```c
// Name: Rodrigo Ignacio Rojas Garcia
// Course Number: ECE 2230
// Section: 001
// Semester: Spring 2017
// Assignment Number: 2

// Library Declartion Section
#include <stdio.h>
#include <stdlib.h>
#include "guitar.h"
#include "guitardb.h"
#include "list.h"

// Allocates dynamic memory for the a new list data base for guitars which will be used as an intermmediate between guitar.c and list.c and lab2.c
guitardb_t guitardb_init(void)
{
    guitardb_t new_list;
    new_list = (guitardb_t)calloc(1,sizeof(struct guitardb_s));
    new_list->list = list_init();
    return new_list;
}

// Function guitardb_add will first check if there has been an item addded to the list, if not true, it will prompt the user to fill the item characteristics with guitar_fill
// and create a new database on the list with list_append. If an item in the list, then it will check if either the item's id number is the same, greater, or less than the
// items already store in the list. If the id numbers are the same, it means that the item has already been added therefore it returns a -1. If the id number of the item
// passed in the function is greater than the current one on the list, it will call list_next and return the next item on the list and compare until it reaches NULL. If the item passed
// in the function is less than the next one on the list, it will return a -1, and guitar_fill will be called to fill the items characteristics and then list_insert_before will be
// called to add the item before the current item returned. If the id number of the item is greater than all of the ones already on the list, the variable temp_item will return
// a NULL exiting the while loop and calling guitar_fill to enter the item characteristics and then call list_append to set the item at the end of the list
key_t guitardb_add(guitardb_t database, guitar_t item)
{
    // Variable Declartion Section
    guitar_t temp_item;
    int comparison_result = -2;
    // Returns the first item in the list
    temp_item = list_first(database->list);
    // If no item on the list, it will prompt the user for item characteristics and insert it on the list
    if (temp_item == NULL)
    {
        guitar_fill(item);
        list_append(database->list, item);
        return 0;
    }
    // If there is at least one item on the list this will run
    else
    {
        // While loop runs as long as item returned from the list is not NULL
        while(temp_item != NULL)
        {
            // Compares the id numbers of item passed on the function and the item that was returned from the list
            comparison_result = guitar_compare(item, temp_item);
            // If id numbers equal, returns a -1, meaning item is already on list
            if (comparison_result == 0)
            {
                return -1;
            }
            // If the item id number is greater than the one returned it will call for the next item on the list
            else if (comparison_result == 1)
            {
                temp_item = list_next(database->list);
            }
            // If the item id number is less than the one returned from list, it will prompt user for the item characteristics and insert it before
            // the current item returned
            else if (comparison_result == -1)
            {
                guitar_fill(item);
                list_insert_before(database->list, item);
                return 0;
            }
        }
    }
    // If id number of item passed in the function is greater than all of the ones on the list, will pronmpt the user for item characteristics and
    // set it at the end of the list
    guitar_fill(item);
    list_append(database->list, item);
    return 0;
}

// Function gutiardb_lookup will creates two variables, temp_item and return_item. Variable temp_item will be used to compare the id number
// of items already on list, and if there is, it will return the address to the variable return_item. If the return is not NULL, it means
// that an item with exact id number is in list and returns the address, if not, NULL is returned
guitar_t guitardb_lookup(guitardb_t list, key_t id_number)
{
    guitar_t temp_item, return_item;
    temp_item = guitar_init();
    guitar_setid(temp_item, id_number);
    return_item = list_find(list->list, temp_item, (cmpfunc)guitar_compare);
    if (return_item != NULL)
    {
        guitar_free(temp_item);
        return return_item;
    }
    else
    {
        guitar_free(temp_item);
        return NULL;
    }
}

// Function gutiardb_report goes through the whole list using list_first to check that there is at least one item on the list, and then enters a while loop
// printing the first item and then checking for the next item on the list with list_next and if it does not return a NULL it prints it and it goes on until
// it reaches the end of the list which returns a NULL
void guitardb_report(guitardb_t list)
{
    guitar_t item;
    item = list_first(list->list);
    while(item != NULL)
    {
        guitar_print(item);
        item = list_next(list->list);
```

```
    }
}

// Function guitardb_delete will create a variable called item and will set it to t
he function guitardb_lookup which will return the address of the item that
// is being serached for if is in list, if true, it will call list_remove which wil
l remove the item from the list and return the address of it. If not found
// returns a NULL
guitar_t guitardb_delete(guitardb_t list, key_t id_number)
{
    guitar_t item;
    item = guitardb_lookup(list, id_number);
    if (item != NULL)
    {
        list_remove(list->list);
        return item;
    }
    else
    {
        return NULL;
    }
}

// Function gutiardb_finalize will first check if there is at least one item on the
 list, if so it will enter a while loop which will free the first item on the list
// then it will return the next one if there is one and will free it, this goes on
until the end of the list is reached which will return a NULL. Then it will call
// function list_finalize which will free all databases on the list and at the end
it will free the list
void guitardb_finalize(guitardb_t database)
{
    guitar_t temp_item;
    temp_item = list_first(database->list);
    while (temp_item != NULL)
    {
        guitar_free(temp_item);
        temp_item = list_next(database->list);
    }
    list_finalize(database->list);
    free(database);
}
```