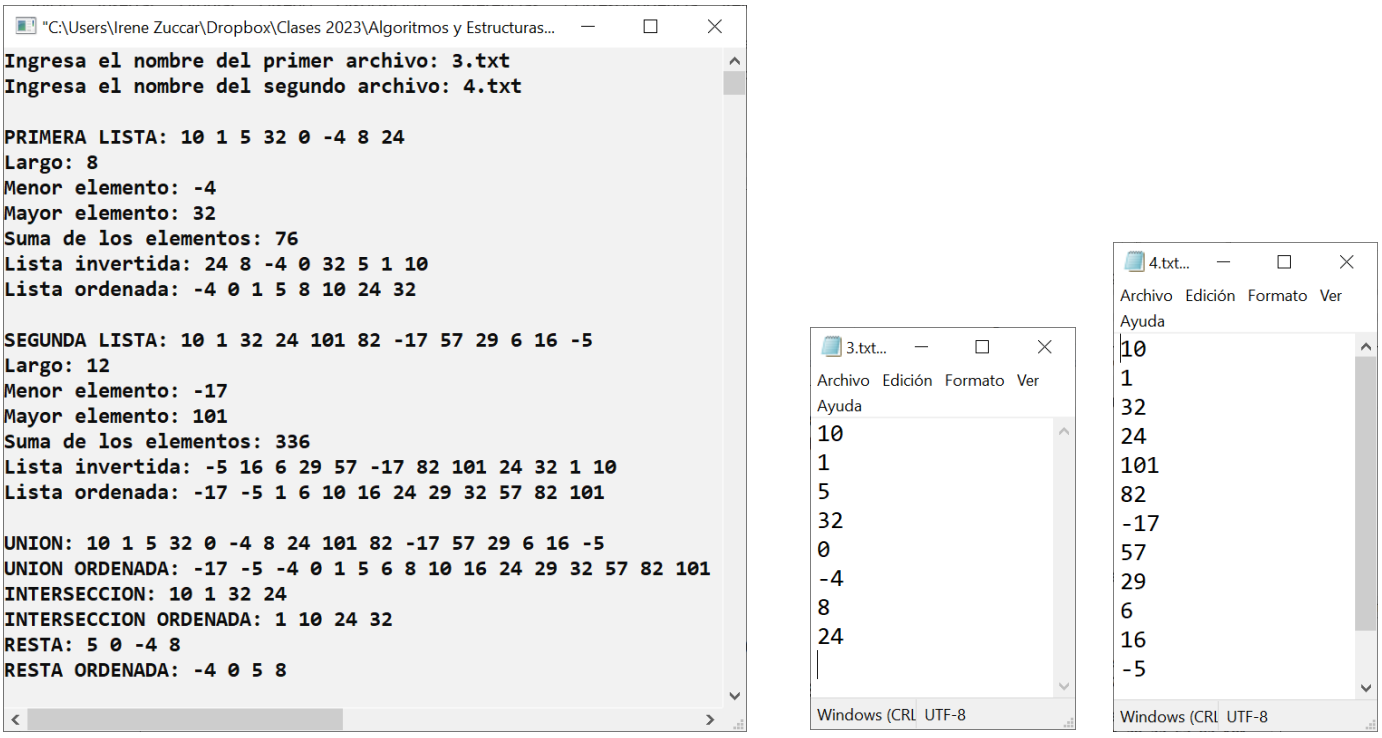
TAREA Nº1
PROGRAMACIÓN EN C: LISTAS.

I. OBJETIVOS.

El presente enunciado tiene como objetivo: Diseñar e implementar operaciones sobre listas de números enteros para el manejo interno de información.

II. INTERFAZ.

La interfaz solicitada para esta tarea se ejemplifica en la figura 1.a. En las figuras 1.b y 1.c, aparecen los archivos de entrada utilizados: Cada uno debe contener al menor un valor entero.



(a) **(b)** **(c)**

Figura 1. (a) Ejemplo de la interfaz de esta tarea (b) y (c) ejemplo de los archivos de entrada.

La estructura para definir un nodo de una lista debe ser la mostrada en la figura 2. Como puedes observar en la línea 10 se define un alias para el tipo de dato `struct nodo`, llamado `tNodo` y en la línea 11 se define un alias para el tipo de dato puntero a un nodo, llamado `Lista`.

```
5  struct nodo{
6      int info;
7      struct nodo *sig;
8  };
9
10 typedef struct nodo tNodo;
11 typedef tNodo *Lista;
```

Figura 2. Definición de la estructura para un nodo de una lista de enteros.

