Amsterdam University College

Data Structure & Algorithms

Assignment 2.

1 MySongPlayer

Your company is developing a new program to stream songs. For this reason you were asked to implement some of the functionalities that allow to manage the playlists of this new software.

The main entity of the software you are developing are *songs*. A song consists of a struct which has three fields, *id*, *title*, and *duration*. Where *id* is an integer that uniquely identifies a song, *title* is the title of the song, and *duration* is a float that specifies the duration of the song in seconds.

This is a general explanation of how a playlist should behave: You can think of a playlist as a dynamic sequence of songs. Songs can be added to the end of the sequence using the function addSong. A playlist will be used by the program to keep track of which song should be played next. To do so the user program will call the playSong function. Unless the playlist was modified (see below) songs are usually played in the same order in which they were added.The user program is also allowed to skip songs using the skipSong function. Once all the song are played or skipped the playlist will restart playing from the fist song. The user program can also start play the playlist from any given position in the sequence using the function playFrom. Moreover, the playlist has two modes of play songs: *skip mode* in which songs that were played at least once before are skipped, and *normal mode* in which all the songs in the playlist are palyed unless they re explicitly skipped. The user program can change the play mode using the function skipAllPlayedSongs. Finally, the user program can modify the playlist using the function swapSongs which allows to change the order of two songs in the playlist.

Since your program will be part of a larger project the software engineer of your company developed an interface for the library that you will have to develop. You can find the interface of your library in the file "playlist.h" which also contains the *song* type. This is a screenshoot of the playlist interface:

```
/*OVERVIEW: is a dynamic sequence of songs. Songs can be inserted in the playlist and will be played in the same order in which they were inserted unless the prder og the sequence was modified using swapSongs, the restart, or playFrom.  Once all the songs have being played or skipped the playlist will restart from the first song in the list.
*/


typedef struct playlistType * playlist;

/* EFFECTS: allocates and initialises an empty playlist which will play in normal mode.*/
playlist initialisePlaylist();

/* EFFECTS: adds s as the last element of p.*/
void addSong(playlist p, song s);

/* EFFECTS: returns the ID of the next song of p to be played.*/
int playSong(playlist p);

/* EFFECTS: set p to play from the ith song. I.e., after the call the ith song will be the next song to be played.*/
void playFrom(playlist p, int i);

/* EFFECTS: if p is not empty, skips the next song which will not be played.*/
void skipSong(playlist p);

/* EFFECTS: if mode is 1 (skip Mode) then set the playlist p to play only songs that were not played before. If mode is 0 (normal mode) resets p and sets it to play all songs.*/
void skipAllPlayedSongs(playlist p, int mode);

/* EFFECTS: resets p making the first song in the sequence to be played the next to be played.*/
void restart(playlist p);

/* EFFECTS: if i and j are smaller than or equal to the number of songs in p, the function swaps the ith song and the jth song in p.*/
void swapSongs(playlist p, int i, int j);


/* EFFECTS: deallocate all the memory required by p.*/
void freePlaylist(playlist p);
```

You should implement the playlist interface. In your implementation you may assume that playlists are **very static**, i.e., most of the songs will be added right after the playlist is created and the addSong function will rarely be called after this initial phase.

Write a short (max 1 page) report where you explain an justify the following implementation choices: [5 pts]

- the data structures you used to implement a playlist. **[2 pt.]**
- your implementation of playSong and playFrom. **[3 pts.]**

Your program should consist of one C file called "yoursurname.c" which implements the interface "playlist.h". Your program will be compiled using "gcc -Wall -pedantic -std=c11 tester.c yoursurname.c -o test" where "tester.c" is a program that will randomly generate and use your playlist. Your program will then be tested 5 times with different testing parameters each passed test is worth 1 point **for a maximum of** 5 **points**. You can find the "tester.c" program on Canvas together with few examples of the expected output. You can upload your program on Canvas and give me your report in class.

Deadline on the 22nd of November at midnight.