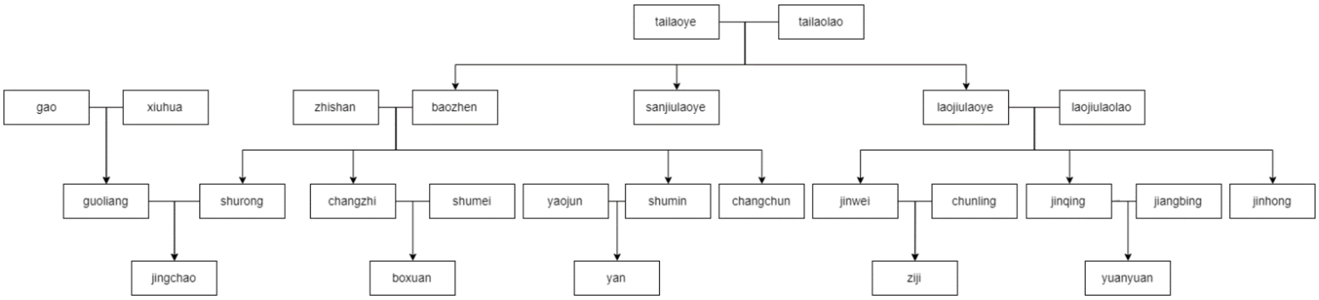
Comp 348 Assignment 1

Jingchao Zhang 40049474

Q1:

```
individual(jingchao, male, guoliang, shurong).

individual(guoliang, male, gao, xiuhua).

individual(shurong, female, zhishan, baozhen).

individual(changzhi, male, zhishan, baozhen).

individual(shumin, female, zhishan, baozhen).

individual(changchun, male, zhishan, baozhen).

individual(boxuan, male, changzhi, shumei).

individual(yan, female, yaojun, shumin).

individual(baozhen, female, tailaoye, tailaolao).

individual(laojiulaoye, male, tailaoye, tailaolao).

individual(sanjiulaoye, male, tailaoye, tailaolao).

individual(jinwei, male, laojiulaoye, laojiulaolao).

individual(jinqing, female, laojiulaoye, laojiulaolao).

individual(jinhong, female, laojiulaoye, laojiulaolao).

individual(ziji, male, jinwei, chunling).

individual(yuanyuan, female, jiangbing,  jinqing).
```



1. offspring(X, Y) :- individual(X, _ , Y, _); individual(X, _ , _ ,Y).
2. sibling(X, Y) :- individual(X, _, F, M), individual(Y, _, F, M), X \= Y.
   niblings(X, Y) :- offspring(X, P), siblings(Y, P).
3. puncle(X, Y) :- individual(Y, _, F, _), individual(F, male, G1, G2), individual(X, male, G1, G2), F \= X .
4. modrige(X, Y) :- individual(Y, _, _, M), individual(M, female, G1, G2),individual(X, female, G1, G1), M \= X.
5. avuncle(X, Y) :- individual(Y, _, _, M), individual(M, female, G1, G2),individual(X, male, G1, G2).

Q2:

1. likes(jane, X) = likes(X, josh).

   Fail. First variable X is instantiated to jane, but jane is not josh.
2. diSk(27, queens, sgt_pepper) = diSk(A, B, help).

   Fail. A is instantiated to 27, B is instantiated to queens, but sqt_pepper ≠ help.
3. [a,b,c] = [X,Y,Z|T].

   Success. X is instantiated to a, Y is instantiated to b, Z|T is instantiated to c. Z is the head, c. T is instantiated to an empty list, [ ].
4. ancestor(french(jean), B) = ancestor(A, irish(joe)).

   Success. A = french(jean), B = irish(joe).
5. characters(hero(luke), X) = characters(X, villain(vader)).

   Fail. X = hero(luke), but hero(luke) ≠ villain(vader).
6. f(X, a(b,c)) = f(d, a(Z, c)).

   Success. X is instantiated to d, a(Z, c) is instantiated to a(b, c). Then Z is unified to b.

7. s(x, f(x), z) = s(g(y), f(g(b)), y).

   Fail. x is not g(y).

8. vertical(line(point(X,Y), point(X,Z))) = vertical(line(point(1,1),point(1,3))).

   Success. line = line, and they both have two arguments, so succeeds. Then for the first argument, point(X, Y) is unified with point(1,1), where X = Y and Y = 1. For the second argument, point(X, Z) is unified with point(1,3). X is still 1, and Z = 3.

