**Consigna 1:**

Vamos a gestionar una lista de reproducción en ~~Winamp~~ Spotify. Para ellos vamos a codificar algunas funciones para simular el comportamiento en el mundo real, bah, el virtual pero en la vida real, se entiende. El primer paso es generar nuestro catálogo de canciones con la siguiente estructura

| **typedef struct stCancion {**  **int duracion;**  **char artista[50];**  **int cantidadReproducciones;**  **char genero[50];**  **char título[50];**  **int idCancion**  **char albmun[50];**  **}stCancion ;** | **typedef struct nodoCancion{**  **stCancion dato ;**  **struct nodoCancion \* siguiente;**  **}nodoCancion;** |
| --- | --- |

Para el funcionamiento de nuestro catálogo vamos a necesitar

* Agregar una canción, ordenadas por duración.
* Borrar una canción
* Sumar uno a la cantidad de reproducciones
* Buscar por título (solo devuelve una canción)
* Buscar por genero (podría devolver una sublista, quizás?)
* Buscar por álbum (podría devolver una sublista, quizás?)
* Buscar por título (podría devolver una sublista, quizás?)
* Mostrar todas las canciones

| **typedef struct fila{**    **struct nodoListaReproduccion\* primero;**  **struct nodoListaReproduccion\* ultimo;**  **}fila ;** | **typedef struct nodoListaReproduccion{**  **int idCancion;**  **struct nodoListaReproduccion\* siguiente;**  **struct nodoListaReproduccion\* anterior;**  **}nodoListaReproduccion;** |
| --- | --- |

Para el funcionamiento de la lista de reproducción usaremos las siguientes funciones

* Agregar una canción. El usuario primero la tiene que buscar según la info que quieran y se guarda solo el id en la lista de reproducción.
* Eliminar una canción. Quizás me equivoque y no quiero escucharla. Va por id.
* Calcular la duración de la playlist
* Dado un género, artista o álbum, contar la cantidad de canciones.
* Reproducir una canción (siempre es la primera)
  + Extraer el id
  + Sumar +1 a cantidadReproducciones del catálogo
  + Eliminar la canción
  + Avisarle al usuario que la lista finalizó y enviarlo al menú

**Consigna 2:**

La UTE nos pide gestionar en una app del chofer la gente que se sube al colectivo y recabar algunos datos.

| **typedef struct stPasajero{**  **int tipodeboleto; //1 común, 2 jubilado, 3 estudiante**  **float recorridoEnMetrosEstimado;**  **float precioBoleto;**  **char DNI[11];**  **}stPasajero ;** | **typedef struct nodoPasajero {**  **stPasajero dato ;**  **struct nodoPasajero\* siguiente;**  **}nodoPasajero;** |
| --- | --- |

Para gestionar la lista de pasajeros necesitaremos codificar

* Agregar un pasajero a la lista doble ordenada por el recorrido estimado que piensa hacer.
* Verificar si existe un pasajero por DNI
* Eliminar un pasajero por el número de DNI.
* Contar pasajeros por tipo de boleto enviado por parámetro.

Una vez que tenemos todos los posibles pasajeros de un colectivo, los vamos a pasar a la fila que se arma en la vereda.

| **typedef struct fila{**    **struct nodoFilaPasajero\* primero;**  **struct nodoFilaPasajero\* ultimo;**  **}fila ;** | **typedef struct nodoFilaPasajero{**  **char DNI[11];**  **struct nodoFilaPasajero\* siguiente;**  **struct nodoFilaPasajero\* anterior;**  **}nodoFilaPasajero;** |
| --- | --- |

El manejo de la fila lo vamos a distribuir de la siguiente manera

* Subir un pasajero al colectivo. Solo preguntar el DNI y verificar si existe un pasajero
* Bajar a un pasajero, siempre se baja el primero
  + Contabilizar cuánto ganó el colectivo en ese viaje.
  + Contar todos los km de la lista
  + Establecer un precio del litro de nafta y asumiendo que consumimos un litro por km, devolver el costo del viaje.