TRABAJO PRÁCTICO 7

Ejercicios combinados,estructuras algorítmicas, cadenas de caracteres, arrays, struct, funciones de cadenas, etc.

1. Dado un vector de registros con los siguientes campos:

   - Nombre y Apellido
   - Edad
   - Categoría (A,B o C)
   - Sueldo

   Diseñar un algoritmo que genere un vector de registros con las especificaciones anteriores, y un entero que indique cuántos registros se han de cargar.

   Se pide:

   a. Cargar la cantidad de registros correspondientes.
   b. Listar el contenido del vector completo.
   c. Calcular y mostrar el promedio de sueldos pagados por la empresa.
   d. Calcular y mostrar, cuántos empleados hay de cada categoría.
   e. Listar todos los empleados que tienen más de 50 años en la empresa. Mostrar un listado ordenado de forma ascendente por edad, de la siguiente forma:

Nombre y apellido	edad

2. Reformular el ejercicio anterior para que el usuario ingrese tres enteros como fecha de nacimiento y calcule la edad teniendo en cuenta el siguiente formato de estructuras:

```
struct Fecha{
    int Dia;
    int Mes;
    int Anio;
};
struct Datos{
    char NomApe[20];
    struct Fecha Edad;
    char Categoria;
    float Sueldo;
};
```

   En este ejercicio solo realice el punto (a) para listar los datos de todos los empleados.

3. Cargar un vector de struct (ver Nota punto c), el mismo va a registrar DNI y estatura de participantes de una competencia de salto en alto. No se sabe la cantidad de participantes que va a ingresar. Solo se ingresa a la carga con 'S' o 'N' según corresponda.

   Se pide:

   a. Cargar los datos con una función que además debe retornar la cantidad de participantes cargados.
   b. Listar los datos completos a través de una función de la siguiente forma:

DNI	Altura

   El sistema debe pedir antes de listar si desea mostrar ordenado por DNI o por altura, luego mostrar el informe correspondiente.

   c. Calcular y mostrar el participante más alto. Con una función que retorne el registro completo teniendo en cuenta el siguiente prototipo:

```
Postulante maximaAltura(Postulante [],int);
```

   Nota: "Postulante", es el tipo de dato struct creado utilizando typedef.

4. La información de 10 notas de todos los alumnos de una institución se encuentra guardada en un arreglo de estructura con el siguiente formato:

```
typedef struct
{
    int Legajo;
    int notas[10];
} datos;
```

   Se pide:

   a. Diseñar una función para simular la carga de información de los 20 alumnos.
   b. Listar los promedios obtenidos de cada alumno, ordenado de manera descendente por promedio de la siguiente forma (tener en cuenta la Nota abajo):

Nro. Legajo	Promedio

Nota: Crear una función para ordenar, una función para listar y una función para calcular el promedio de cada alumno según el prototipo proporcionado a continuación (pensar que argumento o argumentos se necesitan para dicha función).

```
float promAlumno();
```

5. Se tiene almacenada la información de 30 competidores a saber: nombre y apellido, país, deporte del atleta, cantidad de medallas obtenidas. Crear la siguientes estructuras:

```
struct Datos
{
char nombre[40];
char pais[25];
};
struct Deportista
{
char deporte[30];
struct Datos persona;
int cmedallas;
};
```

Se pide:

a. Diseñar una función para simular la carga de los 30 competidores y luego muestre por pantalla un listado general, de la siguiente forma:

Nombre y apellido	Deporte	País	Cantidad de Medallas

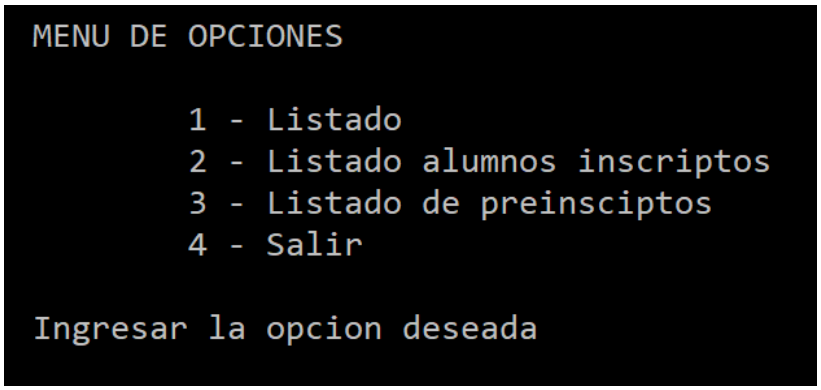
b. Informar Nombre completo y cantidad de medallas del o los deportistas que más medallas ganaron, crear una función que retorne el máximo.

6. Se tiene un vector que guarda la información de posibles ingresantes a la carrera de informática, el mismo guarda los siguientes datos : Legajo, Nombre y apellido y mail. La siguiente declaración será usada como base de datos para luego procesar la información:

```
Inscripto personas[]={{7166,"Susana Juarez","sujua@email.com"},{5563,"Rafael Sordio","rasor@email.com"},
                      {9584,"Carlos Ritero","car05@email.com"},{0,"Sabrina Ortega","otega23@email.com"},
                      {5139,"Raul Frenchi","rafre@email.com"},{7559,"David Gwemes","davidg@email.com"},
                      {0,"Nicolas Wete","niwe@email.com"},{6789,"Vanesa Sanchez","va2022@email.com"},
                      {0,"Cintia Trani","citra@email.com"},{6057,"Pedro Buerta","pebue@email.com"},
                      {5145,"Cecilia Clevand","ceci2002@email.com"},{0,"Erica Carson","eri20@email.com"},
                      {7121,"Berta Gallegos","bega100@email.com"},{9254,"Caro Stres","caes@email.com"},
                      {9825,"Camila Seras","camis0501@email.com"},{0,"Segio Kramer","sergi@email.com"},
                      {5050,"Yi Young","yiyo@email.com"},{6620,"Ariel Zamora","arizamo@email.com"},
                      {0,"Catalina kiroz","catak@email.com"},{6559,"Sofia Reina","so041999@email.com"}};
```

Se pide:

a. Generar un menú de opciones de la siguiente forma:



b. La opción 1 utiliza una función para listar todo el vector completo.
c. La opción 2 utiliza una función para listar el registro completo de los inscriptos que ya tienen legajo asignado.
d. La opción 3 utiliza una función para listar el registro completo de los preinscriptos es decir aquellos personas que tienen legajo 0 (cero).
e. Todos los informes deben salir ordenados por nombre y apellido. Crear una función para tal fin.