CS344: Programming Languages
Homework 10: Prolog

For both of the following problems, you are required to submit a sample prolog session that demonstrates each predicate working correctly, with comments (hand written or typed) added to describe what is being tested and what you are doing.

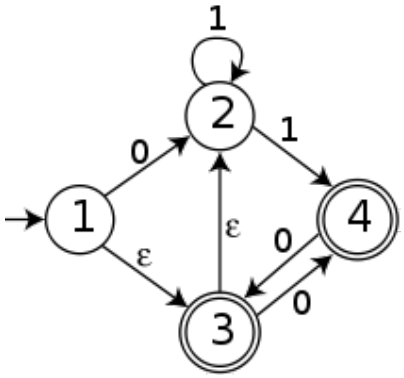
As a suggestion, you can use the Unix command script to store this easily as follows:

```
script sfilename                  //  a copy of your session will be saved
bash$ swill                       //  script may change the unix prompt
?-                                // Do your PROLOG thing.
?- ...
?- ^D                             // exit PROLOG
bash$ ^D                          // control-D exits from script
Script done. File is sfilename.
$lpr sfilename                    // print your script and annotate it
                                  // (or use pico and add comments that way)
```

Required Problems

1. As mentioned in class, Prolog is ideal for simulating NFA's as well as DFA's, since the language will backtrack and find valid parsings if some parsing fails.

   Code the specification for the following NFA in prolog as NFA.pl, and test your code by showing the derivations for several strings both in the language and not in the language. Show at least one string which has several possible parsings (although only one may end up in an accept state).



2. Write a PROLOG program that implements a family database for your family. Save it as an ordinary text file named family.pl.

   Your program should implement the following facts for your immediate family, grandparents, and great-grandparents.

```
parent_of(X,Y).
male(X).
female(Y).
```

That is, your database should consist of a number of facts about who is the parent_of of whom, about which individuals are male and about which individuals are female. For example:

```
parent_of(joe,susie).
parent_of(joe,dan).
parent_of(mary,susie).
parent_of(mary,dan).
male(dan).
male(joe).
female(susie).
female(mary).
```

All other predicates should be implemented as rules – i.e., as predicates involving variables and logical implication (:-). For example, the rules for father_of and daughter_of would be:

```
father_of(X,Y) :- parent_of(X,Y),male(X).
daughter_of(X,Y) :- parent_of(Y,X),female(X).
```

Define the following predicates for your database. Your database should be rich enough to test all of these predicates. For example, if you are only child, you may have to make up fictitious siblings to test the sibling_of predicate.

```
predicate           interpretation
---------           --------------
father_of(X,Y)      X is the father of Y
mother_of(X,Y)      X is the mother of Y
son_of(X,Y)         X is the son of Y
daughter_of(X,Y)     X is the daughter of Y
sibling_of(X,Y)     X is the sibling of Y
brother_of(X,Y)     X is the brother of Y
sister_of(X,Y)      X is the sister of Y
grandparent_of(X,Y) X is the grandparent of Y
ancestor_of(X,Y)    X is the ancestor of Y
```

Some of these predicates may be defined in terms of other predicates. For example, a sister is a female sibling and a grandparent is the parent of a parent. The ancestor_of predicate can be defined recursively to handle ancestors from any generation.