IMPORTANTE:

- **Crear un proyecto con su Nombre y Apellido.**
- **Añadir comentarios TANTO EN LAS FUNCIONES COMO EN EL MAIN identificando con el número de ejercicio e inciso a la/s función/es vinculadas a la resolución.**
- **Si no se identifica, la resolución no tendrá validez.**

Una importante empresa de venta de ropa deportiva les encarga realizar un pequeño sistema para manejar el stock en sus distintas sucursales.
La información se encuentra almacenada en el archivo subido en el espacio de entrega del parcial dentro del campus, el cual hay que descargar y copiar dentro de la carpeta del proyecto del Parcial.
Se trata de un archivo de estructuras "stRegistro" que responde a la siguiente estructura de datos:

```
typedef struct{
    int idSucursal;
    char nombreSucursal[25];
    char nombreProducto[25];
    char deporte[25];
    int stockProducto;
}stRegistro;
```

Se nos pide desarrollar la siguiente funcionalidad:

1. **Hacer una función que recorra el archivo adjuntado y cargue los datos en una LISTA (simple) de LISTAS (simples) (LDL),** clasificando los datos de la siguiente manera:

En cada nodo de la **LISTA PRINCIPAL** se ubica c/u de las sucursales de la empresa, de acuerdo a la siguiente struct:
 	
`typedef struct{`
`  int idSucursal;`
`  char nombreSucursal[25];`
`}stSucursal;`
 	
(CREAR LA STRUCT QUE CORRESPONDA PARA EL NODO DE LA LISTA PRINCIPAL)
	En cada nodo de la lista principal también se encuentra una **SUB-LISTA** con todos los productos que tiene en stock cada sucursal, de acuerdo a la siguiente struct:
	
	`typedef struct{`
	`  char nombreProducto[25];`
	`  char deporte[25];`
	`  int stockProducto;`
	`}stProducto;`
	
	(CREAR LA STRUCT QUE CORRESPONDA PARA EL NODO DE LA SUB-LISTA)

   Para facilitar la tarea, en el espacio de entrega del parcial dentro del campus se les adjunta el archivo "funcionesLista.c", en el cual se encuentran las funciones básicas para el manejo de listas. Deberán copiar del mismo a su proyecto las funciones que necesiten, y realizar las adaptaciones a las mismas que sean necesarias para trabajar con las estructuras de la Lista de Listas.
   Luego deberán codificarse (**modularizando adecuadamente**) las funciones que sean necesarias para:
   - Copiar los datos correspondientes a la estructura stSucursal y a la estructura stProducto.
   - Buscar la sucursal a la cual corresponde el producto a cargar en la LDL
   - Agregar una nueva sucursal si ésta no existe **(agregar en el lugar de la lista donde resulte más eficiente realizarlo).**
   - Agregar un nuevo producto a la sucursal que corresponda **(agregar en el lugar de la lista donde resulte más eficiente realizarlo)**

   (30 puntos)

2. Realizar las funciones necesarias para **mostrar todos los elementos de la estructura compuesta**, permitiendo visualizar cada sucursal con sus productos en stock.

- Como ayuda, se nos da la porción de código correspondiente al mostrado de una sucursal:
  printf("Id de la Sucursal.....: %d ",p.idSucursal);
  printf("\nNombre de la Sucursal.....: %s ",p.nombreSucursal);

- y de un producto:
  printf("\nNombre del Producto.....: %s ",p.nombreProducto);
  printf("\nDeporte al que pertenece.....: %s ",p.deporte);
  printf("\nStock del producto.....: %d \n",p.stockProducto);
  Modularizar adecuadamente pensando en la responsabilidad de cada TDA.

(8 puntos).

3. Sumar el stock total de un producto a elección del usuario en TODAS las sucursales de la empresa.

- La búsqueda para sumar el stock se realizará por dos campos a la vez: "nombreProducto" y "deporte" (por ejemplo: sumar stock de "calza" de "running").
- **Modularizar adecuadamente pensando en la responsabilidad de cada TDA.**

(15 puntos).

4. Realizar las funciones necesarias para recorrer la estructura compuesta y **pasar a un ARBOL (llamado "menorStock") todos los productos existentes en c/u de las sucursales cuyo stock sea menor a una cifra a elección del usuario.**

- **Los productos se insertarán en el árbol en orden alfabético de acuerdo al campo "nombreProducto"**
- Ojo: **puede haber productos con nombres repetidos** (ej: "calza" puede estar repetido porque hay calzas de running, de ciclismo, etc, y también puede estar en distintas sucursales). **Tener en cuenta dónde se guardarán los repetidos**.
- Los productos no deberán ser borrados de la estructura compuesta
- Como dato en el nodo del árbol, deberá utilizarse la estructura "stRegistro", ya que además de los datos del producto, necesitamos saber a qué sucursal pertenece.
- **Modularizar adecuadamente pensando en la responsabilidad de cada TDA.**

(20 puntos).

5. Hacer una función que busque en el ARBOL, y retorne si existe o no en él, un producto a elección del usuario.

- Por parámetro ingresarán el nombre del producto, el deporte al cual corresponde, y el id de la sucursal a la cual pertenece.
- Ojo! recordar que el árbol está ordenado de acuerdo al campo "nombreProducto", y que ahora hay que buscar por tres campos: "nombreProducto", "deporte" e "idSucursal". **Pensar bien cómo puede buscarse por los tres campos pero sin tener que buscar en todo el árbol, para no perder la eficiencia que tienen las búsquedas binarias en los Arboles Binarios de Búsqueda.**

(17 puntos)

6. Hacer una función main()

- Para hacer esto, cree las variables que considere necesarias e invoque las funciones (de forma directa o indirecta) como corresponde en cada caso.
- Muestre los resultados cada vez que sea necesario.
- A fin de identificar cada inciso, comente su código indicando a qué apartado corresponde, por ejemplo: // Ejercicio 3.a

(10 puntos)