Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science

6.087: Practical Programming in C

IAP 2010

Problem Set 7

Using and creating libraries. B-trees and priority queues.

Out: Friday, January 22, 2010. **Due:** Monday, January 25, 2010.

Problem 7.1

In this problem, we will be utilizing the SQLite serverless database library to load and query a database file. The required library `libsqlite3.so` and header file `sqlite3.h` appear to be installed on some Athena[1] workstations (but not on the Athena[1] server). If you cannot find it on your own machine's distribution, you can download the source code from `http://www.sqlite.org/download.html`. Generally, for your machine, you will have to compile the library yourself. For Linux, use the `.tar.gz` source code amalgamation package that contains the configure and makefile support. To compile for Linux, extract the files, run `./configure`, `make`, and `make install`. For other platforms, you will need to compile and install the library yourself.

(a) Download the source code `prob1.c` and movie database file `movies.db` from Stellar. The database contains a list of movies indexed by BarCode. The list also contains the MovieTitle, MovieCategory, ProductionYear, Format, Language, and Web fields for each movie. In this part, we will use the `sqlite3` library to load this database and perform simple SQL queries. (In case you're curious, the database lists information for movies available from the library in the Sidney Pacific dormitory.) The source code will get you started.

For this part, you need to fill in the missing code to open the database file, perform an SQL query, and close the database. In particular, you will be expected to use the `sqlite3_open()`, `sqlite3_exec()`, and `sqlite3_close()` functions in your code. Read the documentation available on the SQLite website for details.

Via Stellar, turn in your code for this part, as well as your program's output for the first three SQL queries provided in your code.

(b) In this part, you will read the entire table into a B-tree, sorted by the name of the movie (assumed to be unique). In the provided code, you will find mostly completed for the B-tree. Complete the missing lines in the B-tree functions. For this and future parts, use the final SQL query, which reads the entire `movies` table.

Once you have done this, add code to your `main()` function to print the results of an inorder traversal of the B-tree to a file (second command line argument to the program). Via Stellar, submit online your code and the file containing the result of the traversal.

[1] Athena is MIT's UNIX-based computing environment. OCW does not provide access to it.

(c) Now, modify your code to ask the user for a movie title. Use the `find_value()` function to locate the movie and print the movie information using `display_record()` to the console. Your program should continue to ask the user until the input string is an empty line. Be sure to remove the trailing newline `'\n'` character from the end of the string returned from `fgets()`.

Submit via Stellar your code for this part and your console output for a few illuminating test cases. For instance, `"Citizen Kane"`, `"Casablanca"` and `"Gone with the Wind"` are all in the database.

(d) For this final part, you will create your own library from the code you wrote for your program. In particular, you should provide the following functions:

- **int** initialize_db (**const char** ∗ filename); – loads the SQLite database from file *filename* and reads the contents into a B-tree, sorted by movie title. The function should return zero if successful, or non-zero otherwise. Don't forget to close the database when you're done loading from it.
- **int** locate_movie(**const char** ∗ title); – searches the B-tree for the movie with title *title* and writes the result to the console. The function should return non-zero if a record was found, or zero if no record matches the specified title.
- **void** dump_sorted_list(**const char** ∗ filename); – performs an inorder traversal of the B-tree, dumping the sorted list of movies to the file specified by *filename*. This function does not return anything.
- **void** cleanup(**void**); – this function should free the memory used by the B-tree and its records. This function does not return anything.

Be sure to remove your main function from the library, and go ahead and compile this library. On Stellar, you will find the program `prob1d.c` that will use the library. Compile the program, either statically or dynamically linking it against your library (and `sqlite3`).

Submit via Stellar a copy of your code and the output of running the program defined in `prob1d.c`.

6.087 Practical Programming in C

January (IAP) 2010