CMSC 461
Homework 1 - The Relational Algebra and RDB
Section 101 - Out September 21, Due September 30
Section 201 - Out September 20, Due September 29

This homework involves using an implementation of the relational algebra (RDB) on the UMBC8. A manual and tutorial for RDB have been provided as a separate handout. In submitting answers, provide a copy of interactive sessions in which the various values and functions are defined, and in which results are displayed. To do this, you should use the **script** command, which saves an interactive session in a file that you list after the command (do *"man script"* for details).

Please submit readable output; we do not want to see your mistakes, only final answers. It will help to do a trial run, or to edit the script file once you are done. You may also use a pen to beautify your output and to add a few appropriate comments– in particular, to indicate in a conspicuous color where each question or part question begins and to explain the use and meaning of identifiers where appropriate.

The first set of exercises uses RDB with a database of suppliers and parts. To access this, copy the supplier-parts database into your own area by saying

```
cp ~finin/sp.rdb .
```

You can then say *"use 'sp.rdb';"* while in RDB to load it. If you then save updated relations during an RDB session, you will not overwrite the original version. Note – you can also find this file on the CS Gopher server in the area for this class.

The Suppliers and Parts DB

The first four queries are simple, and require access to a single relation. The rest require access to multiple relations (remember your binary operators). You may break an answer up into a series of queries, using "it" to refer to the previous result. You may also store an intermediate answer in variables (*e.g.* "let temp=....."). Use the supplier-parts database to construct the following relations:

1. The CITYs of Suppliers (S).
2. The PNAMEs of Parts (P).
3. The PNAMEs of Parts that are Red.
4. The SNUMs and SNAMEs of Suppliers located in "Paris".
5. The SNUMs and PNUMs of Suppliers and Parts that are in the same CITY.
6. The SNAMEs of Suppliers that supply at least one Part.
7. The STATUSs of Suppliers that supply some Part in QTY greater than 250.
8. The PNUMs of Parts that are supplied by more than one Supplier.
9. The PNUMs and WEIGHTs of Parts that can be obtained from Suppliers in London.
10. The SNUMs of Suppliers who supply all parts. (Hint: recall "/".)

11. All pairs of SNUMs such that the two Suppliers are located in the same CITY. Avoid listing suppliers as being located in the same city as themselves, and listing duplicate tuples (*e.g.* (Blake,Jones) and (Jones,Blake)). (Hint: RDB does not like to compare fields with the same name. Use the "rename" command to avoid this problem.)
12. The total quantity of part "P2" supplied by suppliers.
13. The average quantity of part "P2" supplied by suppliers.

The DBB DB

Construct a database over the domains (i.e., attribute types) **drinkers, beers,** and **bars** that contains the relations **likes(Drinker, Beer)**, **serves(Bar, Beer)**, and **frequents(Drinker, Bar)**. To construct a relation, you'll need to assign the rows to the relation name like this:

```
rdb> let LIKES = {drinker beer
...>             [john  guiness]
...>             [susan amstel ]
...>             [wayne becks  ]}
value LIKES
```

Just make up any values you want for three relations; make them large enough that the queries below are likely to yield non-empty relations. Now find the following information by writing the appropriate RDB expressions:

1. The name of each drinker who frequents a bar which serves all of beers he likes.
2. The name of each drinker who frequents a bar which serves at least one of the beers he likes.
3. The name of each drinker who frequents a bar which serves none of the beers he likes.
4. The name of each drinker who frequents at least one bar; but none of the bars he frequents serves a beer that he likes.

The Family DB

Create a relation *family(Parent,Child)*, indicating that *Parent* is the parent of *Child* in some family. Insert the following tuples: (Anne,Boris), (Boris,Charles), (Boris,Emily), (Charles,Darlene), (Darlene,Fred). Using RDB, construct relations for:

1. grandparent(Gparent,Gchild)
2. greatgrand(Ggparent,Ggchild)
3. Generalize this to find the relation *ancestor(Anc,Person)*, where *Anc* is an ancestor of *Person.*