Name/

Najd Nader Alsubaie

ID/

431001012

1. Ask Prolog if Albert is the parent of Peter by entering the query.

   Answer:

   ```
   ?- parent(albert, peter).
   ```

   out:

   true

2. Ask Prolog if Albert is the parent of Brian by entering the query:

   Answer:

   ```
   ?- parent(albert, brian).
   ```

   out:

   false

3. Ask, "Who are the parents of Brian?":

   Answer:

   ```
   ?- parent(X, brian)
   ```

   out:

   ```
   X = jim
   X = pat
   ```

4. Ask, "Is Irene a grandparent of Brian?" This can be answered by finding out if Irene is the parent of someone who is the parent of Brian.

   Answer:

```
?- grandparent(irene, brian).
```

```
out:

true
```

5. Find all the grandchildren of Irene

   Answer:

```
?- grandparent(irene, X).
```

```
out:

X = brian
X = lee
X = sandra
X = james
X = kate
X = kyle
```

6. Add rule to define grandparent relation.

   Answer:

```
grandparent(Grandparent, Grandchild) :-
    parent(Grandparent, Parent),
    parent(Parent, Grandchild).
```

7. Add rule to define who is the older.

Answer:

```
older(A, B) :-
    yearOfBirth(A, YearA),
    yearOfBirth(B, YearB),
    YearA < YearB.
```

8. Using the rules that you are defined in 7, and 6 ,(siblings, older), define new rule: bigBrother. Hint the big brother should be male.

Answer:

```
bigBrother(Brother, Sibling) :-
    siblings(Brother, Sibling),
    male(Brother),
    older(Brother, Sibling).
```

Test:

```
?- bigBrother(james, kate).
```

out:

```
true
```

The Latest Version Of The Program

```
% parent(Parent, Child)
parent(albert, jim).
parent(albert, peter).
parent(jim, brian).
parent(john, darren).
parent(peter, lee).
parent(peter, sandra).
parent(peter, james).
parent(peter, kate).
parent(peter, kyle).
parent(brian, jenny).
parent(irene, jim).
parent(irene, peter).
parent(pat, brian).
parent(pat, darren).
parent(amanda, jenny).

% female(Person)
female(irene).
female(pat).
female(lee).
female(sandra).
female(jenny).
female(amanda).
female(kate).

% male(Person)
male(albert).
male(jim).
male(peter).
male(brian).
male(john).
male(darren).
male(james).
male(kyle).
```

```prolog
% yearOfBirth(Person, Year)
yearOfBirth(irene, 1923).
yearOfBirth(pat, 1954).
yearOfBirth(lee, 1970).
yearOfBirth(sandra, 1973).
yearOfBirth(jenny, 1996).
yearOfBirth(amanda, 1979).
yearOfBirth(albert, 1926).
yearOfBirth(jim, 1949).
yearOfBirth(peter, 1945).
yearOfBirth(brian, 1974).
yearOfBirth(john, 1955).
yearOfBirth(darren, 1976).
yearOfBirth(james, 1969).
yearOfBirth(kate, 1975).
yearOfBirth(kyle, 1976).


% ------------------------------
% RULES
% ------------------------------

% grandparent(Grandparent, Grandchild)
grandparent(Grandparent, Grandchild) :-
    parent(Grandparent, Parent),
    parent(Parent, Grandchild).

% older(A, B)
older(A, B) :-
    yearOfBirth(A, YearA),
    yearOfBirth(B, YearB),
    YearA < YearB.

% siblings(A, B)
siblings(A, B) :-
    parent(X, A),
    parent(X, B),
    A \== B.
```

```
% sibling_list(Child, Siblings)
sibling_list(Child, Siblings) :-
    findall(Sibling, siblings(Child, Sibling), List),
    remove_duplicates(List, Siblings).

% remove_duplicates(List, Result)
remove_duplicates([], []).
remove_duplicates([X|Rest], Result) :-
    member(X, Rest), !,
    remove_duplicates(Rest, Result).
remove_duplicates([X|Rest], [X|Result]) :-
    remove_duplicates(Rest, Result).

% bigBrother(Brother, Sibling)
bigBrother(Brother, Sibling) :-
    siblings(Brother, Sibling),
    male(Brother),
    older(Brother, Sibling).

% descendant(Person, Descendant)
descendant(Person, Descendant) :-
    parent(Person, Descendant).
descendant(Person, Descendant) :-
    parent(Person, Child),
    descendant(Child, Descendant).

% ancestor(Person, Ancestor)
ancestor(Person, Ancestor) :-
    parent(Ancestor, Person).
ancestor(Person, Ancestor) :-
    parent(Parent, Person),
    ancestor(Parent, Ancestor).

% children(Parent, ChildList)
children(Parent, ChildList) :-
    findall(Child, parent(Parent, Child), ChildList).

% listCount(List, Count)
listCount([], 0).
listCount([_|Tail], Count) :-
```

```
    listCount(Tail, TailCount),
    Count is TailCount + 1.

% countDescendants(Person, Count)
countDescendants(Person, Count) :-
    findall(Desc, descendant(Person, Desc), List),
    listCount(List, Count).
```