Tema D

Hacer primero Ej 1 y Ej 2, el Ej 3 es sólo para sacar B+, es decir se considerará si todo lo demás esta perfecto.
Para compilar un archivo .c escribir en la terminal en la carpeta donde esta el archivo:
$> gcc -Wall -Wextra -std=c99 miarchivo.c -o miprograma
Para ejecutar escribir:
$> ./miprograma

Ejercicio 1

Considerar la siguiente asignación múltiple:

```
var x, y, z : Int;
{Pre: x = X, y = Y, z = Z, Y > 0}
x, y, z := y, x+y-z,x*(1 + y*y)
{Pos: x = Y, y = X+Y-Z, z = X*(1+Y^2)}
```

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

- Se deben verificar las pre y post condiciones 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.

Ejercicio 2

Programar las siguientes funciones:

a)

```
void llena_con_char(char a[], int tam);
```

que dado un arreglo `a[]` de tamaño `tam, pide al usuario ingresar los valores (char) del arreglo.`

b)

```
bool hay_mas_de_2_consonantes(char a[], int tam);
```

que dado un arreglo `a[]` de tamaño `tam`, devuelve `true` sólo si en el arreglo hay por lo menos 3 consonantes. Por ejemplo:

a[]	tam	resultado	Comentario
['a','e','i','o','u']	5	**false**	Ya que hay más de 2 consonantes en el arreglo.
['j','u','k','o','e']	5	**false**	Ya que sólo hay 2 consonantes en el arreglo.
['l','p','m']	3	**true**	Ya que hay más de 2 consonantes en el arreglo.
['o','l','p']	3	**false**	Ya que hay sólo 2 consonantes en el arreglo.

Cabe aclarar que la función `hay_mas_de_2_consonantes` no debe mostrar ningún mensaje por pantalla ni pedir valores al usuario.

c)

En la función `main` se debe:

- declarar un arreglo de longitud `N`. Definir a `N` como una constante, **el usuario no debe elegir el tamaño del arreglo**. Recordar que las constantes se definen al principio del archivo usando ***#define***
- Verificar con assert que N sea mayor estricto que 0.
- Llamar a la función llena_con_char.
- Llamar a la función hay_mas_de_2_consonantes.
- Mostrar el resultado de hay_mas_de_2_consonantes por pantalla.
- Dejar un par de ejemplos de ejecución como comentario en el código.

Ejercicio 3*

Hacer un programa que cuente la cantidad de ‘a’, ‘x’ y ‘z’ que hay en el arreglo del Ejercicio 2. Para ello programar la siguiente función

```
s_total totales(char a[], int tam);
```

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

```
typedef struct {
    int cuantas_a;
    int cuantas_x;
    int cuantas_z;
} s_total;
```

La función toma un arreglo `a[]` y su tamaño `tam`, y devuelve una estructura con tres enteros que respectivamente indican: la cantidad de ‘a’ (cuantas_a), la cantidad de ‘x’ (cuantas_x) y la cantidad de ‘z’ (cuantas_z) hay en el arreglo a[ ]. La función `totales` debe implementarse con un único ciclo y **no debe mostrar mensajes** por pantalla **ni pedir valores al usuario**.

En la función `main` declarar un arreglo de longitud `N` (definir a `N` como una constante, el usuario no debe elegir el tamaño del arreglo), llenarlo con la función llena_con_char, llamar a la función totales y luego mostrar el resultado de la función por pantalla (los tres valores).