CS3200: Programming Languages
Homework 10: Prolog

For 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. We're going to return to the idea of storing a set in a list structure, but in Prolog instead of Haskell. Consider a set to be a list of integers (so that we have a comparison operator) where elements are stored in the list in sorted order, and no repetition is allowed. Given this representation, define the following predicates in Prolog:

   (a) member(X,L), which holds iff the element X occurs in L.

   (b) subset(L,K), which holds iff L is a subset of K.

   (c) disjoint(L,K), which holds iff L and K are disjoint (i.e. they have no elements in common).

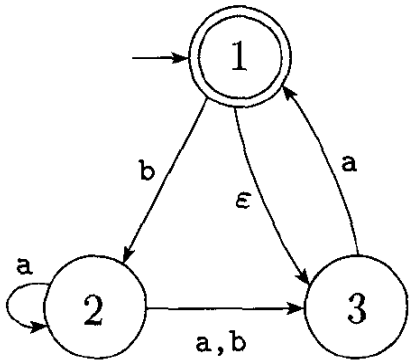
   (d) union(L,K,M), which holds iff M is the union of L and K.

   (e) intersection(L,K,M), which holds iff M is the intersection of L and K.

   (f) difference(L,K,M), which holds iff M is the difference of L and K.

2. As mentioned in class, Prolog is ideal for simulating NFA's as well as DFA's, since the language will backtrack and explore alternate possible 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).



3. Extra Credit: Use prolog to write a set of rules to determine whether a number is prime. A user should be able to enter the goal: prime(10) (for example) and get a yes or no.

   Hints: The base cases are that 1 and 2 are prime. (Technically, 1 isnt prime, but it might help you to assume it is depending on how you decide to write this.) Youll want to have a helper rule that tells you whether a number evenly divides another and handles the recursion - you'll probably use the "is" predicate to check if numbers are equal, and might also want to use the "mod" command to check remainders.