CSCI 3210 Project 5

Due: see class calendar

Goal: To be familiar with Prolog, a logic programming language .

Description:

In this assignment, you need to solve the following problems using Prolog.

1. Design rules to find out if a value is a member of a list. The predicate should be in the format: **mymember(value, list)**. For example, mymember(X, [1, 2, 3, 4]) should return X = 1.

2. Design rules to check if an integer is an even number or not. The predicates should be in the format: **myeven(a_number).**

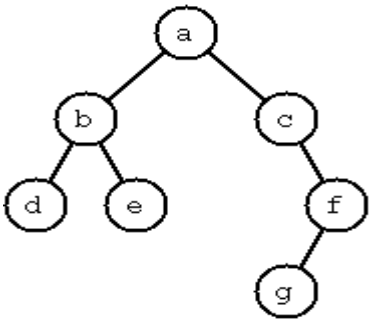
3. Design rules to find out the number of even values in a list of integers. The predicates should be in the format: **myevennumber(a_number, list).** For example, myevennumber(X, [1, 2, 3, 4, 5, 6]) should return X = 3.

4. Design rules to find the minimum value in a list of integers. The predicates should be in the format: **myminlist(list, minimum_value)**. For example, myminlist([1,2,0,4,5], X) should return X = 0.

5. Design rules to find out whether a list is a palindrome. A palindrome can be read forward or backward; e.g. [x,a,m,a,x]. The predicates should be in the format: **palindrome(list).**

6. A binary tree is either empty or it is composed of a root element and two successors, which are binary trees themselves. In Prolog we represent the empty tree by the atom 'nil' and the non-empty tree by the term **t(X,L,R)**, where X denotes the root node and L and R denote the left and right subtree, respectively. The following Prolog term represents the given binary tree below.

   T1 = t(a,t(b,t(d,nil,nil),t(e,nil,nil)),t(c,nil,t(f,t(g,nil,nil),nil)))



   Please design rules to find out the number of leave nodes in a binary tree. A leaf node is a node without successors. The predicates should be in the format: **leafcount(T,N),** where T has N leaf nodes. For example leafcount(t(a,t(b,t(d,nil,nil),t(e,nil,nil)),t(c,nil,t(f,t(g,nil,nil),nil))), X) should return X = 3.

7. Puzzle: Farmer needs to bring a wolf, a goat, and a cabbage across the river from west bank to east bank. The boat is tiny and can only carry one passenger at a time. If he leaves the wolf and the goat alone together, the wolf will eat the goat. If he leaves the goat and the cabbage alone together, the goat will eat the cabbage. Write a prolog program to find the solution..

Requirement:

- You are not allowed to use any built-in Prolog functions that are not covered in class.
- You must use the given predicate formats
- All solutions should be in one file
- Comments are required for each user-defined predicate.

Tips:

Problem 2:
you probably need to use the **`is`** built-in predicate, which is used in Prolog to force the **evaluation** of arithmetic expressions. If you just write something like `X = 2 + 4`, the result is to bind `X` to the unevaluated term `2 + 4`, not to `6`. Example:

```
?- X = 2 + 4.
X = 2+4
```

If instead you write `X is 2 + 4`, Prolog arranges for the second argument, the arithmetic expression `2 + 4`, to be evaluated (giving the result `6`) before binding the result to X.

```
?- X is 2 + 4.

X = 6
```

Problem 7
Use a list like [Farmer, Wolf, Goat, Cabbage] as the configuration of all objects. If w denotes the West bank and e denotes the East bank, then the initial state is: [w,w,w,w]. If the farmer takes the wolf across, then the configuration becomes: [e,e,w,w] (and the goat eats the cabbage). The desired final configuration is: [e,e,e,e] (everyone is on the East bank).

The final predicate should be **solution(Config,MoveList)** where Config represents the current configuration, and MoveList is a list of moves. Each move is wolf, goat, cabbage, or nothing. If the Move is wolf, it means the farmer takes the wolf to cross the river. If the move is nothing, it means the farmer crosses the river by himself.

The other predicates you need to design including:

- safe(Config) returns true if current configuration is safe, i.e. nothing got eaten.
- move(Config, Move, NextConfig): takes the given move and change the current safe configuration to next safe configuration.

Your Prolog program may not specify how long the solution has to be. In fact, a solution could be arbitrary long (e.g. insert an infinite number of nothing moves when the goat is on one side and the wolf and the cabbage on the other). But if Prolog is asked for a solution of a specific length, it will oblige: To find the shortest solution, use the following command:
length(X, 7), solution([w,w,w,w], X).

How to install Prolog?

Download SWI-Prolog from http://www.swi-prolog.org/Download.html and install it according to the instruction. Use the editor that you are familiar with to edit Prolog file, and run it using SWI-Prolog.

How to submit?

- Copy the rubric3.doc to the project5 folder in your individual repository. Edit the file to put your name.
- Commit the whole project5 folder to the server. Make sure the folder contains your source program.
- **Any commit of the project after the deadline is considered as cheating. If this happens, the latest version before the deadline will be graded, and you may receive up to 50 points deduction.**
- **No hard copy needed**