ECE 223 Programming Assignment 1
Due Date in Canvas

Simple Guitar Database Program

A guitar is represented with a C structure as follows:

```
struct guitardb_guitar {
   int  item_key;    /* unique id number for database */
   char make[15];    /* Fender, Gibson, Gretsch, etc. */
   char model [15];  /* Telecaster, Les Paul, etc. */
   char submodel[15];/* Deluxe, Standard, Custom, etc. */
   char body_type[2];/* SB solid body, HB hollow body, SH semihollow */
   int  year;        /* year of manufacture */
   char finish[25];/* description of finish including color, binding, etc. */
   int  strings;/* number of strings - usually 6 or 7 or 12 */
   int  pickups;/* number of pickups — usually 1 — 3 */
   char neck;  /* pickup type - H humbucker, C single coil, */
               /* P P-90, F Filtertron, N none */
   char middle;/* Same as above */
   char bridge;/* Same as above */
};
```

The database should have the following API:

```
struct guitardb *guitardb_create();
int guitardb_add(struct guitardb *, struct guitardb_guitar *);
struct guitardb_guitar *guitardb_lookup(struct guitardb *, int);
int guitardb_delete(struct guitardb *, int);
struct guitardb_guitar *guitardb_first(struct guitar_db *);
struct guitardb_guitar *guitardb_next(struct guitar_db *);
int guitardb_destroy(struct guitardb *);
```

A struct guitardb refers to the entire database. There can be more than one guitardb in the program at once. The int arguments refer to the item_key field, used as a unique locator for the item. All functions returning int should return 0 on success and -1 on error. All functions returning a pointer should return NULL on error.

The guitardb must be implemented as an array. Each guitar added to the guitartdb occupies one array slot. Your code must manage the slots. Each slot of the array should contain a pointer to one of the guitardb_guitar structs above.

When the Inventory is full guitardb_add() calls return an error. When it is empty guitardb_delete() returns an error. Attempting to add a guitar with the same item_key as one already in the guitardb fails with an error. Attempting to delete a guitar not present fails with an error.

The guitardb_first() function always returns the first guitar in the array or error if there are none. The guitardb must include a cursor which indicates the last guitar in the array returned by first or next. First should set this cursor to the first available item in array order. The guitardb_next() function returns the next guitar in the array after the last one returned by either first or a previous next, unless the last one returned was the last one in the array, in which case it returns an error. It assumes the cursor has been set either by a previous call to first or next. If that is not the

case, it should return error. Hint: have a “reset” value such as -1 to indicate the cursor is not set. The next function sets the cursor unless the last item in the array was returned, inc which case it resets the cursor. Calls to add or delete should also reset the cursor.

Write a program that tests the database of guitars. It should read in a loop from the command line one of five commands:

```
> ADD
> LOOK item-key
> DEL item-key
> LIST
> QUIT
```

The first word of each command must be written exactly as shown. The remaining words are strings and may be of any length. Commands with an unexpected first word, or with an incorrect number of arguments are ignored and a suitable 1-line error message is printed.

The **ADD** prompts for each field of the guitar data, one field on each line, and then prints “Data added”. The test program should create and keep up with item_keys, the user should not have to create them. The easiest way to do this is to have a static int in the test program that you increment each time a new item is created. The test program should also deal with mallocing and freeing the memory used to hold guitar structs.

The **LOOK** command prints the guitar data fields, one per line, or the message “Data not found” if the item-key is not present.

The **DEL** command prints “Data deleted” unless the item-key is not present, in which case is prints “Data not found”.

The **LIST** command should list all of the guitars in the database in the order they exist in the array. It should only print the item_key, make, model, and year.

The **QUIT** command should end the program. All commands must print suitable error messages if an error is returned from the guitardb functions.

This program must use an array of pointers to the C structures that contain the information about each guitar (struct guitardb_guitar *) as its storage medium. The code must consist of the following files:

guitardb.c - which contains the abstract data type code. The interface functions must be exactly as described above.

guitardb.h – the interface declarations for the ADT.

<userid>.assign1.c - which contains the main() function, menu code, and all other functions not part of the actual database.

All items specified in the ECE2230 Programming guide are included as part of this assignment. All code and documents must be turned in via canvas.