**Homework 1028: Contact DB**

Last modified: 24 October 2021

Due: 28 October, noon

**Description**

This assignment must be completed entirely in a Unix environment, either Linux or Mac OS.

Write a C program that serves as a simple contact database. When the program starts, the database is read in from disk file. While the program is running, the database resides in RAM and is maintained based on interactions with the user. Upon exiting, the program writes the current state of the database back to the same disk file.

The name of the file will be given as the only command-line argument. Every line of the file will contain the information on one contact. The lines will be in no particular order. The structure of a line is as follows:

name[TAB]phone number[TAB]email address

You may assume that the file contains only printable ASCII characters, tabs, and newlines. You may assume that no piece of information in the file contains a tab character; the tabs are only used as separators. Any other printable ASCII character is legal in any field. You should also assume that there will never be more than 1024 contacts in the file.

When the program starts, it reads all of the contact information in the file into an array of dynamically allocated structs (in other words, an array of pointers to structs). If no file is named or if there is an error opening it, the program should exit with an appropriate error code. Allocate exactly as many structs as are required for the information in the file. Then repeatedly prompt the user for an action.

There are four actions that the user can perform on the database:

1. List a set of entries from the database. The program will prompt the user for a search string. The program will display all entries for which the string matches (case insensitive) anywhere in the first or last name.
2. Add an entry to the database. The program will prompt the user for the various fields: names, phone number, email address. The new entry should be added to the end of the list of existing entries, and the entire list should be displayed on screen.
3. Delete an entry from the database. The program will prompt the user for an entry id to delete. If the entered id does not match an entry, that is not an error. If it does match, that entry is deleted, and all subsequent entries are moved up, so there are no holes in the list. At the conclusion of the actions, the entire list is displayed on screen.
4. Exit the program. This causes the current state of the database to overwrite the original data file, ready to be read again when the program is next run.

A run of the program should look exactly like this.

$ ./program contacts.txt

0: Harry Potter 423-749-6268 boy\_who\_lived@hp.com

1: Ginny Weasley 880-813-1994 gf@hp.com

2: Ron Weasley 756-816-0257 best\_bud@hp.com

3: Hermione Granger 243-961-7287 best\_girl@hp.com

4: Draco Malfoy 143-456-7890 meanie@hp.com

5: Luna Lovegood 566-517-4443 dreamy@hp.com

S: Search

A: Add

D: Delete

Q: Quit

Choice: s

Enter search string: er

0: Harry Potter 423-749-6268 boy\_who\_lived@hp.com

3: Hermione Granger 243-961-7287 best\_girl@hp.com

S: Search

A: Add

D: Delete

Q: Quit

Choice: d

Enter the id of the contact to delete: 4

0: Harry Potter 423-749-6268 boy\_who\_lived@hp.com

1: Ginny Weasley 880-813-1994 gf@hp.com

2: Ron Weasley 756-816-0257 best\_bud@hp.com

3: Hermione Granger 243-961-7287 best\_girl@hp.com

4: Luna Lovegood 566-517-4443 dreamy@hp.com

S: Search

A: Add

D: Delete

Q: Quit

Choice: q

Your program must use this exact set of definitions (you can define additional things as well):

#define MAX\_NAME\_CHARS 40

#define MAX\_PHONE\_CHARS 12

#define MAX\_EMAIL\_CHARS 30

typedef struct

{

char name[MAX\_NAME\_CHARS + 1];

char phone[MAX\_PHONE\_CHARS + 1];

char email[MAX\_EMAIL\_CHARS + 1];

} Contact;

The program must be separated into appropriate modules. Each module should handle a particular set of tasks in your program. A possible organization might be:

* The main module, which contains the main logic of the program; calls functions in other modules. Performs only minimal IO.
* A file operation module, which contains functions that handle reading the contents of the file into the array, and writing the array back to the file.
* A module that contains the functionality to add and delete entries in the database.
* A module that contains the functionality to display single entries, and the entire list of entries.

Each module except main should have an appropriate matching header file that gives relevant definitions and the declarations of the functions it contains. In addition, you will probably need a header file that contains the definitions and structs that are used by more than one of the modules.

You must write a Makefile that contains rules to build each module and the executable, along with rules for all, and clean. The final executable program must be named contacts.

You must create a gzipped tarball that creates a directory named yourlastname, and within this directory you must have all of the .h and .c files, and the Makefile, and nothing else.

The only input function used in your entire program should be fgets. The only output functions used in your entire program should be printf and fprintf.

Note that the array is statically declared at compile time, while the elements of the array (the structs) are dynamically allocated as the program runs.

As always, strictly follow all rules in the coding style guide. Make sure your name appears in every file.

Submit your .tgz tarball to the [homework submission](https://borax.truman.edu/250/submit.php) page.