PROBLEM 6 REVISED ON APRIL 21

Comp 333 Homework #7 Spring 2020

Due: Monday April 27 12 Noon ( upload to Canvas)

Points: 25

*Directions: Put all your answers in a SINGLE Word file ( not in a pdf file) called Homework7.docx . Put your answers in order. Skip at least 4 lines between the answers. In any problem that asks you to write a program, paste in the source code and the test cases from SWI Prolog. Test all examples given. Add your own test cases if no examples are given. Put your name, course , homework # and due date at the top of your file. Upload your file to Canvas by 12 noon on Monday April 27.*

**Problem 1**

Write a Prolog predicate min3( X,Y,Z, R) to find the minimum of 3 numbers. Here X,Y, Z are the input numbers and R is the minimum of X,Y,Z.

% Problem 1 - Min3

min3(X,Y,Z,R) :- X =< Y, X =< Z, R = X.

min3(X,Y,Z,R) :- Y =< X, Y =< Z, R = Y.

min3(X,Y,Z,R) :- Z =< Y, Z =< X, R = Z.

**Problem 2**

Write a recursive Prolog predicate maxList(L, R) to find the maximum in a list of numbers. Here L is the list and R is the maximum value in the list. It should work like this

% Problem 2 - maxList(L,R)

maxList([H|T], R) :- maxList(T,R), H > R, H = R.

maxList([H|T], R) :- maxList(T,R), H =< R, T = R.

maxList([H], R) :- R = H.

**Problem 3**

Suppose your Prolog facts and rules look like this:

e(a,b).

e(b,c).

e(a,c).

e(c,d).

e(a,f).

p(X,X).

p(X,Y) :- e(X,Z), p(Z,Y).

Describe in English sentences (in great detail) how Prolog solves the following queries and what the answers to the queries are. [Do not just run the query and/or print the Prolog trace. You need to give a verbal description that explains what is happening and that shows that you understand how Prolog solves queries.]

**?- p(a,d).**

First Prolog looks at the rule for p, which is P(X,Y) :- e(X, Z), p(Z,Y). So the first up to evaluate is e(X,Z). In this case, ‘a’ becomes the value for X, and Z remains a placeholder to evaluate all facts for e().

Starting from the top, the first fact for an e(a, Z) evaluation is e(a,b), now Z becomes ‘b’ and the second part of the p(X,Y) rule is evaluated. It fails but returns with a now becoming ‘b’, since the fact e(a,b) gives this relationship. This conintunes on through e(b,c), with ‘b’ becoming ‘c’, and ‘c’ becoming ‘d’ through e(c,d).

Now the second part of p(X,Y), can be evaluated, and p(d, Y) becomes p(X, X), becoming p(d,d), which is true, so it returns true.

**?- p( c,f).**

Very similar to how prolog ‘walks’ the facts on e(), except now we start at ‘c’ instead of ‘a’. This ends up evaluating false because e(c,d), stops with no relationship defining e(d,f), and since the rule p(X,Y) requires p(X,X), the ‘d’ can never become an ‘f’, so it can never pass in p(f,f).

**Problem 4**

Write a recursive Prolog predicate insert(Item, L,R) to insert an Item into list L. The list R contains the result. Repeated calls to insert using the semicolon should insert the item into all possible locations.

?- insert( a, [ d,o,g] , R).

R = [a,d,o,g];

R = [d,a,o,g];

R= [d,o,a,g];

R = [d,o,g,a];

R = false.

Create insert predicate by completing the following outline. Test your code.

insert(X, L , [X|L]).

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

*The next 3 problems MUST be modelled after similar problems done in the Chap 16, Part 3 power point slides.*

**Problem 5**

Modify the Pythagorean Triple Problem from the Chap 16,Part 3 power point slides so that it finds all lists [A, B, C ] satisfying the conditions

A,B,C are between 1 and N, A <=B<=C, gcd(A,B) = 1, A\*A + B\*B =:= C\* C.

[Hint: start your predicate like this

pythTriple( [A,B,C], N ) :- *put conditions here*

Examples:

?- pythTriple(L, 20).

L = [3, 4, 5] ;

L = [5, 12, 13] ;

L = [8, 15, 17] ;

Example:

?- pythTriple(L, 40). //left for second test case

pythTriple([X,Y,Z], N) :-

length(\_, N),

between(1,N,X),

between(1,N,Y),

between(1,N,Z),

Z\*Z =:= X\*X + Y\*Y.

**Problem 6**

Create a Prolog predicate to find a 3x3 magic square. An 3x3 **magic square** is a **square** grid (3 cells on each side, See below) filled with positive integers in the range 1 to 9 , such that each cell contains a different integer and the sum of the integers in each row, column and diagonal are equal. In this case the row sums , columns sums and diagonal sums must each be 15. Label your 9 grid cells as follows

|  |  |  |
| --- | --- | --- |
| X1 | X2 | X3 |
| X4 | X5 | X6 |
| X7 | X8 | X9 |

**Requirement**: Model your solution on the Math Puzzle problem from the Chap 16, Part 3 power point slides. Do not try to make the solution efficient. I am looking for a brute force solution like the Math Puzzle Problem. Use permutation. Also do not copy a solution from the internet.

Example ?- magicSquare(X1, X2,X3, X4, X5, X6, X7, X8, X9) returns values for X1, X2, …, X9 satisfying the constraints. You should find AND SUBMIT THE FIRST 3 SOLUTIONS when you test your answer. How many different solutions to magicSquare does your program find? Don’t print them all.

**Solution**

magicSquare(X1, X2, X3, X4, X5, X6, X7, X8, X9):-

permutation([1,2,3,4,5,6,7,8,9], [X1, X2, X3, X4, X5, X6, X7, X8, X9]),

X1 + X2 + X3 =:= 15, X4 + X5 + X6 =:= 15, X7 + X8 + X9 =:= 15,

X1 + X4 + X7 =:= 15, X2 + X5 + X8 =:= 15, X1 + X5 + X9 =:= 15,

X1 + X5 + X9 =:= 15, X3 + X5 + X7 =:= 15.

**First 3 Solutions:**

X1 = 2,

X2 = 7,

X3 = 6,

X4 = 9,

X5 = 5,

X6 = 1,

X7 = 4,

X8 = 3,

X9 = 8

X1 = 2,

X2 = 9,

X3 = 4,

X4 = 7,

X5 = 5,

X6 = 3,

X7 = 6,

X8 = 1,

X9 = 8

X1 = 4,

X2 = 3,

X3 = 8,

X4 = 9,

X5 = 5,

X6 = 1,

X7 = 2,

X8 = 7,

X9 = 6

The program prints 9 total solutions.

**Problem 7**

Modify the Subset Sum problem from Char 16,Part 3 power points so that it finds all subsets of a set A that sum to M and have length N. Call the predicate subsetSum2 (A, B, M,N). B is instantiated to the answer if there is one; otherwise predicate returns false.

Example for Problem 8

?- subsetSum2([2,4,6,8,10], B, 10, 2).

B = [2, 8] ;

B = [4, 6] ;

false.

Second test case follows.

?- subsetSum2([ 3,5,7,8,4,9,1, 6, 10, 20, 22], B, 32, 4).

**Note: You MUST use the subsets and sumList predicates from the Chap 16, Part 3 power point slides. The Prolog built-in ‘subset’ predicate will not generate subsets.**