//CSC 173 Project 4 README
//Adam Kravtiz and Ari Geller

This project is a relational database that can create 5 different relations, including a CSG (Course-StudentID-Grade) relation, a SNAP (StudentID-Name-Address-Phone) relation, a CDH (Course-Day-Hour) relation, a CP (Course-Prerequisite) relation, and a CR (Course-Room) relation. Our submission already has the executable main included in it. In order to run the program, use the terminal to go to the directory in which our project folder is in and type "./main". We made this executable with a Makefile, which is also included as Makefile.make. Although it is not necessary, since the executable is included, the Makefile is used by entering make -f Makefile.make in the directory where the project files are.

We used the Textbook as a guide of how to make 1 of the relation structs, then we used that example as a guideline to the other 4 the relation structs. The other ideas we saw in the text book was a picture of how to find what room a student is in given a students name, and also given the day and the hour. The assignment asked us to do these parts of the project as the textbook said.

Problems and Solutions:

A big issue we had was that, as we understood the project, we had to account for the fact that there could be multiple tables of the same kind (e.g. 2 CSG tables, 4 SNAP tables, etc). In order to implement this, we created a struct that was like a linked list of tables.

A problem we faced was how to do the relational algebra part of the project since relational algebra dynamically changes the size of the relation we have. We were told that we should make structs for tables based on the textbook, but those structs were set up in a way seemingly incompatible with relational algebra, specifically join functions. So, we created another struct that represented a “General Relation,” that had enough room to store any item in our database (e.g. course, room, name, address). However, in any given instance of this struct, some of these fields would be NULL. We also created methods to convert from our other structs to this General Relation so as to perform relational algebra operations on them.

Another problem we had was reading back from a file. The first problem we had was identifying if spaces were separating the values of the different tuples we were looking at or if it was part of the value of a tuple. For example we had problems identifying if C. Brown was 1 value for Name or if it was part of 2 separate values since there was a space in-between C. and Brown. We solved that problem by writing an “@” symbol instead of spaces for single values that contain spaces, and spaces were used to separate different tuple values. When reading from a file, we simply replaced “@” signs with spaces. The other problem we had from reading from the file was that we had trouble reading our indicators for when we are at the end of a relation, or at the end of the file. Because we had to account for multiple table of the same kind, we needed a way to differentiate not just between types of tables, but different tables of the same kind. We did this by writing the word “NEXT” after a table was finished. However, since different tables had different amounts of columns, we had to tailor the number of “NEXT”’s written to the type of table. For example, the CSG portion of the text file would have three “NEXT”’s after each CSG table, while the SNAP part would have four.

This project was done as a two-person group consisting of Adam Kravitz (NetID akravit2) and Ari Geller (NetID ageller2).

The LinkedList code we use in the project was taken from code Professor Ferguson gave us in Project 1.