CSc 600-01 (Section 1)

Homework 3 - Logic Programming in Prolog

prepared by Ilya Kopyl

CSc 600 Homework 3 - Logic Programming in Prolog

March 16, 2018

Homework is prepared by: Ilya Kopyl.
It is formatted in LaTeX, using TeXShop editor (under GNU GPL license).
Diagrams are created in LucidChart online editor (lucidchart.com).

1. Write a PROLOG program that investigates family relationships using lists. The facts should be organized as follows:

```
m([first_male_name, second_male_name, ..., last_male_name]).
f([first_female_name, second_female_name, ..., last_female_name]).
family( [father, mother, [child1, child2, ..., child_n]] ).
```

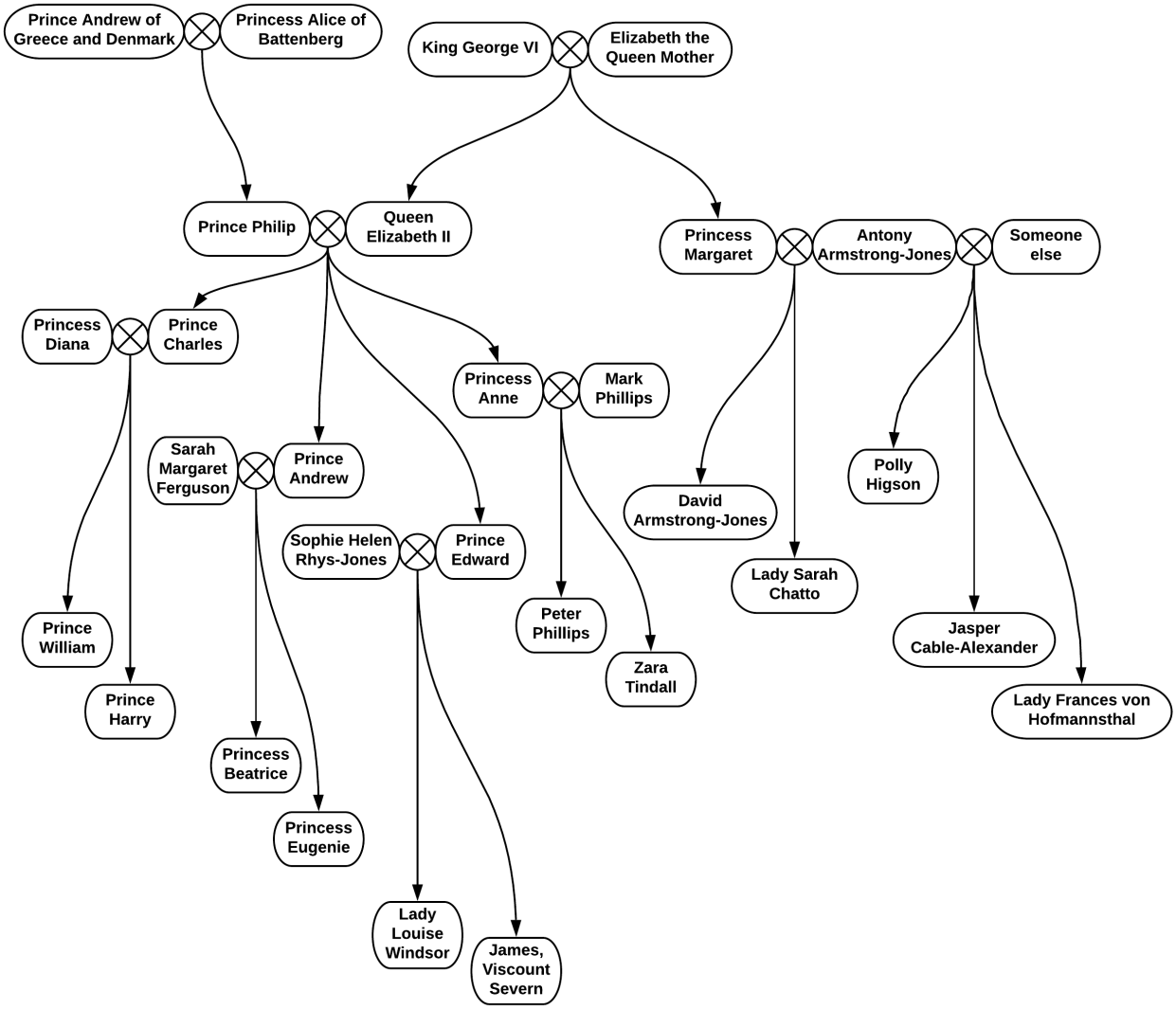
Write rules that define the following relationships:

```
male(X)
female(X)
father, mother, parent
siblings1, siblings2
brother1, brother2
sister1, sister2
cousins
uncle, aunt
grandchild, grandson, granddaughter
greatgrandparent
ancestor
```

For each of these rules show an example of its use.

