Ej. 1: Crear una estructura que permita guardar información sobre un punto en el plano (utilizando sus coordenadas en los ejes x e y). Luego realizar un programa que solicite al usuario que se ingresen los datos de dos puntos e indique cuál de los dos puntos se encuentra más lejos del origen de coordenadas (0;0)

Ej. 2: Utilizar la estructura definida del ejercicio anterior para representar puntos en el plano. Desarrollar una función que lea del archivo "puntos.csv" los datos de diez puntos (*x1,y1,x2,y2,...xn,yn*) y almacene dichos puntos en un arreglo de estructura punto. Desarrollar otra función que calcule e retorne la distancia mínima entre dos puntos pasados por parámetro.
Luego, utilizando las funciones anteriores, desarrollar un programa que calcule e imprima las siguientes distancias mínimas:

2.1. **mínima distancia entre dos puntos consecutivos** (según el orden de carga).

2.2. **mínima distancia entre dos puntos cualesquiera**. *Es decir, buscar aquellos puntos que están más cerca e imprimir dicha distancia.*

Es un buena práctica crear una función para cada item a resolver.

Ej. 3: Utilizando los archivos generados en los ejercicios 1.13 **`cabeceras.txt`**, **`items.txt`**, **`datos.txt`**. realizar:
Desarrollar una función que cargue los datos de **`items.txt`** y **`datos.txt`** en un único arreglo de estructura, el cual recibe por parámetro, además de los nombres de los dos archivos. La estructura debe contener el nombre, la cantidad de partidos ganados, la cantidad de partidos perdidos, la cantidad de partidos empatados, la cantidad de goles a favor y la cantidad de goles en contra.
Desarrollar una segunda función que pasado por parámetro el arreglo de estructuras y el nombre del archivo **`cabeceras.txt`**, mostrará en consola una tabla ordenada (primero por puntos, luego por diferencia de gol y luego por goles a favor) de los equipos como se ejemplifica en el ejemplo de salida.
Realizar un programa que invocando a las funciones anteriores, obtenga la siguiente salida:

Ejemplo de salida:

```
Equipo       PTS  PJ   PG   PE   PP   GF   GC   DIF
-------------------------------------------------
Milan         10   4    3    1    0   12    3    +9
Real Madrid    9   4    3    0    1   11    2    +9
Ajax           4   4    1    1    2    8    8     0
Arsenal        3   4    0    3    1    5   11    -6
Bayern Munich  1   4    0    1    3    2   14   -12
```

Ej. 4: Desarrollar una función que le permita al usuario cargar en un archivo (csv) `personas.txt` los datos de personas (n° de documento, nombre, país). Se debe permitir ingresar por teclado de los datos hasta una cantidad máxima N (constante),o finalizando cuando el documento ingresado es cero.

Ej. 5: Desarrollar una función que le permita al usuario imprimir en formato de tabla el archivo creado en el ejercicio 2.4.

Ej. 6: Realizar una función que lea el archivo creado en el punto 2.4 y ofrezca al usuario el siguiente menú de opciones:

```
1- Ver listado ordenado por nombre
2- Ver listado ordenado por documento
3- Ver listado ordenado por país
4- Salir del programa
```

Las opciones 1,2 y 3 deben imprimir por pantalla un listado ordenado con el formato del siguiente, ejemplo:

```
Documento     Nombre           Pais
========================================

4815162       Matias Zorro     Argentina
3424815       Benjamin Lino    Brasil
1623424       Pedro Chang      China
8151623       Carlos Masanch   Uruguay
```

Ej. 7: Se dispone de las siguientes estructuras:

```
typedef struct
{
  char nombre[100];
  int legajo, materias[30];
}
t_alumno;
```

```
typedef struct
{
  char nombre[100];
  int codigo;
}
t_materia;
```

```
typedef struct
{
  int legajo_alumno;
  int cod_materia,nota;
}
t_nota;
```

De cada alumno se tiene su nombre, su número de legajo y un listado (en el arreglo) donde están guardados los códigos de las materias que ha cursado. De cada materia se tiene su nombre y un código numérico (debe ser un número entero mayor que cero).
Las notas están en la tercera estructura, cada una de las cuales contendrá el legajo del alumno, el código de la materia y la nota obtenida en dicha materia. Se utilizará el cero para aquellas materias para las cuales no tenga calificación.
Realizar un programa que integre el uso de las siguientes funciones :

1) Una función que cargue los datos de los alumnos en el archivo "alumnos.txt" hasta ingresar legajo cero. Se asume que hay una cantidad máxima NA (constante) de alumnos
2) Una función que cargue los datos de las materias en el archivo "materias.txt" hasta ingresar codigo cero. Se asume que hay una cantidad máxima NM (constante) de materias.
3) Una función que reciba como parámetro un arreglo de alumnos, se asume que hay una cantidad máxima NA (constante) de alumnos, y lo cargue con los datos almacenados en el archivo "alumnos.txt".
4) Una función que reciba como parámetro un arreglo de materias, se asume que hay una cantidad máxima NM (constante) de materias, y lo cargue con los datos almacenados en el archivo "materias.txt".
5) Una función que reciba como parámetro, un arreglo de alumnos (ya cargado) y un arreglo de materias (ya cargado) y que permita ingresar, por teclado, las notas de cada alumno para cada materia que cursa, imprimiendo en pantalla, previo al ingreso de la nota, el nombre del alumno, el nombre de la materia. Por ejemplo:"`Ingrese la nota de Luis Huergo para la materia Informática II: `". Finalmente la función debe almacenar en el archivo `notas.txt` los datos correspondientes a la estructura `t_nota` en una línea por registro.
6) Una función que reciba como parámetro un arreglo de notas de alumnos, y lo cargue con los datos almacenados en el archivo `notas.txt`.

7) Una función que reciba como parámetro los arreglos cargados con los alumnos, las materias y las notas y el legajo de un alumno e imprima un listado de las materias de ese alumno que tienen nota cargada junto con su valor.
8) Una función que reciba como parámetro los arreglos cargados con los alumnos, las materias y las notas y un código de materia e imprima un listado de todos los alumnos que tienen nota cargada en dicha materia y su nota.

Ej. 8: Realizar una función que muestre la representación binaria de una variable de tipo char, utilizando los operadores SHIFT (desplazamiento a izquierda y a derecha). Aplicarla en un programa de ejemplo.

Ej. 9: Realizar una función que efectúe el SHIFT A DERECHA completando con 0 desde la izquierda. Aplicarla en un programa de ejemplo.

Ej. 10: Realizar una función que cuente la cantidad de bits que están en 1 en una variable de tipo unsigned int.

Ej. 11: Realizar una función que reciba como parámetros 4 unsigned char y que coloque cada uno de ellos en una variable de tipo unsigned int, que devolverá en su valor de retorno.

Ej. 12: Realizar una función que reciba como parámetro una letra y que la cambie a mayúscula (si está en minúscula) o que la cambie a minúscula (si está en mayúscula). Dicha cambio consiste en cambiar un solo bit del char. Determinar cuál es ese bit, qué operador hay que aplicar sobre el char original y con qué máscara. La función tendrá una única línea:
`return letra` *`<OPERADOR>`* *`<MÁSCARA>`*