



## **UE 204**

# **Introduction to Computer Science 2**

# Implementation of Music Playlist.

#### Goals:

- 1. To understand the real life applications of Linked Lists.
- 2. To use Modular Programming.
- 3. To manipulate I/O Streams.
- 4. To draw plots.
- 5. To create a makefile.

### Task:

You should write a program that holds different playlists. In our context, the name Playlist is abstract, it is a **list of linked songs**.

# **Exercise 0: Modular Programming**

Read the topic, understand it. Define your vision of program in your head. Think of good structure to implement the program. Prepare modules for your program (song.c, song.h, playlist.c, playlist.h, main.c and etc).

### **Exercise 1: Create Data Structures**

Create 2 data structures with following parameters:

a) Song:

Name

Artist

Duration

Genre

Next Song

ID\_in\_playlist

b) Playlist:

Name

First Song Last Song

Number of songs

**Total Duration** 

# **Exercise 2: Implement the following functions**

Song\_New: function which creates new song.

Song\_Print: function which prints the name, artist, duration, genre and

ID\_in\_playlist of a song.

Note: you can add your own functions or sub-functions if you need them to implement a task.

**Playlist\_New**: function which creates new empty playlist with name.

**Playlist\_Add**: function which takes a song and add it to the playlist.

Playlist\_Add\_After: function which takes a song and add it after some song.

**Playlist\_Delete\_byName**: function which takes a name of song, and delete it from playlist.

**Playlist\_Delete\_byID**: function which takes an ID of song, and delete it from playlist.

**Playlist\_Find\_byName**: function which takes a name of song, and print its artist, duration, genre and ID\_in\_playlist.

**Playlist\_Find\_byID**: function which takes an ID of song, and print its name, artist, duration and genre.

**Playlist\_Sort\_byName**: function which sorts a playlist in the alphabetical order (by first letter only).

**Playlist\_Sort\_byID**: function which sorts a playlist in the ascending or descending order.

**Playlist\_Print**: function that prints the whole playlist.

# **Exercise 3: Save the playlist in the file**

**Playlist\_To\_File**: function that take the playlist, and save its songs with corresponding ID, Name, Artist, Duration, Genre in the file.

#### **Exercise 4: Plot the statistics**

Playlist\_Statistics: function which plot the number of songs for each genre.

- 1. Define a enum for genres.
- 2. Count how many songs in each genre
- 3. Plot the graph with:
  - X: genre (Ex: 1 = pop, 2 = rock, and etc)
  - Y: number of songs

### **Exercise 5: Main function**

Fill the main function so that there are 2 playlists:

- 1. Playlist created by user: when running a program it should interact with user and ask him to create a playlist with his favorite songs, getting all the necessary information (Song name, artist, duration and etc) from user.
- 2. Playlist created by yourself: create a playlist, name it, add some songs.

### **Exercise 6: Makefile**

Write a makefile with necessary target and dependencies to compile the program.