9. g(Z, f(A, 17, B), A+B, 17) = g(C, f(D, D, E), C, E).

   Success. C = Z, A = D, D = 17, B = E, A + B = C, E = 17,

   So, A = D = 17, B = 17, C = Z = 17+17

10. f(c, a(b,c)) = f(Z, a(Z, c)).

    Fail. Z = c, then a(b,c) = a(Z,c). Z is substituted with c, but c ≠ b, so fail.

Q3:

1. building(library, lb).

   Ground query. True. It searches the database and finds this fact in the third line.

2. status(finance, A).

   Non-ground query. False. The data base has only one fact, status(engineering, accredited), which cannot unify this query. So it matches the rule, status(X, Z) :- department(X,Y), status(Y,Z). and unified X = finance, Z = A. Then plug in and resolve the conditions, department(finance, Y) and status(Y, A) must both be true. department(finance, Y) can be unified by department(finance, business), so Y = business. The second condition becomes status(business, A). Again, it looks for X = business and Z = A, which is true only if department(business, Y) and status(Y, A) are both true. However, there is no department (business, _) in the database, so return false.

3. department(civil, Bussiness).

   Non-ground query. Success. Bussiness = engineering. Bussiness is a variable and instantiated to engineering.

4. faculty(X, civil).

   Non-ground query.

   X = jones

X = james

X = davis

False.

It found three records in the database and instantiated X to jones, james, davis respectively. Then it tried to match the rule, faculty(X, Y) :- department(Z, Y), faculty(X, Z). X = X, Y = civil, but the department(Z, civil) cannot be unified in the database, so it returns false at the end.

5. faculty(smith, X).

   Non-ground query.

   X = electrical

   X = computer

   X = engineering

   False.

   There are two records in the database can unify this query, faculty(smith, electrical). and faculty(smith, computer). Then it tries the rule, faculty(X, Y) :- department(Z, Y), faculty(X, Z). X1 = smith, Y = X. Then it is resolved with department(Z, X) and faculty(smith, Z). There are two items can unify Z to electrical or computer. Only department(electrical,_) exists in the database and X is unified with engineering. So, the third result is X = engineering. Then there is no other result so return false.

6. department(X, Y).

   Non-ground query.

   X = electrical, Y = engineering

   X = civil, Y = engineering

   X = finance, Y = business

   X = ibm-exams, Y = lb

   Plug in the four records and got the answer.

7. faculty(X, civil), department(civil, Y).

   Non-ground query.

   X = jones, Y = engineering

   X = james, Y = engineering

   X = davis, Y = engineering

   False.

There are three records can unify X to jones, james and davis. And there is only one record can unify Y to engineering. Then it tries the rule faculty(X, Y) :- department(Z, Y), faculty(X, Z). Y = civil, but there is no department(Z, civil). So it returns false.

8. faculty(Smith).

   Non-ground query.

   Smith = smith

   Smith = walsh

   Smith = smith

   Smith = jones

   Smith = james

   Smith = davis

   Smith = smith

   Smith = walsh

   Smith = jones

   Smith = james

   Smith = davis

   False.

   It goes to the last rule in the database and only succeeds when faculty(Smith, _) is true. Smith is a variable, so any record with two arguments succeeds. It plugs in the six facts in the database. And then it tries the rule faculty(X, Y) :- department(Z, Y), faculty(X, Z). X = Smith and Y can be anything. 1) Z = electrical, Y = engineering, and faculty(Smith, electrical). Thus Smith = smith and Smith = walsh. 2) Z = civil, Y = engineering, and faculty(Smith, civil). Thus Smith = jones, Smith = james, Smith = davis. 3) Z = finance, Y = business. So faculty(Smith, finance) must be true. There is no such record. 4) Z = ibm-exams, Y = lb. So faculty(Smith, ibm-exams) must be true. Similarly, there is no such a record after the recursive call. So false at the end.

9. building(_, X).

   Non-ground query.

X = ev

X = mb

X = lb

X = h

X = fg

X = ev

X = ev

X = mb

False.

