In this lab project you are going to leverage a KV store (specifically LevelDB) to build a scaled-down database system, named as TinyTable. Different from implementation of a conventional DB system, TinyDB stores and accesses its data in a table as key-value pairs, each corresponding to a cell in the table. For a specific cell, the key is a combination of the cell's row name and column name (both are character strings), and the value is character string, or the data, stored in the cell. As a DB system, TinyTable allows a user to type below commands, one at a time, to define a table, write (set) individual rows, read (get) one or multiple rows, access individual cells, and exit.

1. TinyTable’s Commands

1: table column-name-1 [column-name-2 …. column-name-n]

Note: This command defines the column names, each a character string, of the table currently maintained by TinyDB. By default, TinyTable maintains only one table. This command must be executed before a table can receive its data for storage. This is especially the case for commands associated with entire rows (without explicitly specifying specific column names) such as ‘setr’, ‘getr’, and ‘delr’.

2: setr row-name column-1-value [column-2-value …. column-n-value]

Note: This command writes (sets) values of a specific row (‘row-name’) into the table. The list of column values must match, in terms of the column count and order, the list of column names specified in the 'table' command.

3: getr row-name-1 [row-name-2 ... row-name-k]
Note: This command reads (gets) values of each of the listed rows (‘row-name-i’) and display the values line by line.

4: delr row-name-1 [row-name-2 ... row-name-k]

Note: This command removes (deletes) values of each of the listed rows (‘row-name-i’).

5: sete row-name  column-name column-value

Note: This command writes (sets) value (‘column-value’) at a specific row (‘row-name’) and a specific column (‘column-name’).

6: gete row-name  column-name

Note: This command reads (gets) value (‘column-value’) at a specific row (‘row-name’) and a specific column (‘column-name’), and display it.

7: dele row-name  column-name

Note: This command writes (deletes) value (‘column-value’) at a specific row (‘row-name’) and a specific column (‘column-name’).

8: exit

Note: This command closes the program.

2. Setup of the experiment platform
You will continue to use the Cygwin environment you have set up for Lab1:

(1) `mkdir C:/cygwin64/home/<YOURNAME>/Lab2`

(2) Uncompress the provided 'leveldb-7650-lab2.zip' in Lab2’s directory:

(3) Build LevelDB in the ./Lab2 directory

   `$ make -j4 tinytable`

(4) Run tinytable. You will see welcome messages and a list of supported commands.

   `$ ./tinytable`

3. Your assignments

(1) Carefully read 'tinytable.cc', which is the source code for implementing the set of TinyTable’s commands, and a main() function for providing a console accepting and interpreting commands. ‘sample.txt’ contains a list of sample commands for you to familiarize yourself with the system. To batch-run all commands in the file, you can redirect the program’s input to the file: “./tinytable < sample.txt”.

In ‘tinytable.cc’, functions prefixed with "cmd_ " are for implementing corresponding TinyTable commands. Functions prefixed with "lldb_ " are wrapper functions for corresponding LevelDB operations (GET/PUT/DELETE).

(2) There are two functions (cmd_delr() and cmd_sete()), that are used to implement ‘delr’ and ‘sete’ commands, are not completed. Please refer to the implementations of similar functions to complete them.
(3) Currently each time you start the program you need to run the 'table' command to supply the list of column names, which are stored in the 'vector<string> columns' in the 'struct TinyTable' but are not saved on the storage.

```cpp
struct TinyTable {
    leveldb::DB * lldb; // low-level LevelDB
    vector<string> columns; // column names
};
```

To run 'tinytable' without having to use the 'table' command again (suppose you don't need to modify the list of columns), please implement two functions called in the `main()` function before the program exit (save_table_scheme()) and at the beginning of the program (load_table_scheme()). Specifically, `save_table_scheme()` stores the data in the 'vector<string> columns' into the LevelDB store, and `load_table_scheme()` loads the saved column names from the store into 'vector<string> columns' (if they have been saved).

All your codes should be included in “tinytable.cc” and you should not modify other files. Submit this file (“tinytable.cc”) via email before the due time.