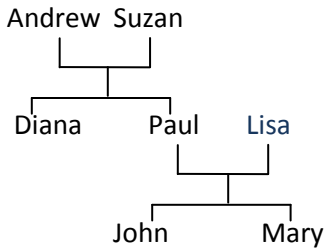
CSE 3401 - Exercise set 1

Updated on January 16, 2013

Ex1. Download the file family.txt from the website, and rename it to family.pl.

a. Add the required facts to the knowledgebase to represent the following family tree:



b. Provide a definition for **`sisterOf(X,Y),`** such that Y is sister of X if they share the same parent and Y is female.

c. Similarly, provide definitions for **`brotherOf(X,Y), siblingOf(X,Y), auntOf(X,Y)`** and **`grandParentOf(X,Y).`**

d. Run various queries to examine the family relationships. Enter semicolon (**`;`**) to get all the answers. Did you notice that the query **`sisterOf(X,mary)`** could have X=mary as a result? (Mary is her own sister!)

e. What is the query that will return son(s) of Andrew?

Ex2. Write a predicate **`listsOverlap(List1, List2)`** that succeeds iff List1 and List2 have at least one common element . To test this predicate at the command prompt, type a query such as:

```
?- listsOverlap([1,2,3],[3,4]).
True.
```

Ex3.

a. Write a predicate **`firstListElement(List1, FirstElement)`** which takes as input List1, and returns the first element of the list, example:

```
?- firstListElement([1,2,3], E).
E=1.
```

b. Write a predicate **`lastListElement(List1, LastElement)`** that which takes as input List1, and returns the last element of the list, example:

```
?- lastListElement([1,2,3], E).
E=3.
```

Ex4. Define a predicate **`every2ndElement(List1, List2)`** which takes List1 as its first argument and returns List2 which consists of every other element in List1, starting at the second element, for example:

```
?-  every2ndElement([1, 2, 3], L2).
L2=[2].
?-  every2ndElement([1, 2, 3, 4], L2).
L2=[2,4].
```

Ex5. Write a predicate **`sumListElements(List1, Sum)`** which takes List1 (a list of numeric atoms) as its input and returns in Sum the sum of values of elements in the list, for example:

```
?-  sumListElements ([1, 2, 3], Sum).
Sum=6.
```

Ex6. Assume we have the following knowledgebase:

```
F1: hasWings(chilly).
F2: hasWings(tweety).
F3: hasWings(jumbo).
F4: hasfeather(chilly).
F5: hasfeather(tweety).
F6: flies(tweety).
F7: flies(jumbo).
R1: normalBird(X):- hasWings(X), hasfeather(X), flies(X).
```

a. Write the search tree for the query **`normalBird(Result)`** to find out what Prolog returns as solutions to Result.

b. What if the user presses **`;`** after getting an answer for the **`Result`**?

c. Use **`trace`** to compare your results to the tree provided.