PROGRAMACION AVANZADA - Taller2-2

1) Dado por el usuario una cantidad de números decimales (*double*), entonces:
- Cree un arreglo con asignación dinámica de memoria, de una dimensión y de ese tamaño dado.
- El usuario debe entrar cada número.
- Debe existir una función que reciba el arreglo y retorne el promedio (*como doublé de 2 decimales*).
- Se debe mostrar el promedio obtenido.

2) Para el anterior ejercicio (*el Item 1.*), adecue el código para que ahora, luego de que el usuario entra el último número:
- Se le debe solicitar al usuario entre cuantos números adicionales quiere entrar (*atienda la solución relocalizando el espacio de memoria*)
- Capture los números adicionales.
  (*se debe aún seguir calculando, obteniendo y mostrando el respectivo promedio*)
- Realice la correcta liberación de memoria.

3) Se tiene el sgte **ejemplo** de una declaración e inicialización de un arreglo 2D asignado estáticamente:

```
//d1=2 y d2=3
int arr1[3][2] = {
    {10,20},
    {30,40},
    {50,60}
};
cout<<arr1[2][1]<<endl; //60
```

Y la sgte porción de código declara un apuntador (*int ***) y le hará asignación dinámica de memoria (*es decir un arreglo 2D de apuntadores*):

```
5       int d1=2, d2=3;
6       int **arr2 = (int **)calloc(d2, sizeof(int *));
7       for (int i=0; i<d2; i++)
8           *(arr2 + i) = (int *)calloc(d1, sizeof(int));
9
```

3.1) Realice una representación gráfica (*tal como se han hecho en clase*) de lo sucedido de las líneas #5 a #8.
3.2) Asígnele (*notación apuntador*) correctamente valores al "arreglo" 'arr2' tal que quede exactamente con los valores que se muestran en la inicialización de 'arr1'.
3.3) ¿Qué datos mostrara (*resuélvalo basándose en la representación gráfica 2.1*)?

```
cout<<arr2[1][0];
cout<<*(*(arr2 + 2) + 0);
```

3.4) ¿Qué direcciones mostrara (*resuélvalo basándose en la representación gráfica 2.1*)?

cout<<arr2 + 2;

cout<<(*(arr2 + 2) + 0);

3.5) En notación de arreglo, haga la correcta liberación de toda la memoria reservada por 'arr2'.

4) Con base al sgte código C/C++:

```
1  #include <iostream>
2  #include <cstring>
3  #include <cmath>
4  using namespace std;
5
6  struct OrdenCompra {
7      int numeroOC;
8                          //Defina aqui: una variable char para la fecha de la OC
9                          //Defina aqui: una variable double para el total de la OC
10 };
11
12 struct Proveedor {
13     int nit;
14     char razonSocial[32];
15                         //Defina aqui: la variable para las 'Ordenes de Compra' del Proveedor, pueden ser n pro Proveedor
16 };
17
18 int main() {
19
20     return 0;
21 }
```

4.1) Defina adecuadamente las variables de las Lineas: 9, 14 y 15.

4.2) Realice, tal que el programa haga:

4.2.1) Que el usuario entre la cantidad de 'proveedores'.

4.2.2) Que el usuario por cada proveedor:

Entre cada uno de los campos.

Entre la cantidad de 'ordenes de compra'.

Que el usuario por cada 'orden de compra' entre cada uno de los Campos.

4.2.3) Se muestren los datos en el orden como fueron cargados.

4.2.4) Se muestre la suma de todos los totales de las 'ordenes de compra' por 'proveedor', a partir de un apuntador a funcion que ahora tendra que definir 'Proveedor', y esta funcion tendra esta firma:

```
double totalizarOC(OrdenCompra *oc, int cantidadOC)
```

4.2.5) La correcta y total liberacion de la memoria reservada.