

# Labos: Ejercicios de final

—

## Ejercicio 1

En un stream especial, Luquita Rodríguez y Davo están revisando canciones que manda el chat para encontrar un nuevo tema para el mundial. La idea es superar al tema "Pa' la Selección" de La T y la M. Ya no se puede vivir solo de los clips gritando "ESTA LOCURAAAAAAAA" cada vez que aparece Messi.

Los viewers mandan sus canciones a una bandeja de entrada (TDA), pero varias son un desastre. De estas canciones malas, algunas pueden salvarse si se les dedica un poco de tiempo, pero otras son inescuchables y hay que borrarlas para siempre.

Se sabe que las canciones desastrosas que pueden salvarse con un poco más de esfuerzo tienen una puntuación mayor a 5.

Se pide

Crear una función que reciba una instancia cargada de un TDA que representa la bandeja de entrada del stream que contiene todas las canciones y un vector dinámico de canciones en el que se guardarán las canciones malas que pueden salvarse. La función debe quitar de la bandeja de entrada las canciones malas, guardando en el vector dinámico aquellas que pueden salvarse con tiempo y dedicación y eliminando las que no, manteniendo el estricto orden de reproducción de las demás.

```
typedef struct cancion {
    char autor[MAX_AUTOR];
    int puntuacion;
} cancion_t;

struct bandeja_stream;

typedef struct bandeja_stream bandeja_stream_t;

// Pre: -
// Post: * Crea un TDA bandeja_stream en el heap y devuelve un puntero
al mismo.
```

```

//      * Devuelve NULL si no la pudo crear.
bandeja_stream_t* bandeja_stream_crear();

// Pre: * La bandeja_stream fue previamente creada con
bandeja_stream_crear.
//      * La canción ya está completamente inicializada. // Post: Agrega
una nueva cancion al final de la bandeja_stream.
void insertar_cancion(bandeja_stream_t* bandeja_stream, cancion_t
cancion);

// Pre: La bandeja_stream fue previamente creada con
bandeja_stream_crear.
// Post: Quita la cancion del principio de la bandeja_stream y la
devuelve
// mediante la referencia cancion_quitada o NULL si la bandeja_stream
// estaba vacia.
void quitar_cancion(bandeja_stream_t* bandeja_stream, cancion_t*
cancion_quitada);

// Pre: La bandeja_stream fue previamente creada con
bandeja_stream_crear.
// Post: Devuelve true si la bandeja_stream no contiene canciones
// o false en caso contrario.
bool bandeja_stream_esta_vacia(bandeja_stream_t* bandeja_stream);

// Pre: La bandeja_stream fue previamente creada con
bandeja_stream_crear.
// Post: Libera la memoria que se reservó en el heap para la
bandeja_stream.
void bandeja_stream_destruir(bandeja_stream_t* bandeja_stream);

```

## Ejercicio 2

Luego de una exhaustiva noche de concentración en el predio de Kansas, queda actualizar la tabla de posiciones del torneo interno de truco (tabla\_truco.csv) con los puntos de la última partida para saber en qué puesto quedó cada dupla y así consagrar a los ganadores.

Se sabe que el archivo de los puntos de la partida solo contiene el nombre de la dupla por cada vez que sumó un punto, mientras que la tabla de posiciones histórica tiene el siguiente formato:

dupla;puntos

Se pide

Crear una función que genere un nuevo archivo `podio_truco.csv` donde se escriba el top 3 de las duplas con los puntos actualizados.

Aclaración

- Se sabe que no puede haber más de 30 puntos en una partida (juegan a 15).

Ejemplo

Sabiendo que la dupla Messi-De Paul sumó 3 puntos y Alexis-Gonzalez sumó 15:

`puntos_partida.csv`

Alexis-Gonzalez  
Alexis-Gonzalez  
Alexis-Gonzalez  
Messi-De Paul  
Alexis-Gonzalez  
Alexis-Gonzalez  
Alexis-Gonzalez  
Messi-De Paul  
Alexis-Gonzalez  
Messi-De Paul  
Alexis-Gonzalez  
Alexis-Gonzalez  
Alexis-Gonzalez  
Alexis-Gonzalez  
Alexis-Gonzalez  
Alexis-Gonzalez  
Alexis-Gonzalez  
Alexis-Gonzalez

`tabla_truco.csv`

Messi-De Paul;57  
Cutí-Licha;35

Alexis-Gonzalez;32

Enzo-Julian;21

El archivo final `podio_truco.csv` debería quedar:

`podio_truco.csv`

Messi-De Paul;60

Alexis-Gonzalez;47

Cuti-Licha;35

Enzo-Julian;21

### Ejercicio 3

Terminó el Mundial y un coleccionista está desesperado por completar su álbum. Para organizarse mejor, modeló las hojas de su álbum y las figuritas repetidas que tiene guardadas en folios. Tiene los folios organizados en una grilla (matriz), donde cada folio guarda figuritas de una página específica.

Cada folio de la colección se puede representar con el siguiente *struct*:

```
typedef struct folio {
    figurita_t figuritas[MAX_FIGURITAS];
    int cantidad_figuritas;
    char tipo_papel; // 'M' (mate), 'B' (brillante) o 'E' (extra)
    char codigo_pagina[MAX_CODIGO];
    char seleccion[MAX_NOMBRE_SELECCION];
} folio_t;

typedef struct figurita{
    int numero_figurita;
    char nombre[MAX_NOMBRE];
    char posicion[MAX_POSICION]; //Ej 'Arquero', 'Defensor', 'Delantero'
} figurita_t;
```

Se pide

Crear un procedimiento que reciba una matriz que representa la grilla de folios del coleccionista y un vector vacío de códigos de página. El procedimiento deberá recorrer **recursivamente** la matriz y

guardar en el vector los `codigo_pagina` de todos los folios que cumplan las siguientes condiciones, para saber qué páginas ya puede ir a pegar:

- La página del folio pertenece a la selección de Argentina.
- Las figuritas que le faltaban eran "Enzo" o "Julian".
- Los números de figurita de estos jugadores en la edición de Panini están entre el 14 y el 21, inclusive.