//Readme.txt

//CSC 173 Project 2

//Adam Kravtiz and Ari Geller

This project is a relational database that can create 5 different relations, which consists of 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 lastly a CR (Course-Room) relation. Our submisison 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,

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.

Problems and Solutions:

A problem we faced was how to do the relation algebra part of the project since relation algebra dynamic changes the size of the relation we have. We tried ideas that tried to make the relations using linked list instead of an array (the way the textbook made the relations), however we did not was all the code we already wrote so instead we made a method that would convert our relations into a linked list that will be temporarily made just so we can do the relation algebra (selection, projection, join).

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. So the way we solved that problem was to write a “@” symbol instead of spaces for single values that contain spaces, and spaces were used to separate different tuple values. 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. The problem was the way we were reading, the way we read for example for a 3-tuple relation is by reading 3 strings separated by spaces at 1 time then reading the next line. But the problem was that our indicators were only 1 string long so the function we used to read kept reading the next 2 strings that were written down and messed up and we unable to identify where our indicators were, so we ran into a lot of problems. The way we fixed that problem was by varying the amount of time we write our indicators, for example if we are looking at 3-tuple relation which read 3 strings per line at once, then we write our indicators 3 times after the relation, this allowed us to identify if we are at the end of a relation or at the end of a file.

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

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