ninguna de las anteriores.`

d) El comando para mostrar 10 líneas del programa que estoy ejecutando en GDB es:
   1) display
   2) next
   3) list
   4) Ninguno de los anteriores.

Ejercicio 2

Considere el siguiente código con asignaciones múltiples:

```
var x, y, z : Int;
{Pre: x = X, y = Y, z = Z, Y > X, X > 0}
if (x < y) →
     x, y, z := y, z+y+x, 2*x
□  (x ≥ y)→
    x, y, z := y, z+2*y, y / x
fi
{Pos: (X<Y ∧ (x=Y ∧ y=Z+Y+X ∧ z=2*X)) ∨ (X≥Y ∧ (x=Y ∧ y=Z+2*Y ∧ z=Y/X))}
```

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

- Se deben verificar la pre y la post condición usando la función `assert()`.
- Los valores iniciales de `x, y, z` deben ser ingresados por el usuario.
- Los valores finales de `x, y, z` deben mostrarse por pantalla usando la función `imprimirEntero` del proyecto 3.

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

Ejercicio 3

Dada la siguiente estructura:

```
struct datos {
    bool hay_multiplo_de_10;
    int mayor_multiplo_de_10;
};
```

Programar la función:

```
struct datos hay_multiplo(int tam, int a[]);
```

que dado un tamaño de arreglo `tam` y un arreglo `a[]`, devuelve una estructura **`struct datos`**, en el campo `hay_multiplo_de_10` será **`true`** si en el arreglo `a[]` hay un `múltiplo de 10` y **`false`** en caso contrario. Pueden asumir que el arreglo tiene al menos 2 elementos (chequear esto con `assert`) .En el campo `mayor_multiplo_de_10` se deberá retornar el máximo múltiplo de 10 que haya en el arreglo, y INT_MIN en caso de no haber ningún múltiplo de 10. La función debe programarse utilizando un solo ciclo.

Por ejemplo:

tam	a[]	resultado variable res	Comentario
5	[7,90,4,30,6]	`res.hay_multiplo_de_10 == true` `res.mayor_multiplo_de_10 == 90`	En el arreglo **hay** múltiplo de 10 y **90** es el mayor múltiplo de 10 del arreglo.
5	[9,88,77,66,5]	`res.hay_multiplo_de_10 == false` `res.mayor_multiplo_de_10 == -2147483648`	En el arreglo **no hay** múltiplos de 10 y **-2147483648** es el valor que debe devolver en mayor_multiplo_de_10.
4	[1,2,3,4]	`res.hay_multiplo_de_10 == false` `res.mayor_multiplo_de_10 == -2147483648`	En el arreglo **no hay** múltiplos de 10 y **-2147483648** es el valor que debe devolver en mayor_multiplo_de_10.

Cabe aclarar que la función `hay_multiplo` 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**.

Finalmente desde la función `main` se debe llamar a la función `hay_multiplo` 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 (puedes hacer un copiar y pegar de la consola).

Ejercicio 4

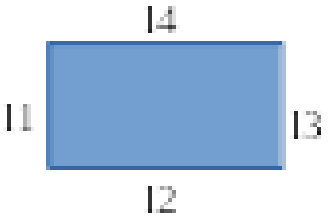
Programar la siguiente función

```
struct tipo_cuadrilatero verificar_cuadrilatero(struct cuadrilatero t);
```

donde las estructuras `cuadrilatero` y `tipo_cuadrilatero` se definen de la siguiente manera:

```
struct cuadrilatero {
    int l1;
    int l2;
    int l3;
    int l4;
};
```

```
struct tipo_cuadrilatero {
    bool es_cuadrado;
    bool es_rectangulo;
    bool es_trapecio;
    bool ninguno_anteriores;
};
```



La función `verificar_cuadrilatero` toma una `struct cuadrilatero`, y devuelve una `struct tipo_cuadrilatero` con cuatro booleanos que respectivamente indican:

- `es_cuadrado` es **`true`** si y sólo si, los cuatro lados l1, l2, l3 y l4 son iguales. Caso contrario es **false**.
- `es_rectangulo` es **`true`** si y sólo si, l1 y l3 iguales y l2 y l4 son iguales, pero no son iguales entre sí. Caso contrario es **false**.
- `es_trapecio` es **`true`** si y sólo si, l1 y l3 son iguales y l2 y l4 son distintos. Caso contrario es **false**.
- ninguno_anteriores es **`true`** si y sólo si, es_cuadrado, es_rectangulo y es_trapecio son **false.** Caso contrario es **false**.

En la función `main` se debe solicitar al usuario ingresar los valores de la **`struct`** `cuadrilatero` y luego de llamar a la función `verificar_cuadrilatero` mostrar el resultado por pantalla (los cuatro booleanos de **`struct`** `tipo_cuadrilatero`).

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