```c
/* guitardb.c
 * Christopher Brant
 * cbrant
 * ECE 2230
 * Section 001
 * Spring 2017
 * Programming Assignment #2
 * Due on 2/15/17 at 11:59 PM
 * Professor Walt Ligon
 */

#include <stdio.h>
#include <stdlib.h>
#include "list.h"
#include "guitar.h"
#include "guitardb.h"

#define MAXLINE 5


/* Initialize a new guitar database */
guitardb_t guitardb_init(void)
{
        guitardb_t db_new = (guitardb_t)malloc(sizeof(struct guitardb_s));

        db_new->dbsize = 0;
        db_new->dblist = list_init();

        return db_new;
}

/* Add a guitar to the database */
key_t guitardb_add(guitardb_t dbpoint, guitar_t gpoint)
{
        char line[MAXLINE];
        int slot_found, keynum = 0;
        guitar_t match, rover, check1;

        while (keynum < 1)
        {
                printf("Enter your desired item key number, greater than 0: ");
                fgets(line, sizeof(line), stdin);
                sscanf(line, "%d", &keynum);
                printf("\n");
        }

        guitar_setid(gpoint, keynum);
        match = (guitar_t)list_find(dbpoint->dblist, gpoint, (cmpfunc)guitar_compare);

        if (match != NULL)
                return -1;

        guitarfill(gpoint);

        if (dbpoint->dbsize == 0)
        {
                list_insert(dbpoint->dblist, gpoint);
                dbpoint->dbsize++;
        }

        else
        {
                rover = list_first(dbpoint->dblist);
                check1 = rover;

                while (rover != NULL && slot_found != 5)
                {
                        slot_found = guitar_compare(rover, gpoint);

                        if (slot_found != 1)
                        {
                                if (check1 == rover)
                                {
                                        list_insert(dbpoint->dblist, gpoint);
                                        slot_found = 5;
                                        dbpoint->dbsize++;
                                }
                                else
                                {
                                        list_insert_before(dbpoint->dblist, gpoint);
                                        slot_found = 5;
                                        dbpoint->dbsize++;
                                }
                        }

                        else
                                rover = list_next(dbpoint->dblist);
                }

                if (rover == NULL)
                {
                        list_append(dbpoint->dblist, gpoint);
                        dbpoint->dbsize++;
                }
        }

        return 1;
}

/* Lookup a guitar by ID and return a pointer to it */
guitar_t guitardb_lookup(guitardb_t dbpoint, key_t keynum)
{
        guitar_t match_guitar;
        guitar_t test_guitar = (guitar_t)malloc(sizeof(struct guitar_s));

        guitar_setid(test_guitar, keynum);

        cmpfunc temp = (cmpfunc)guitar_compare;

        // Come back and deal with the callback here later
        match_guitar = (guitar_t)list_find(dbpoint->dblist, test_guitar, temp);

        free(test_guitar);

        return match_guitar;
}

// Delete a guitar from database, return pointer to it, do not free
guitar_t guitardb_delete(guitardb_t dbpoint, key_t keynum)
{
        guitar_t match_guitar = guitardb_lookup(dbpoint, keynum);

        if (match_guitar != NULL)
        {
                list_remove(dbpoint->dblist);
                dbpoint->dbsize--;
```

```c
        }

        return match_guitar;
}

// Reports all of the guitars in the database with all info for each guitar
void guitardb_report(guitardb_t dbpoint)
{
        if (dbpoint->dbsize != 0)
        {
                int i;
                guitar_t rover = list_first(dbpoint->dblist);

                for (i = 0; i < dbpoint->dbsize && rover != NULL; i++)
                {
                        guitar_print(rover);

                        rover = list_next(dbpoint->dblist);
                }
        }

        else
                printf("\nList is empty.\n\n");
}

/* Free all resources used by the guitar database.
   Remove everything from the linked list and free all pointers there
   before finalizing the list */
void guitardb_finalize(guitardb_t dbpoint)
{
        if (dbpoint->dbsize != 0)
        {
                int i;
                guitar_t todelete = (guitar_t)list_first(dbpoint->dblist);

                for (i = 0; i < dbpoint->dbsize && todelete != NULL; i++)
                {
                        free(todelete);
                        todelete = (guitar_t)list_next(dbpoint->dblist);
                }

                list_finalize(dbpoint->dblist);
        }

                free(dbpoint);
                dbpoint = NULL;
}
```