Comp 333 Project #4 (32 pts) Spring 2016

Due: May 3

**GENERAL DIRECTIONS:** In this project you will write, compile and test four Prolog programs. Use SWI-Prolog. You may work in teams of 2 or 3 or you may do an individual project. Each team should submit only one project. If you form a team, complete an index card distributed by your instructor that lists the team members. All team member names must be embedded in the source code as a comment. Your source code file must be called project4.pl. **Instructor Test Cases will be posted on April 26.**

**Problem 1.** (8 pts) (Note: each individual on team will test Problem 1 with their own family tree)

* Create a Prolog database of your family tree back to your great grandparents (Based only on *parent, male, female* facts. Do not add marriage facts). Add parent, male and female facts as needed if your family is small or unknown. Similar to our Ancestor DB but for your family. Draw a graph of the family tree.
* Add relationship rules for ancestor, grandparent, grandfather, grandmother, mother, father, aunt (female sibling of a parent), uncle (male sibling of a parent), firstCousin (child of an aunt or uncle), sister, brother and descendant.
* Add a rule for relative. X and Y are relatives if they have a common ancestor or a common descendant
* Test your Prolog program. All rules should be tested at least once.

**Problem 2.** (8 pts)

* Write a Prolog program to parse the expression grammar

E 🡪 E + T | E – T | T

T 🡪 T \* F | T / F | F

F 🡪 ( E ) | <lowercase letter> | <number>

* Create an e(X), t(X) and f(X) predicates. Represent input X as a list of symbols. For example the expression 34\*(45 + b ) is represented by [ 34 , \*, ‘(‘, 45, +, b , ‘)’] and

e([ 34 , \*, ‘(‘, 45, +, b , ‘)’] ) returns true. The quotes around the parentheses are needed to distinguish input parentheses from Prolog parentheses. Nothing else needs parentheses.

* Do not use any built-in Prolog grammar parsers. Model your solution on the examples done in class.
* Test your program thoroughly using good and bad input.

**Problem 3. ( 8 pts)**

* SubseqSum Problem. Given a list of integers L and an integer M, find subsequences of L whose sum is exactly M. Usage: *subseqSum(L, M, X)* is satisfied if X is a subsequence of L and the sum of the elements of X is M. For example,

>?subseqSum([ 2,4,1,7,8, 4], 8, X) sets X = [4,4]; X = [1,7]; X = [8]

* Write a predicate *subseqSum(L, M, X)*. Solve program using a generate and test technique.
* Use the following Prolog predicate to generate all subsequences of a list

subseq([], []).

subseq( [H|T], [H|R]) :- subseq(T,R).

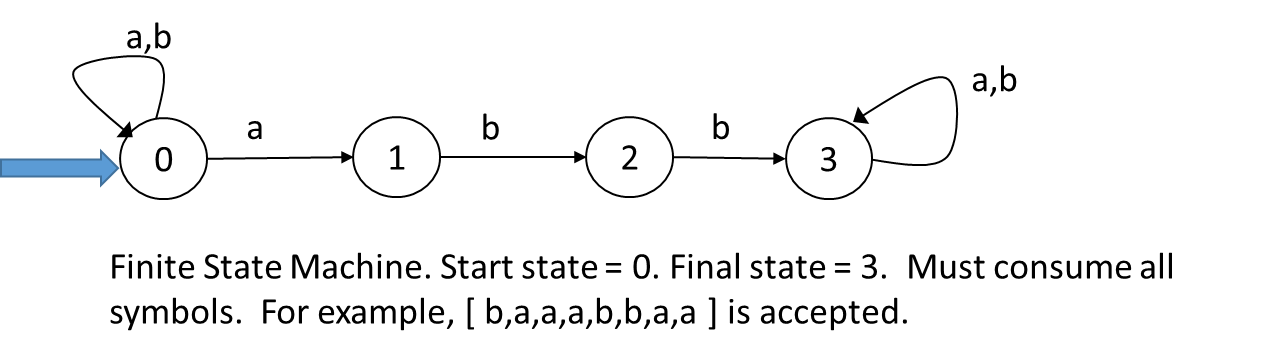
subseq( [ H|T], R) :- subseq(T,R).

Usage: subseq(L,X). will generate all subsequences of L.

* You may need a helper predicate sum(L,T) which finds the sum of a list L of numbers.
* Test your program thoroughly.

**Problem 4 ( 8 pts)**

* Create a prolog program that simulates the following finite state machine.



* To do this create a Prolog predicate fsm(state, list). Here state can be 0,1,2,3 and list is a list of symbols. fsm(k, list) is true if starting at state k, the list can be consumed by the finite state machine and end at state 3. Here is an example of one of the rules you will need:

fsm(0, [ a | T] ) :- fsm(0, T).

* Usage: fsm(0, [b,a,a,a,b,b,a,a,]) is true. (succeeds)
* Usage: fsm(0, [a,b,a,b,a,a,b] ) is false. (fails)

See Next Page for what needs to be submitted.

**Individual Hard Copy Turn in:** **(Due May 3 in class) [Separate from the Team Submission]**

Each individual on the team should submit a 3-page document w**hich** includes: Copy of code for Problem 1 with their own family data base (1 page); testing for Problem1 (1 page) with their family database; graph of their own family tree (1 page). Team grade on team submission depends on all team member submitting this part of the project.

**Team Submission Hard Copy Turn in” (Due May 3 in class)**

* 1. Title Page with team member names, date and project #
  2. Source code file that contains all code for Project#4. Use family data base given in Instructor’s Test Case. Call this file project4.pl
  3. Instructor Test Case Results for Problems 1, 2, 3, 4 ( Label problems and test case results clearly)
  4. Clip or staple all pages together in above order.

**Electronic Copy Turn in: (Due 7:30 am on May 3)**

Upload a single source file called project4.pl with all of your Prolog programs to Moodle. Your source file should contain as a comment all team member names, date and Project #. Submit only one source file per team.