The answer is listed on the pages TBD through TBD.

Partial diagram of the family tree of the British Royal family.



The entire code listing of problem 1:

```
% Facts:

m([
    'Prince Andrew of Greece and Denmark',
    'King George VI',
    'Prince Philip',
    'Antony Armstrong-Jones',
    'Prince Charles',
    'Mark Philips',
    'Prince Andrew',
    'David Armstrong-Jones',
    'Prince Edward',
    'Peter Philips',
    'Jasper Cable-Alexander',
    'Prince William',
    'Prince Harry',
    'James, Viscount Severn'
]).

f([
    'Princess Alice of Battenberg',
    'Elizabeth the Queen Mother',
    'Queen Elizabeth II',
    'Princess Margaret',
    'Someone else',
    'Princess Diana',
    'Princess Anne',
    'Polly Higson',
    'Sarah Margaret Ferguson',
    'Sophie Helen Rhys-Jones',
    'Lady Sarah Chatto',
    'Zara Tindall',
    'Lady Frances von Hofmannsthal',
    'Princess Beatrice',
    'Princess Eugenie',
    'Lady Louise Windsor'
]).

family([
    'Prince Andrew of Greece and Denmark',
    'Princess Alice of Battenberg',
    [
        'Prince Philip'
    ]
]).
```

```
family([
    'King George VI',
    'Elizabeth the Queen Mother',
    [
        'Queen Elizabeth II',
        'Princess Margaret'
    ]
]).

family([
    'Prince Philip',
    'Queen Elizabeth II',
    [
        'Prince Charles',
        'Princess Anne',
        'Prince Andrew',
        'Prince Edward'
    ]
]).

family([
    'Antony Armstrong-Jones',
    'Princess Margaret',
    [
        'David Armstrong-Jones',
        'Lady Sarah Chatto'
    ]
]).

family([
    'Antony Armstrong-Jones',
    'Someone else',
    [
        'Polly Higson',
        'Jasper Cable-Alexander',
        'Lady Frances von Hofmannsthal'
    ]
]).

family([
    'Prince Charles',
    'Princess Diana',
    [
        'Prince William',
        'Prince Harry'
    ]
]).
```

```
family([
    'Mark Philips',
    'Princess Anne',
    [
        'Peter Philips',
        'Zara Tindall'
    ]
]).

family([
    'Prince Andrew',
    'Sarah Margaret Ferguson',
    [
        'Princess Beatrice',
        'Princess Eugenie'
    ]
]).

family([
    'Prince Edward',
    'Sophie Helen Rhys-Jones',
    [
        'Lady Louise Windsor',
        'James, Viscount Severn'
    ]
]).

% Rules:

% X is male if it is a member of a list m.
male(X) :-
    m(List),
    member(X, List), !.     % the first match would suffice

% Y is female if it is a member of a list f.
female(Y) :-
    f(List),
    member(Y, List), !.

% parent is either a first or second element of the list family
parent(X, Child) :-
    family([X, _, Children]),
    member(Child, Children);
    family([_, X, Children]),
    member(Child, Children).
parent(X) :-
    parent(X, _), !.
```

```
% X is father if X is a parent and is male.
father(X) :-
    parent(X),
    male(X).

% Y is a mother if Y is a parent and is female.
mother(Y) :-
    parent(Y),
    female(Y).

% X and Y are siblings2 if they have both parents in common
siblings2(X, Y) :-
    family([_,_,Children]),
    member(X, Children),
    member(Y, Children),
    X \= Y.

% X and Y are siblings1 if they come from
% different families, but have 1 parent in common.
siblings1(X, Y) :-
    family([Father,_,Children1]),
    family([Father,_,Children2]),
    Children1 \= Children2,
    member(X, Children1),
    member(Y, Children2).

siblings1(X, Y) :-
    family([_,Mother,Children1]),
    family([_,Mother,Children2]),
    Children1 \= Children2,
    member(X, Children1),
    member(Y, Children2).

% X is brother1 to Y if X is sibling1 to Y and is male.
brother1(X, Y) :-
    siblings1(X, Y),
    male(X).

% Y is sister1 to Y if X is sibling1 to Y and is female.
sister1(X, Y) :-
    siblings1(X, Y),
    female(X).

% X is brother2 to Y if X is sibling2 to Y and is male.
brother2(X, Y) :-
    siblings2(X, Y),
    male(X).
```

```
% X is sister2 to Y if X is sibling2 to Y and is female.
sister2(X, Y) :-
    siblings2(X, Y),
    female(X).

% X and Y are cousins if a Parent of X is a sibling2 to a Parent of Y
cousins(X, Y) :-
    parent(Parent1, X),
    parent(Parent2, Y),
    siblings2(Parent1, Parent2).

% X is uncle of Y if X is a brother of a parent of Y
uncle(X, Y) :-
    brother2(X, Parent),
    parent(Parent, Y).

% X is aunt of Y if X is a sister of a parent of Y
aunt(X, Y) :-
    sister2(X, Parent),
    parent(Parent, Y).

% child X of Y if Y is a father of X
child(X, Y) :-
    parent(Y, X).

% X is grandchild of Y if Y is a parent of a parent of X.
grandchild(X, Y) :-
    parent(Parent, X),
    parent(Y, Parent).

% X is grandson of Y if X is grandchild of Y and is male.
grandson(X, Y) :-
    grandchild(X, Y),
    male(X).

% X is a granddaughter of Y if X is granddaughter of Y and if female.
granddaughter(X, Y) :-
    grandchild(X, Y),
    female(X).

% X is a grandparent of Y if X is a parent of a parent of Y
grandparent(X, Y) :-
    parent(X, Parent),
    parent(Parent, Y).

% X is a greatgrandparent of Y if X is a parent of a grandparent of Y
greatgrandparent(X, Y) :-
    parent(X, Parent),
    grandparent(Parent, Y).
```

```
% Base case: X is ancestor of Y if X is a parent of Y
ancestor(X, Y) :-
    parent(X, Y).

% Recursive defintion:
ancestor(X, Y) :-
    parent(X, Parent),
    ancestor(Parent, Y).
```

