UTN FRMDP - TUP - Laboratorio 2 – Recuperatorio Primer Parcial – 21 de octubre de 2022.-

IMPORTANTE:

- **Crear un proyecto con su Apellido y nombre.**
- **Realizar todas las funciones que se indican, añadiendo comentarios a su código identificando con el número de ejercicio e inciso TANTO A LA/S FUNCIÓN/ES VINCULADAS CON SU RESOLUCIÓN, COMO A LA PARTE DEL MAIN CORRESPONDIENTE AL EJERCICIO E INCISO.**
- **SI NO SE IDENTIFICA, LA RESOLUCIÓN NO TENDRÁ VALIDEZ.**

Una empresa que patrocina a distintos Youtubers con canales dedicados a juegos nos ha encomendado realizar un pequeño sistema.
Los datos para trabajar se encuentran almacenados en el **archivo subido al campus**, el cual hay que descargar y copiar dentro de la carpeta del proyecto del Parcial. Se trata de un archivo de estructuras "stYoutuber" que responde a la siguiente estructura de datos:

```
typedef struct {
    int id;
    char nombreCanal[40];
    char rubro[30];            //los rubros de juegos a los que se dedica cada canal son (y estan escritos
                               // tal cual: "fantasia","creativos","accion","aventura","retro","construccion"
    int cantSuscriptores;      //cant suscript total del canal
    int cantVistasSemestre;    //cant de vistas que recibieron sus videos en el último semestre
} stYoutuber;
```

Tenemos que desarrollar la siguiente funcionalidad:

1. Deberá realizarse una **UNICA FUNCION PRINCIPAL** que recorra POR UNICA VEZ el archivo de Youtubers, y con los datos leídos del mismo codifique las funciones auxiliares necesarias para **construir dos listas**:

A) una **LISTA SIMPLEMENTE ENLAZADA con los Youtubers correspondientes a los rubros "retro", "construccion" o "creativos"**. Para realizar esto, se deberá:

- crear la estructura necesaria,
- inicializar lo que corresponda,
- y codificar las funciones que sean necesarias, **modularizando de forma correcta.**
- Los datos deben ser agregados a la lista **en orden de acuerdo a la cantidad de suscriptores**

B) **una LISTA DOBLEMENTE ENLAZADA con todos los Youtubers correspondientes a los restantes rubros.** Para ello se deberá:

- crear la estructura necesaria,
- inicializar lo que corresponda,
- y codificar las funciones que sean necesarias, **modularizando de forma correcta.**
- los datos deben ser agregados **en el lugar de la lista en el cual resulte más EFICIENTE realizar el proceso de agregado** (al final?, al principio?, o en orden?),
- **TODAS LAS FUNCIONES NECESARIAS PARA CONSTRUIR ESTA LISTA DOBLE (todas las de este apartado B) DEBEN UTILIZAR COMO PARAMETRO PUNTEROS DOBLES.**

Para ahorrar tiempo, ya se nos ha dado el código correspondiente a las estructuras de los nodos de ambas listas:

```
typedef struct nodoLS {
    stYoutuber dato;
    struct nodoLS* sig;
}nodoLS;
```

```
typedef struct nodoLD{
    stYoutuber dato;
    struct nodoLD* sig;
    struct nodoLD* ante;
}nodoLD;
```

(7 puntos función principal, correcta invocación de las funciones auxiliares y correcta modularizacion de las mismas,
12 puntos funciones de manejo de listas simples,
12 puntos funciones de manejo de listas dobles, y
10 puntos manejo de punteros dobles).

2. En relación a la **LISTA SIMPLEMENTE ENLAZADA** obtenida en el primer punto, realizar una función que permita modificar el campo cantVistasSemestre de un Youtuber a elección del usuario. Para ello debemos realizar lo siguiente:

• Una **función auxiliar <u>RECURSIVA</u>** que **busque** el nodo de la lista en el cual se encuentra el Youtuber a modificar.

• Una **función principal** que invoque a la anterior y **modifique** el campo cantSuscriptores del Youtuber elegido reemplazándolo por el nuevo dato (pensar en qué lugar de nuestro código debemos pedir al usuario que ingrese el Youtuber a modificar y el nuevo dato de la cantidad de suscriptores).

(16 puntos)

3. En relación a la **LISTA SIMPLEMENTE ENLAZADA** obtenida en el primer punto, realizar una **<u>FUNCION RECURSIVA</u>** que permita **contar** la cantidad de Youtubers que **superan** los 10.000 suscriptores

(16 puntos).

4. En relación a la **LISTA SIMPLEMENTE ENLAZADA** obtenida en el primer punto, realizar las funciones que sean necesarias para construir una **<u>PILA</u>** en la cual se guarden los Youtubers de dicha lista que pertenezcan al rubro "**construcción**".

- **Los nodos <u>no deben ser borrados de la Lista Simple Original</u>, <u>la Lista original debe permanecer sin alterar.</u>**
- La estructura de la **<u>PILA</u>** es la siguiente, y se encuentra **<u>IMPLEMENTADA CON LISTAS SIMPLES</u>,** por lo cual **deberán invocarse las funciones que sean necesarias ya desarrolladas en el apartado 1. A) (respetar la "responsabilidad de cada librería", no codificar en las funciones de Pilas cosas que ya fueron codificadas para listas):**

```
typedef struct{
    nodoLS * lista;
}Pila;
```

(17 puntos)

5. 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.
- **Las dos listas creadas en el ejercicio 1 y la Pila creada en el ejercicio 4 deberán ser mostradas codificando las funciones correspondientes**.

  Para ahorrar tiempo, ya se nos ha dado la porción de código correspondiente al mostrado por pantalla de los datos de UN YOUTUBER:

```
printf("\n ***************************** YOUTUBER  ********************************** \n";
printf("Id del Youtuber.....: %d ",p.id);
printf("\nNombre del Canal.....: %s ",p.nombreCanal);
printf("\nRubro del Canal (tipo de juegos).....: %s ",p.rubro);
printf("\nCantidad de Suscriptores.....: %d ",p.cantSuscriptores);
printf("\nCantidad de  vistas del semestre ….....: %d \n\n",p.cantVistasSemestre);
```

- A fin de identificar cada inciso, comente su código indicando a qué apartado corresponde, por ejemplo: // Ejercicio 3.a

(10 puntos)