CS 210 Spring 2012
Module 6
Relational Operations

There are 6 relational operations that are commonly used in databases to manipulate sets of data. You will implement these 6 operations on ***tables***, and print out the results to the command line for module 6.

The 6 operations are:

1. BASIC RELATIONAL OPERATIONS
    1. *select A [where ...];* -- you've already implemented this for module 4, so this should be quite easy!
    2. *project A over X, Y, Z, ...;* -- you will access table A and get and print out the values from every row for those fields named in the 'over' clause (ie X, Y, Z)
    3. *join A and B;* -- access the first row from table A. Then sequentially access each row from table B and add on the values from the row to the end of the values from the first row from A. Then access the second row from A, sequentially access the rows from B and and them to the end of the 2nd row of table A. Repeat for each row in A. The result will be a "table" which will have a number of fields equal to the field size of A PLUS the field size of B, and the number of rows from A TIMES the number of rows of B. The names of the fields should be modified with the names of the tables. For example if table A has fields 1 and 2, and B has fields 2, 3, and 4, the result will have fields A.1, A.2, B.2, B.3, and B.4
2. **SET OPERATIONS** The three set operations can be thought of as operations on Venn diagrams. Note that in order to perform these operations on two tables, the tables must be ***Union Compatible***. This means that both tables must have the same number of fields and each field must be the same datatype as that in the other table. The names of the fields do NOT need to be the same. Give each field in the result the name of the field in the first table listed, so that if you perform one of these operations on A (with fields 1, 2, 3) and B (with fields 4, 5,6) the result will have fields with names 1, 2, 3. Note also the definition of ***row equality*** – rows are considered equal iff every value in the first table's row is equal to its sequentially match value in the second table's row.
    1. *union A and B* – combine all rows of table A, followed by all rows of table B that are NOT in A (if there is a row in A that is equal to a row in B, do not add that row from B)
    2. *intersect A and B* – the result should include only rows from table A that have an equal row in table B
    3. *minus A and B* – the result should only include rows from table A which do NOT have an equal row in table B.

Hints: Make classes for Rows (including collections of Values) as well as for Datasets (which will include collections of Rows). Overriding toString and implementing Comparable on the Rows will make your life very easy indeed.