Project PL5

Due Date: December 7

Purpose

This is lil' project is meant to help you learn a bit of PROLOG. And to have some fun with some German names (see below).

Problem

PROLOG is especially good at searching databases. It is very easy to construct facts (axioms) to store in the database, and quite straightforward to perform simple queries on those facts. (It can get harder when the path from a query to a fact is not easy to see.) You will implement a small family tree and be able to ask questions about it in PROLOG.

Input

The PROLOG program itself will store the family tree; thus, there is no user "input" in the usual sense. The user will perform queries on the tree as described below. The axioms follow the format: "the mother or father of X is Y," as in `mother(michael,marie)` = "the mother of Michael is Marie." The database should be made up of the following *true* facts (you'll love some of these names):

```
mother(michael,marie)      mother(mark,gail)         mother(sonja,christl)
father(michael,larry)      father(mark,george)       father(sonja,heinz)
mother(marie,elise)        mother(kim,gail)          mother(elke,christl)
father(marie,bernhard)     father(kim,george)        father(elke,heinz)
mother(larry,beatrice)     mother(bruce,gail)        mother(max,elise)
father(larry,gerard)       father(bruce,george)      father(max,bernhard)
mother(berta,elise)        mother(nicole,addie)      mother(reinhard,irmgard)
father(berta,bernhard)     father(nicole,norman)     father(reinhard,max)
mother(hans,elise)         mother(meleita,addie)     mother(lisa,regina)
father(hans,bernhard)      father(meleita,norman)    father(lisa,reinhard)
mother(norman,beatrice)    mother(joey,addie)        mother(sabrina,regina)
father(norman,gerard)      father(joey,norman)       father(sabrina,reinhard)
mother(george,beatrice)    mother(hanni,elise)       mother(mj,christine)
father(george,gerard)      father(hanni,bernhard)    father(mj,mark)
mother(christl,elise)      mother(edith,hanni)       mother(john,jennifer)
father(christl,bernhard)   father(edith,erwin)       father(john,bruce)
mother(aster,jen)          mother(adam,jen)          mother(bettina,christl)
father(aster,michael)      father(adam,michael)      father(bettina,heinz)
mother(jan-ole,elke)       mother(liesl,elke)        mother(raymond,tina)
father(jan-ole,olaf)       father(liesl,olaf)        father(raymond,herve)
```

And just to be a bit more annoying:

```
brother(gerard,herve)
```

You may wish to draw out this family tree to more clearly see all the relationships.

Output

The user can ask the following queries based on the family tree, where the queries follow the format above: "the <relation> of X is Y," as in `parent(X,Y)` = a parent of X is Y.

```
mother(X,Y).                  auntoruncle(X,Y).
father(X,Y).                  firstcousin(X,Y).
parent(X,Y).                  greatgrandmother(X,Y).
sibling(X,Y).                 greatgrandfather(X,Y).
grandfather(X,Y).             spouse(X,Y).
grandmother(X,Y).             nieceornephew(X,Y).
grandparent(X,Y).             secondcousin(X,Y).
firstcousinonce(X,Y).
```

The last axiom `firstcousinonce(X,Y)` refers to first cousin once removed. Note that some queries will not work; that is, there may not be sufficient information in the database. In such cases, a PROLOG response of "no" is sufficient. In other queries, there should be multiple answers; for example, `sibling()` and `cousin()` may return several answers, if in fact there are multiple siblings or cousins. The user may type "a" to get `all` of the output or ";" to get just the next answer.

Assume that the mother and father of a child are married. For aunt/uncle and similar, **all** of the aunts and uncles should be found (on both sides of the family).

Specifics

Write PROLOG code by writing short predicates for each problem above. You are not allowed to create any additional facts (axioms). You can use your own predicates in new predicates, however. For example, you will use `mother()` in the predicate to find grandmother.

You are not allowed to add in any new axioms; everything must be found using the given data.

Notes

As always, build this in increments. Be sure one predicate works before going on to the next.

Send your completed program to me via Canvas, as usual, using the naming convention: *lastName*`PL5.pl`, as in `gousiePL5.pl`. The only comments you need are an introductory comment and a short comment for any predicate whose name is not self-explanatory, as may be the case when you need a "helper" function.

No need to submit hard copy for this assignment.

The root of the kingdom is in the state. The root of the state
is in the family. The root of the family is in the person of its head.
– Mencius (372-289 B.C.)