Prolog

We say that a rule $A \leftarrow F$ is *flat* if its body F is a conjunction of literals. A flat rule is *safe* if every variable that occurs in it occurs also in its body in one of the positive literals that do not contain equality. For instance, all rules of our "family program"

$$\begin{array}{l}
Male(A),\\
Male(M),\\
Parent(S,W),\\
Parent(S,A),\\
Parent(W,M),\\
Female(x) \leftarrow \neg Male(x),\\
Brother(x,y) \leftarrow Parent(z,x) \land Parent(z,y) \land Male(x) \land x \neq y
\end{array}$$

are flat; all rules except $Female(x) \leftarrow \neg Male(x)$ are safe.

If all rules of a logic program are safe then it may be possible to use an implementation of the programming language Prolog for determining which ground literals are entailed by the completion of the program. To guarantee the correctness of the answers given by Prolog, we should rearrange the literals in the body of each rule so that the positive literals that do not contain equality come before the other conjunctive terms. (If some of the rules are not safe, or if the order of literals does not satisfy the condition above, then Prolog may return an incorrect answer due to a phenomenon called "floundering.") Specifically,

- if a ground atom A is given as a query, and Prolog answers `yes`, then A is entailed by the program's completion; if Prolog answers `no` then $\neg A$ is entailed;
- if an atom A containing variables is given as a query, and Prolog terminates, then it returns, one be one, all substitutions of object constants for variables such that the corresponding instances of A are entailed by the program's completion (and for every other ground instance of A, its negation is entailed).

In a Prolog input file, non-ASCII symbols are represented as follows:

$\leftarrow$	$\land$	$\neg$	$\neq$
`:-`	`,`	`\+`	`\=`

Variables are always capitalized; constants are not capitalized. Every rule is followed by a period. For instance, here is the family program written as a Prolog input file:

```
% File family.pro
male(a).  male(m).
parent(s,w).  parent(s,a).  parent(w,m).
female(X) :- \+ male(X).
brother(X,Y) :- parent(Z,X), parent(Z,Y), male(X), X\=Y.
```

Here is an example of a query answering session with Prolog:

```
> /p/bin/sicstus
SICStus 3.11.2 (x86-linux-glibc2.2): Wed Jun  2 11:43:47 CEST 2004
Licensed to cs.utexas.edu
| ?- ['family.pro'].
% consulting /v/filer4b/v41q001/vl/tmp/family.pro...
% consulted /v/filer4b/v41q001/vl/tmp/family.pro in module user, 0 msec 1232 bytes
yes
| ?- male(a).
yes
| ?- male(w).
no
| ?- female(a).
no
| ?- female(w).
yes
| ?- brother(a,w).
yes
| ?- brother(a,a).
no
| ?- brother(X,Y).
X = a,
Y = w ? ;
no
| ?- female(X).
no
| ?- halt.
>
```

The query `female(X)` floundered.

We can make the family program safe by rewriting the unsafe rule as

$$Female(x) \leftarrow \neg Male(x) \wedge Person(x),$$

where the new predicate constant *Person* is defined by the facts

$$\begin{array}{l} Person(S), \\ Person(W), \\ Person(A), \\ Person(M). \end{array}$$

Alternatively, *Person* can be defined by the rules

$$\begin{array}{l} Person(x) \leftarrow Parent(x, y), \\ Person(y) \leftarrow Parent(x, y). \end{array}$$

Problem 16^e. Check that for the safe version of the family program, Prolog answers the query `female(X)` correctly.

Problem 17^e. Extend the program from the previous example by definitions of other binary predicates, such as *Uncle*, *Grandmother*, and *Granddaughter*, and use Prolog to test your definitions.