CENG 311 - Programming Assignment 1

In this homework, you must write a C program to create an array of playlists. Each playlist is a simple linked list, with each node's data pointing to a song_t object. You are expected to modify the source code provided with this homework document(also in the second and third page).
You:

- are required to replace ???????? with correct data types.
- are required to implement all functions.
- cannot define any other functions, global variables, and structs; and cannot modify the fields of structs, the data types in any function signatures(except ????????).
- all memory allocations done in given functions must be on the heap area by using malloc, calloc, realloc.
- must use free to destroy the objects.

In the main function, you must:

- initialize array_of_playlist_ptrs by using create_array_of_linked_list_ptrs with an initial size of 5.
- create 5 playlists by using create_link_list and insert them into array_of_playlist_ptrs by using set_element_of_array_of_linked_list_ptrs. Then, resize array_of_playlist_ptrs to a size of 10 by using resize_array_of_linked_list_ptrs, create five more playlists, and insert them into available areas in array_of_playlist_ptrs.
- create four songs for each playlist by using create_song and insert them into playlists by using add_to_linked_list. Each song must have a name and duration.
- remove the second song of each playlist by using remove_from_linked_list and destroy them.
- print all playlists and their contents by using get_element_of_array_of_linked_list_ptrs.
- destroy each playlist by using destroy_linked_list.
- destroy array_of_playlist_ptrs by using destroy_array_of_linked_list_ptrs.

Moreover, you must implement the functions while adhering to the following rules:

- initially, when you initialize an array in create_array_of_linked_list_ptrs, assign NULL to each element.
- when you initialize a linked list in create_link_list, assign NULL to the fields.
- do not destroy the elements of the array in destroy_array_of_linked_list_ptrs; only destroy the array itself.
- do not destroy the *data* inside linked list nodes in destroy_linked_list; only destroy the nodes.
- when you expand the array size using resize_array_of_linked_list_ptrs, assign NULL to each newly created element.
- in add_to_linked_list function, if the *head*'s *data* is NULL, then emplace *data* into that field; if not, emplace data into the newly created node at the end of the list.
- you cannot use `song_t` data type in the functions except main and create_song

Hints:

- You can use rand to generate random data.
- In the solution source code whe have used the following pointer types:

    - linked_list_node_t*
    - linked_list_node_t**
    - linked_list_node_t***
- You can use another linked_list_node_t to save the playlists' 2nd nodes' data's addresses.

Submission:

- Change the source code name to your student id, and upload it e.g. 280000000.c
- You must only upload the .c source code file.

Template:

```c
#include <stdio.h>
#include <stdlib.h>

typedef struct linked_list_node_t {
   void* data;
   struct linked_list_node_t* next;
} linked_list_node_t;

void create_array_of_linked_list_ptrs(???????? destination, int size) {
   //Fill this body
}

void get_element_of_array_of_linked_list_ptrs(???????? array, int index,
???????? destination){
   //Fill this body
}

void set_element_of_array_of_linked_list_ptrs(???????? array, int index,
???????? head) {
   //Fill this body
}

void destroy_array_of_linked_list_ptrs(???????? array) {
   //Fill this body
```

```
}

void resize_array_of_linked_list_ptrs(???????? destination, ????????
array, int size, int new_size){
   //Fill this body
}

void create_link_list(???????? destination){
   //Fill this body
}

void destroy_linked_list(???????? head) {
   //Fill this body
}

void add_to_linked_list(???????? head, const void* data) {
   //Fill this body
}

void remove_from_linked_list(???????? destination, ???????? head, const
void* data) {
   //Fill this body
}

typedef struct song_t {
   const char* name;
   float duration;
} song_t;

void create_song(song_t* destination, const char* name, float duration)
{
   //Fill this body
}

???????? array_of_playlist_ptrs = NULL;

int main(void){
   //Fill this body
```

```
    return 0;
}
```