159.240 Advanced Exercises: SQLite

Please note that these tasks are not awarded any marks, and are purely intended for some interesting further development. Please do not hand in any work from this, and do make sure that your assignments are done before giving this a go.

In software development jobs, it is uncommon to be on a project that does not use an SQL-based database. SQL (structured query language) is a domain-specific language used to extract information, but also to maintain data held in a (mostly) centralised location.

SQLite is a small but powerful database system. It has bindings for C but also for various other languages. An introduction to the C/C++ interface for SQLite can be found here:

http://www.sqlite.org/cintro.html

Your task is to create a program using SQLite from a C/C++ program. Start from this code sample:

```
#include <stdio.h>
#include “sqlite3.h”

int main() {
    sqlite3 *db;
    int rc;

    rc = sqlite3_open("test.db", &db);

    if (rc){
        printf("Can't open database: %s\n", sqlite3_errmsg(db));
       return -1;
    }else{
        printf(“Opened database successfully\n");
    }
    sqlite3_close(db);
}
```

To run this program on Windows, you need to:
1. Save the code as main.cpp in its own folder
2. Download this file:
http://www.sqlite.org/2014/sqlite-amalgamation-3080702.zip
Which is the source code for SQLite. And extract the files into the folder with main.cpp
3. Hold shift, and right click in the folder with main.cpp in it.
4. Click on Open Command Window Here
5. In the command window, type in:
gcc -c sqlite3.c
g++ main.cpp sqlite3.o -o main.exe
6. Your program should have compiled without errors. Now just run main.exe
7. Check that you have a file called test.db in the folder afterwards. This is your database.

Every time you compile your main.cpp, you need to compile it using the g++ line above. You can't compile it using SciTE alone.

You have a database, but no tables in it. SQL operates by storing data in tables, where each column is used to store a particular thing, like a Date, or a name, or an email address. The columns have types, just like we have int, char and float in C.

```
#include <stdio.h>
#include <stdlib.h>
#include “sqlite3.h”

static int callback(void *NotUsed, int argc, char **argv, char **azColName){
   int i;
   for(i=0; i<argc; i++){
      printf("%s = %s\n", azColName[i], argv[i] ? argv[i] : "NULL");
   }
   printf("\n");
   return 0;
}

int main(int argc, char* argv[])
{
   sqlite3 *db;
   char *zErrMsg = 0;
   int  rc;
   char *sql;

   /* Open database */
   rc = sqlite3_open("test.db", &db);
   if( rc ){
      fprintf(stderr, "Can't open database: %s\n", sqlite3_errmsg(db));
      exit(0);
   }else{
      fprintf(stdout, "Opened database successfully\n");
   }

   /* Create SQL statement */
   sql = "CREATE TABLE Employee ("  \
         "ID INT PRIMARY KEY     NOT NULL," \
         "NAME           TEXT    NOT NULL," \
         "AGE            INT     NOT NULL," \
         "ADDRESS        CHAR(50)," \
         "SALARY         REAL );";

   /* Execute SQL statement */
   rc = sqlite3_exec(db, sql, callback, 0, &zErrMsg);
   if( rc != SQLITE_OK ){
   fprintf(stderr, "SQL error: %s\n", zErrMsg);
      sqlite3_free(zErrMsg);
   }else{
      fprintf(stdout, "Table created successfully\n");
   }
   sqlite3_close(db);
   return 0;
}
```

The code above will create a table called Employee. Each row in this table will be an employee with their name, age, address and salary. Notice that the data types aren't exactly the same as the ones in C. For example, we have TEXT, CHAR(50) and REAL. The CREATE TABLE string is actually a valid SQL statement.

Implement this code, and make sure it works. Remember that once your code finishes, the SQLite database is saved, so you might run it again, and try to create the table again, and it might give you an error message, saying that the table already exists. This is the great thing about databases, things are persistent.

Now that you have done a SQL statement (CREATE TABLE), have a look online at SQLite's INSERT statement, and insert some data into the table.

Something like this:

insert into employee (name,age,address,salary) values ('James', 25, '123 Fake St', 90000.0)

The next thing to do is to read this data from the database. This is a lot more complicated, but is pretty important. Use this code, and read through it to see how it works:

```
#include <stdio.h>
#include <stdlib.h>
#include <sqlite3.h>

static int callback(void *data, int argc, char **argv, char **azColName){
   int i;
   fprintf(stderr, "%s: ", (const char*)data);
   for(i=0; i<argc; i++){
      printf("%s = %s\n", azColName[i], argv[i] ? argv[i] : "NULL");
   }
   printf("\n");
   return 0;
}

int main(int argc, char* argv[])
{
   sqlite3 *db;
   char *zErrMsg = 0;
   int rc;
   char *sql;
   const char* data = "Callback function called";

   /* Open database */
   rc = sqlite3_open("test.db", &db);
   if (rc){
      printf("Can't open database: %s\n", sqlite3_errmsg(db));
      exit(0);
   }else{
      printf("Opened database successfully\n");
   }

   /* Create SQL statement */
   sql = "SELECT * from Employee";

   /* Execute SQL statement */
   rc = sqlite3_exec(db, sql, callback, (void*)data, &zErrMsg);
   if( rc != SQLITE_OK ){
      fprintf(stderr, "SQL error: %s\n", zErrMsg);
      sqlite3_free(zErrMsg);
   }else{
      fprintf(stdout, "Operation done successfully\n");
   }
   sqlite3_close(db);
   return 0;
}
```