The code listing of main program:

```
somePredicate(A, B) :-
    arbitraryPredicate(A, _, 1, 2),
    predicateWithAtom(someAtom),
    anotherPredicate(B, someAtom, myPredicate(A, _)),
    findall(X, ('testString'(X), myPredicate(A, X)), L1),
    member(A, L1),
    !.
```

The result of the program execution:

```
Standard output:
```

2. Write a PROLOG program that includes the following operations with lists:

membership testing (is an element member of a list?)
first element
last element
two adjacent elements
three adjacent elements
append list1 to list2 producing list3
delete element from a list
append element to a list
insert element in a list
compute the length of list
reverse a list

```
check whether a list is a palindrome
display a list
```

For each of these operations write your implementation of the operation and show an example of its use. If a predicate already exists (predefined in Prolog), modify its name (e.g. myappend or append1). Lists to be processed can be created by an auxiliary program, defined as facts, or entered from the keyboard.

The answer is listed on the pages TBD through TBD.

The code listing of the two-dimensional array that stores bit pattern of each BigInt digit. It is declared in the global space (outside of any function).

```
somePredicate(A, B) :-
    arbitraryPredicate(A, _, 1, 2),
    predicateWithAtom(someAtom),
    anotherPredicate(B, someAtom, myPredicate(A, _)),
    findall(X, ('testString'(X), myPredicate(A, X)), L1),
    member(A, L1),
    !.
```

Main program, excluding the declaration of BIG_DIGITS array:

```
somePredicate(A, B) :-
    arbitraryPredicate(A, _, 1, 2),
    predicateWithAtom(someAtom),
    anotherPredicate(B, someAtom, myPredicate(A, _)),
    findall(X, ('testString'(X), myPredicate(A, X)), L1),
    member(A, L1),
    !.
```

3. Write a PROLOG program that solves the 8 queens problem (location of 8 queens on a chess board so that no queens have each other in check, i.e. are not located in the same row/column/diagonal).

The answer is listed on the pages TBD through TBD.