COSC 2P93 Prolog Assignment #1

Due date: 12:00 noon, Friday January 25.
Lates: accepted 12:00 noon Monday January 28, -25%; not accepted afterwards.
Hand in: Printouts of your programs and adequate examples of their execution. Submit your source code and data via "submit2p93" on sandcastle. (See end of assignment for details).

1. Family tree.

Write Prolog predicates which represent the following family relationships: brother, sibling, father, mother, wife, husband, married, sister-in-law, mother-in-law, great-grandmother, ancestor, aunt, cousin, related, unrelated. You may use additional definitions if they help, as well as ones from the textbook. Some of the relations will be rules, while others will be facts. Try to use the minimal number of fact relations, while using existing rules whenever possible. Include a database describing your own family. Test all the rules and facts on your own family tree.

2. Unification.

Do the following expressions unify? If yes, give their variable substitutions. If not, briefly discuss why. Work them out by hand, and verify your answers with Sicstus.

(a) apples=Oranges
(b) Apples=Oranges
(c) apples=oranges
(d) addr(main, l2s3a1) = addr(A, B)
(e) X=1, Y=cat, Z=hello(X,Z)
(f) what(do, You, know) = what(DoYou, Know)
(g) A=a, B=b, C=A, List=[A, B, C]
(h) A=a, B=A, [A | [B]] = List
(i) [First, Second|Rest] = [1,2,3,4,5]
(j) List = [1,2,3], List = [H | T]

3. Unification & Arithmetic.

What will Prolog's answers be to the following? Work them out by hand, and verify them in Prolog.

(a) R is 5, Area is 3.14* R*R.
(b) [A, B, C] = [1, 2, 3], Sum is A+B+C.
(c) ferrari(foo, 25) = ferrari(X, Y), Z is Y+1.
(d) X = 25*50.5, Y is 25+50.5.
(e) [1, X, 3]= [H | T], X is H/2.0.
(f) 5 is A-B, A=10, B is 5.
(g) A=10, B is 5, 5 is A-B.
(h) Cat = 2, 4 is cat*2.
(i) X is [2, 3]*4.
(j) X is 100 / 0.

4. Simple math program

Write a predicate *quadratic*/4. Given an equation $ax^2 + bx + c = 0$, it computes the solution to x using the quadratic formula:

$$x = \frac{-b \pm \sqrt{(b^2 - 4ac)}}{2a}$$

The predicate goal is *quadratic(A, B, C, X),* where A, B, C are the formula coefficients, and X is the solution. If there are 2 solutions to the expression, then each is returned separately via backtracking. If the formula in the square root is negative, the query results in "no".

Make sure that your predicates in (a) and (b) behave well with backtracking, and that it won't give errors for unexpected argument values. You may find the builtin predicates number/1 and others useful for error checking.

Comments:

1. Each program file should begin with a comment with your name, student number, course and assignment number, and question number.

2. Each predicate you write must begin with a comment section describing:
   - the arguments to the predicate (intended values, and whether they are input, output, or both)
   - what the predicate does

3. Always use descriptive identifiers (predicate names, variable names, constants). Spread large predicates out onto multiple lines. You will be marked on the correctness of your programs, the completeness of your test cases, and general programming style (readability, inline documentation).

4. Thoroughly test every predicate. Run them on difference cases. Use the linux "script" utility to save the terminal I/O. Print the scripted file.

5. Submit your entire assignment directory for marking using the command "submit2p93". This will recursively submit all your code and data to the marker. You should call it in the top directory of your 2P93 assignment folder.