Collaborators: Sarmishtha PV and Melissa Wen
NETIDs: sprathi2 and mwen2
CSC 173: Project #4 README
April 11th, 2018

A. BUILD- To compile the project, enter the following command in Terminal or Docker:

   gcc -std=c99 -Wall -Werror -o run run.c part1.c part2.c part3.c HashMap.c part1num4.c genericTuple.c

   To run the project, enter the following command in Terminal or Docker:
   ./run

B. DEMONSTRATION

Part 1 Implementation

A. We followed the book's structure in 8.2 and 8.4 for the storage and implementation of the relations. Based off the book, we created separate structs for each of the relations present. Hence, there are 5 structs present in total: CSG, SNAP, CR, CDH, CP. The outline for the struct can be seen in part1.h file and more information regarding the methods are written in part1.c. The relations were stored using a Hashmap- which can be seen in HashMap.c. The Hashmap contains a schema, size (of the array used for the Hashmap), number of tuples, and a bucket. Our bucket will hold a tuple which has a pointer to another tuple. This way, if functions like a linked list. The overall database can be seen in the run.c file- where we create a new Hashmap for every relation present and store it.
B. The insert, delete, and lookup operations are shown in the part1.c file. For insertion, the key is put through a hash function, and then the tuple is inserted into that index of the hashmap array. If there is already something there, it will be put at the end of the linked list in the bucket. Our database does not have secondary indexing, so the delete function will iterate through the whole hashmap if the specified arguments are not the key. In lookup, we ended up making it so that the whole hashmap will be iterated through anyways. When the project runs, it asks the user for a relation to use to test the delete and lookup functions, and then prompts the users for input to perform those functions. For convenience, the tables in the book (CSG, SNAP, CR, CDH, CP) have already been inserted. The insertion functions are also in part1.c.
C. This implementation is shown in the run.c file (as specified above). To populate the table, navigate to run.h and test using the insert functions specified above. The insert function will take in the tuple. For instance, CSG will take in a char course, char studentID, and char grade.
D. The files are written/read in our project folder titled "FILES." To test this implementation, open the FILES folder and change any of the files that begin with 'read.' Example code is provided in all the files. The code is written to an external file that begins with 'write' in the same folder. If the file is not present, it creates a new file. You may want to clear the write files in the folder to ensure that the write functions work.

Part 2 Implementation

A. This implementation is when the run.c file is compiled and run. During part 2, it will prompt the user for a student name and course name and provide the output according to the tables implemented in Part 1. The current tables are set-up according to the values in the book. Make sure the course name is 5 characters. Because we do not have secondary indexing, we used a nested for loop – one to iterate the tuple in the SNAP relation, and one for the CSG relation.
    a. Example Input:
        i. Student Name: "C. Brown"
        ii. Course Name: "CS101"
B. This implementation is when the run.c file is compiled and run. During part 2, it will prompt the user for a student name, time, and day and and provide the output according to the tables implemented in Part 1. The current tables are set-up according to the values in the book. Make sure there are no spaces in time. Because we don't have secondary indexing, we have many nested loops to iterate through the tuples of the different relations.
    a. Example Input:
        i. Student Name: "C. Brown"
        ii. Time: "9AM"
        iii. Day: "M"

Part 3 Implementation

A. Part 3 contains the selection, projection and join operations. The selection operation is based on example 8.12 and extracts all the classes from a Hashmap. In the run.c file, the current class used is "CS101." To test, simply change this in the run.c file
B. The projection operation tests to see only the student IDs for a particularly class. The current class used is "CS101." To test, simply change this in run.c file
C. The join operator joins the relations CR and CDH. The current print is the tables CR and CDH. To test, simply change the tables in the run.c file and examine the output.

Extra Credit

We implemented the first extra credit option for 20 points. We created a struct for genericTuple and generic implementations for insert, delete, and lookup. We used an altered figure 8.1 to demonstrate the implementation in the run.c file. To test, simply change the data values to your choosing (in accordance with the proper parameters that is specified in genericTuple.H)

As we had trouble getting user input from the command line that included spaces, we used a function getlin3() to help with that. The code for this function was referenced from https://stackoverflow.com/questions/314401/how-to-read-a-line-from-the-console-in-c