HOMEWORK 1

In this homework you are supposed to write a C program in which you can create a simple song list. You must use given data structures and implement given methods properly. The user of the program must be able to

- add a song to the list by providing its name and duration
- delete a song from the list by providing its name
- list all songs

Your implementation must satisfy these constraints below

- You must use the *dynamic_array* data structure for storing songs. The initial *capacity* of the *dynamic_array* must be 2, and you must allocate 2 *void** in the heap by using *malloc()* and assign *NULL* value to them. *elements* pointer must store the returned address from *malloc()*. Implement these operations in the *init_array* function.
- When the user chooses to add a new song you must create that song in the heap by using *malloc().* You must put this song address into the songs by using the *put_element* function. In the *put_element* function you must increase the size of the *elements* array and put the newly added element's address into the *elements* array. Emplace it to the first available position.
- When the user chooses to delete the song, you must find it by using *get_element* function, and use *free()* function to deallocate them from the heap. And use *remove_element* to remove its address from the songs.
- Everytime the *size* of the *dynamic_array* reaches *capacity/2* you must increase the *capacity* to 2 times the old one, and copy the elements of the *elements* array into the new allocated *elements* array. You must assign *NULL* value for *elements* array elements that haven't pointed to any valid songs yet. All these operations must be implemented in the *put_element* function.
- Everytime the size of the *dynamic_array* drops down to *capacity/2,* reduce *capacity* by factor of 2 and allocate space for that *capacity*, copy the values of the *elements* array to the newly allocated *elements* array and deallocate the old *elements* array by using *free()* function. All these operations must be implemented in the *remove_element* function.
- While you list the songs you must use the *get_element* function to get the song address at that position. Note that get_element returns *void*,* so you can cast the type of *void** to *song**, and then access the song fields.

Notes:

- The *elements* field of dynamic_array is a pointer that stores the address of elements rather than elements as values. You can regard *elements* as an array of *void*.*
- The heap mentioned above is not a data structure, it is the memory region. You can create or manipulate an object on the heap by using dynamic memory allocation functions(e.g. *malloc()*, *free()* etc.) declared in *stdlib.h*
- *NULL* is also declared in stdlib.h, it is used as an address that points to *0* which means there is no valid data at that address, or the data hasn't been allocated yet. To indicate the pointer hasn't been initialized yet or the pointed value deallocated, you can use *NULL.*

```
char* myptr = NULL;
```

- void* is a general purpose pointer that helps you to store an object without knowing the actual pointed data type. You can cast it to the actual pointed data type. E.g.

```
void* myptr = malloc(sizeof(int)); //Allocates 4 bytes space on
the heap
int* my_num_ptr = (int*) myptr; //Cast it to int*, we are allowed
to do since we allocate 4 bytes
*my_num_ptr = 25; // Change the value at that address through our
int*
printf("%d", *my_num_ptr);  //Prints 25
printf("%d", *((int*)myptr)); //Prints 25
```

Submission details:

- You are supposed to submit a C source file named <your_id>_hw1.c
- No collaboration is allowed.
- Copy the template below and fill the necessary fields.
- You are allowed to define additional functions for your own usage.

The template:

```
typedef struct dynamic_array {
   int capacity;
   int size;
   void** elements;
} dynamic_array;

void init_array(dynamic_array* array) {
   //Fill this body
}
void put_element(dynamic_array* array, void* element) {
   //Fill this body
}
void remove_element(dynamic_array* array, int position) {
   //Fill this body
}
void* get_element(dynamic_array* array, int position) {
   //Fill this body
}
```

```c
typedef struct song {
   char* name;
   float duration;
} song;

int main() {
   //Fill this body
   return 0;
}
```