SE 3306: Assignment 1

Note: You are permitted to discuss the problems with your fellow students to understand the problems better, however, the solution and the code should be your own. Problems 2 through 5 have to be programmed in Prolog.

Question 1: Solve problems #5, #10, #15, #23, # 35 from the problem set on pages 78-80 in the textbook in the 7th edition.

Question 2: Consider the following family tree:

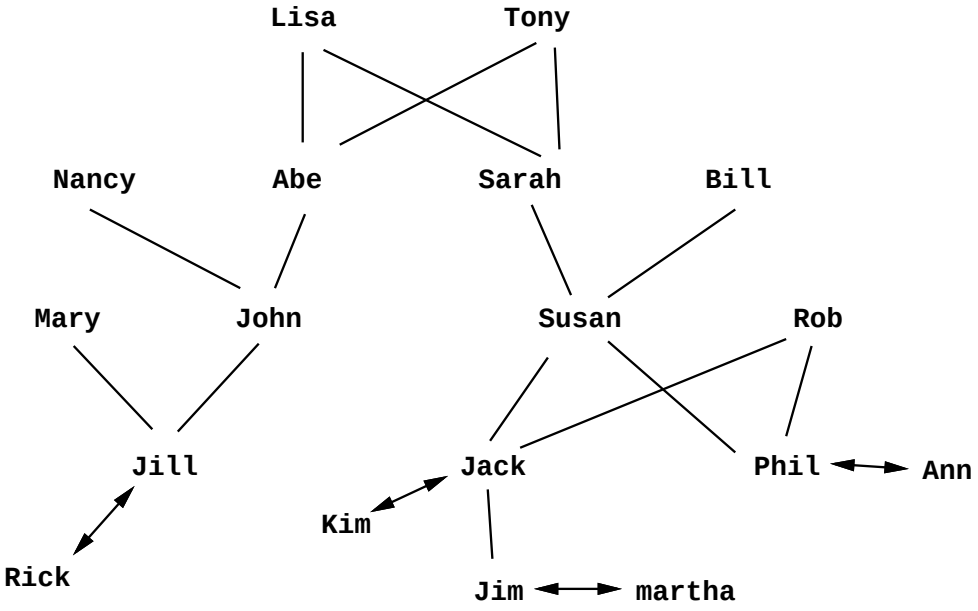

Figure 1: Family Tree

1. Transcribe the above diagram into father and mother relatinships as Prolog facts. Also add facts describing gender.
2. Define rules for the following relationships:
   (a) fcousin(X,Y): X and Y are first cousins
   (b) scousin(X,Y): X and Y are second cousins
   (c) aunt(X,Y): X is an aunt of Y.
   (d) grtaunt(X,Y): X is a great aunt of Y.
3. Define a rule for checking if X and Y are "cousins of the same generation," i.e., X and Y are descendents of a common person and both are same no. of links down from the common ancestor.

4. Suppose the double arrows depict the relationship "married." Two individuals are also married if they have a common offspring. Write a rule for finding out if X is the spouse of the nephew of Y.

5. Draw (part of) the search tree that Prolog will create for the query `aunt(sarah,john)` using your definition in Q 3.

Question 3: Write a predicate sumlists(List1, List2, List3) that takes two lists of integers, List1 and List2 and computes their sum in List3. For example, sumlists([1,2,3],[3,4,5],X) should return X = [4,6,8]. Try to handle the case where the two lists are of different lengths (the output list would be the length of the longest input list, and the elements in it after the shorter list ran out would just be equal to the elements in the longer list).

Question 4: Program the Tower of Hanoi puzzle.

Question 5: Suppose we represent sets (no duplicated elements) ranging over integers as lists of integers. Write Prolog predicates for performing the following set operations:

1. union(S1,S2): returns the set-union of sets S1 and S2.

2. intersection(S1,S2): returns set-intersection of sets S1 and S2.

3. setdiff(S1,S2): returns the set-difference of sets S1 and S2

4. subset(S1,S2): returns true if S1 is a subset of set S2, fails otherwise.

5. powerset(S1,S2): returns true if S2 is the powerset of set S2, fails otherwise.