Las funciones que debes construir para cumplir con la interfaz establecida son:

- `Lista leeArchivo(char nombreArchivo[50]):` recibe un nombre de archivo, y almacena los enteros que posee el archivo en el mismo orden en una lista que luego retorna.
- `Lista creaNodo(int dato):` recibe un entero llamado `dato`, y crea y retorna un puntero a un nuevo nodo creado, cuyo campo `info` almacenará a `dato` y el puntero `sig` apunta a `NULL`.
- `Lista insertaFinal(Lista L, int dato):` recibe una lista `L` y un entero `dato`, y genera un nuevo nodo que almacena a `dato`, e inserta el nuevo nodo al final de la lista `L`, retornándola.
- `void imprimeLista(Lista L):` recibe una lista e imprime los números que almacena separados por un espacio. Si la lista estuviese vacía, debe imprimir un "-".
- `Lista insertaInicio(Lista L, int dato):` recibe una lista `L` y un entero `dato`, y genera un nuevo nodo que almacena a `dato`, e inserta el nuevo nodo al principio de la lista `L`, retornándola.
- `Lista insertaOrdenadoLista(Lista L, int dato):` recibe una lista `L` y un entero `dato`, y genera un nuevo nodo que almacena a dato, e inserta el nuevo nodo en forma ordenada en la lista `L`, retornándola.
- `Lista ordenaLista(Lista L):` recibe una lista `L` y, a partir de ella, genera una nueva lista pero con sus valores ordenados. Retorna la nueva lista.
- `int largoLista(Lista L):` recibe una lista y retorna la cantidad de elementos que almacena.
- `int menorLista(Lista L):` recibe una lista y retorna el menor valor que almacena.
- `int mayorLista(Lista L):` recibe una lista y retorna el mayor valor que almacena.
- `int sumaLista(Lista L):` recibe una lista y retorna la suma de todos los elementos que almacena.
- `Lista invierteLista(Lista L):` recibe una lista `L` y, a partir de ella, genera una nueva lista pero con sus valores invertidos. Retorna la nueva lista.
- `bool perteneceLista(Lista L, int dato):` recibe una lista `L` y un `dato` entero, y retorna `true` si el `dato` está almacenado en `L`, y `false` si no lo está.
- `Lista unionLista(Lista L, Lista M):` recibe dos listas y, a partir de ellas, genera una nueva lista con los elementos de ambas listas, pero sin elementos repetidos. Retorna la nueva lista.
- `Lista interseccionLista(Lista L, Lista M):` recibe dos listas y, a partir de ellas, genera una nueva lista con los elementos en común de ambas listas. Retorna la nueva lista.
- `Lista restaLista(Lista L, Lista M):` recibe dos listas L y M y, a partir de ellas, genera una nueva lista con los elementos que pertenecen a L pero que no pertenecen a M. Retorna la nueva lista.
- `Lista borraLista(Lista L):` recibe una lista y libera la memoria de cada nodo que posee, retornando finalmente `NULL`.

III. Actividades de los talleres y autónomo.

En la **primera clase de taller** asociado a esta tarea desarrollarás junto a tu profesor las siguientes funciones: `leeArchivo`, `creaNodo, insertaFinal, imprimeLista` y `largoLista.`

En la **segunda clase de taller** asociado a esta tarea desarrollarás junto a tu profesor las siguientes funciones: `insertaInicio, invierteLista, menorLista, perteneceLista` e `interseccionLista.`

Autoestudio:

- Para tu autoestudio deberás desarrollar el resto de las funciones que no se realicen en taller.
- Recuerda que en tus clases de cátedra se te han entregado diferentes algoritmos que te pueden servir para ejercitar tanto para la **evaluación de esta tarea como para la Solemne N°1.**
- **Recuerda contactar a tu profesor de taller por cualquier duda que tengas en este proceso.**

IV. Sobre Buenas Prácticas de Programación.

1. Debes usar identificadores representativos para tus constantes, variables, parámetros de entrada y funciones.
2. Las variables locales a cada función **debes** definirlas al **principio** de la función. Puedes darle valores iniciales al momento de definirlas (esto se llama "inializarlas").
3. Tu código debe estar correctamente *indentado* (uso de sangrías para cada sub-bloque de instrucciones), esto incluye el correcto alineamiento de las llaves ("`{`" y "`}`") que delimitan tales bloques.
4. Tu código no puede presentar más de 1 línea en blanco.
5. Tu código no puede poseer instrucciones "basura".
6. Debes comentar cada una de las funciones que definas como se indicó en la tarea 0.

IV. Sobre la Evaluación.

En el taller que **corresponda la evaluación de esta tarea** (tercer taller luego de iniciada la Tarea N°1) se pedirá que escribas en **papel** solo una función que realice alguna tarea sobre el arreglo que aprendiste a manejar en los talleres previos.