1. Investigating Prolog
   (1) Reorder

male(tom).
male(brian).
male(kevin).
male(zhane).
male(fred).
male(jake).
male(bob).
male(stephen).
male(paul).

parent(tom , stephan).
parent(stephen , jennifer).
parent(zhane , melissa).
parent(zhane , mary).
parent(melissa , brian).
parent(mary , sarah).
parent(bob , jane).
parent(tom, mary).
parent(paul , kevin).
parent(jake , bob).
parent(emily , bob).
parent(stephen , paul).

Explanation: For I move male() fact to the first. For parent() fact, since it is "parent(X , Z) , parent(Z , Y)", I move parent(tom , stephen) to the first and parent(stephan , jennifer) to the second to make query faster.

   (2) Show evidence of faster execution time.

Let's see the screenshot for implement "trace." for grandfather(X,Y) :- male(X), parent(X,Z),parent(Z,Y).

Original Execution Time:

```
[trace]  ?- grandfather(tom,jennifer).
   Call: (8) grandfather(tom, jennifer) ? creep
   Call: (9) male(tom) ? creep
   Exit: (9) male(tom) ? creep
   Call: (9) parent(tom, _7240) ? creep
   Exit: (9) parent(tom, mary) ? creep
   Call: (9) parent(mary, jennifer) ? creep
   Fail: (9) parent(mary, jennifer) ? creep
   Redo: (9) parent(tom, _7240) ? creep
   Exit: (9) parent(tom, stephen) ? creep
   Call: (9) parent(stephen, jennifer) ? creep
   Exit: (9) parent(stephen, jennifer) ? creep
   Exit: (8) grandfather(tom, jennifer) ? creep
true
```

Reorder Execution Time:

```
[trace]  ?- grandfather(tom , jennifer).
   Call: (8) grandfather(tom, jennifer) ? creep
   Call: (9) male(tom) ? creep
   Exit: (9) male(tom) ? creep
   Call: (9) parent(tom, _7592) ? creep
   Exit: (9) parent(tom, stephan) ? creep
   Call: (9) parent(stephan, jennifer) ? creep
   Exit: (9) parent(stephan, jennifer) ? creep
   Exit: (8) grandfather(tom, jennifer) ? creep
true
```

It is obviously that the execution time for reorder one is faster than that of the original one.

(3) **grandmother?**

No, we cannot applied the above rules to arrive an answer for relation grandmother. The reason is the following. First, there is no universe facts for female and male. For example, we cannot make sure that ‘melissa’ is male or female even though this name is female one in common. Therefore, we are supposed to form up complete universe of facts.

(4) Define **aunt** and **uncle**.

See attachement.

2. Prolog Rules

See attachment.

3. Unification

(1) d(15) & c(X)

Two functors are different.

(2) 42 & 23

Constant 42 is not equal to constant 23, so they cannot unify.

(3) a(X, b(3, 1, Y)) & a(4, Y)

Infinite recursion: Y = f(Y).

(4) a(X, c(2, B, D)) & a(4, c(A, 7, C))

X = 4

c(2 , B , D) = c(A , 7 , C)

A = 2

B = 7

D = C

(5) a(X, c(2, A, X)) & a(4, c(A, 7, C))

Cannot unify because that 2 is not equal to 7, which conflicts.

(6) e(c(2, D)) & e(c(8, D))

Though the pattern is identical, but constant 2 cannot unify with constant 8.

(7) X & e(f(6, 2), g(8, 1))

X = e(f(6, 2), g(8, 1))

(8) b(X, g(8, X)) & b(f(6, 2), g(8, f(6, 2)))

X = f(6,2)

g(8, X) = g(8, f(6, 2))

(9) a(1, b(X, Y)) & a(Y, b(2, c(6, Z), 10))

The arity of functor b on both side is not identical.

(10) d(c(1, 2, 1)) & d(c(X, Y, X))

d(c(1,2,1)) = d(c(X,Y,X))

X = 1

Y = 2

4. Prolog Adventure

See attachment.

5. Prototype OOLs

(1) x

(2) y

(3) z

(4) x

(5) obj1.x = 20

(6) obj2.x = 20

(7) obj3.x = 20

(8) obj4.x = 10

(9) No this field.

(10) obj2.y = 5

(11) obj3.y = 5

(12) obj3.z = 30