1. Memoria dinámica

Ejercicio 1.1. ¿Qué se mostrará en pantalla?

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int * p;
    p = (int *) malloc(20);
    free (p);
    printf (" %p\n", p);
    printf (" %lu\n", sizeof(p));
    return 0;
}
```

Ejercicio 1.2. ¿Cuál es el error en el siguiente programa?

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int a[3];
    a = (int * ) malloc(sizeof(int)*3);
    free (a);
    return 0;
}
```

Ejercicio 1.3. Desarrolle su propia versión de calloc empleando malloc.

Ejercicio 1.4. Implemente un programa que pida al usuario ingresar temperaturas diarias una por una, de modo que mientras se va ingresando una temperatura, se va mostrando todas las temperaturas ingresadas de manera creciente.

Ejercicio 1.5. Elabore un programa que pida ingresar la cantidad de estudiantes de un salón de clases. Luego, ingrese los promedios ponderados de dichos estudiantes y muestre el ranking. En seguida, se pide ingresar la cantidad de estudiantes que se va adicionar a la lista, se ingresa los promedios ponderados correspondientes y se vuelve a mostrar el ranking.

Ejercicio 1.6. Se pide ingresar una cadena de caracteres y sin emplear funciones de librería:

1. Calcule la longitud de la cadena

2. Invierta la cadena, e.g. si se ingresa “cadena”se debe mostrar “anedac”.

Ejercicio 1.7. Implemente un programa que pida ingresar la cantidad de números **enteros** a generar aleatoriamente, almacenándolos en un **arreglo dinámico**. Luego imprima la lista generada y ordene esta lista de mayor a menor utilizando el **método de ordenamiento de burbuja** de manera **invertida** mediante una función cuyos parámetros de entrada sean un puntero al primer elemento de un arreglo y un entero positivo que almacene la cantidad de elementos de dicho arreglo. Finalmente, imprima la lista ordenada.

Ejercicio 1.8. Elabore un programa que pida ingresar la cantidad de números **enteros pares** a generar aleatoriamente, almacenándolos en un **arreglo dinámico**. Luego imprima la lista generada y ordene esta lista de mayor a menor utilizando el **método de ordenamiento de inserción** de manera **invertida** mediante una función cuyos parámetros de entrada sean un puntero al primer elemento de un arreglo y un entero positivo que almacene la cantidad de elementos de dicho arreglo. Finalmente, imprima la lista ordenada.

Ejercicio 1.9. Cree un programa que pida ingresar la cantidad de números **enteros impares** a generar aleatoriamente, almacenándolos en un **arreglo dinámico**. Luego imprima la lista generada y ordene esta lista de mayor a menor utilizando el **método de ordenamiento rápido** de manera **invertida** mediante una función cuyos parámetros de entrada sean un puntero al primer elemento de un arreglo y un entero positivo que almacene la cantidad de elementos de dicho arreglo. Finalmente, imprima la lista ordenada.

Ejercicio 1.10. Elabore un programa que pida ingresar la cantidad de números **enteros** a generar aleatoriamente, almacenándolos en un **arreglo dinámico**. Luego, devuelva el mínimo y el máximo de estos números mediante una función cuyos parámetros de entrada sean un puntero al primer elemento de un arreglo y un entero positivo que almacene la cantidad de elementos de dicho arreglo.

Ejercicio 1.11. Implemente un programa que pida ingresar la cantidad de partidos y curules en una justa electoral para el Congreso de la República del Perú. Luego, pida ingresar la cantidad de votos que obtuvo cada partido y calcule la cantidad de curules que ganó cada partido mediante el **método de D’Hondt**. Finalmente, imprima los resultados.

Ejercicio 1.12. Implemente un programa que muestre iterativamente el siguiente menú:

¿Desea ingresar una población?

1.- Sí.

2.- No.

Seleccione una de las opciones: _

Si se selecciona la opción 1, se pide ingresar el nombre de un país y la población (en unidades de millones) de dicho país. Luego, se muestra el ranking actualizado de todos los datos ingresados hasta ese momento. Finalmente, si se selecciona la opción 2, el programa debe terminar.