It unifies the first five facts and X = ev, mb, lb, h, fg respectively. Then it goes to the rule building(X, Y) :- department(X, Z), building(Z, Y). X can be anything, Y must be X. This is not the same X. Then we start to check the conditions. 1) X = electrical, Z = engineering. So building(engineering, X). so, X = ev. 2) X = civil, Z = engineering. So building(engineering, X). so, X = ev again. 3) X = finance, Z = business. So building(business, X), so, X = mb. 4) X = ibm-exams, Z = lb. then building(lb, X) must be true. But there is no such a building in facts. So we search the rule building(X, Y) :- department(X, Z), building(Z, Y). X = lb, Y = X, then department(lb, Z) and building(Z, X) must be both true. But there is no such a department. After these four department facts, there is no more department. Therefore it returns false.

10. status(X, accredited), building(X, Y).

Non-ground query.

X = engineering, Y = ev

X = electrical, Y = ev

X = civil, Y = ev

False.

a) The first query can be unified with the fact status(engineering, accredited), where X = engineering. So we check the second query, where the X has been substituted by engineering. So the new query now is building(engineering, Y).

1) building(engineering, Y) can be unified with the first fact and Y = ev.
So **X = engineering and Y = ev**.

2) building(engineering, Y) can also be unified with the rule building(X, Y) :- department(X, Z), building(Z, Y). where X = engineering and Y = Y. then we are looking for department(engineering, Z) and building(Z, Y) must be both true. But there is no department(engineering,_), so fail.

b) The first query can also be unified with the rule status(X, Z) :- department(X, Y), status(Y, Z). where X = X and Z = accredited. This is valid only if department(X, Y) and status(Y, accredited) are both true. We firstly unify department(X, Y)

1) In department(X, Y), X = electrical, Y = engineering
   status(engineering, accredited) can be found in the database. So X can be electrical and we go to the second query building(X, Y), where X = electrical. The new query is building(electrical, Y), and unified by the rule building(X, Y) :- department(X, Z), building(Z, Y). X = electrical, Y = Y. then department(electrical, Z) and building(Z, Y) must be true. Z = engineering. So building(engineering, Y)? So Y = ev. So, **X = electrical, Y = ev**.

2) In department(X, Y), X = civil, Y = engineering
   status(engineering, accredited) can be found in the database, so this is true. So X can be civil and we go to the second query building(X, Y), where X = civil. The new query is building(civil, Y), and unified by the rule building(X, Y) :- department(X, Z), building(Z, Y). X = civil, Y = Y. then department(civil, Z) and building(Z, Y) must be true. Z = engineering. So building(engineering, Y)? So Y = ev. So, **X = civil, Y = ev**.

3) In department(X, Y), X = finance, Y = business
   status(business, accredited) is not a fact. If we use the rule, status(X, Z) :- department(X, Y), status(Y, Z)., then X = business, Z = accredited. Then department(business, Y) and status(Y, accredited) must both be true. But there is no department(business, _), so fail.

4) In department(X, Y), X = ibm-exams, Y = lb
   status(lb, accredited) is not a fact. If we use the rule, status(X, Z) :- department(X, Y), status(Y, Z)., then X = lb, Z = accredited. Then department(lb, Y) and status(Y, accredited) must both be true. But there is no department(lb, _), so fail.

11. status(_, X), building(X, Y).

    Non-ground query.

    False.

    a) unified by the fact status(engineering, accredited). X = accredited. New query building(accredited, Y). No such a fact exists, so use the rule building(X, Y) :- department(X, Z), building(Z, Y). X = accredited, Y = Y. Then department(accredited, Z) and building(Z, Y) must both be true. No such a department, so it fails.
    b) Unified by the rule status(X1, Z) :- department(X1, Y), status(Y, Z). X1 can be anything, Z = X. So department(X1, Y) and status(Y, X) must be true.

        When X1 = electrical, Y = engineering

        ?- status(engineering, X).

        X = accredited, but building (accredited, Y) is false.

        The other three also fails in a similar reason.

12. faculty(X), faculty(X, Y), department(Y, _).

    Non-ground query.

    X = smith, Y = electrical

    X = walsh, Y = electrical

    X = smith, Y = electrical

    X = jones, Y = civil

    X = james, Y = civil

    X = davis, Y = civil

    X = smith, Y = electrical

    X = walsh, Y = electrical

    X = jones, Y = civil

    X = james, Y = civil

    X = davis, Y = civil

    False.

    1) X = smith

       Y = electrical and department(electrical, _). is true.
    2) X = walsh

Y = electrical. and department(electrical, _). is true.

3) X = smith

   Y = electrical and department(electrical, _). is true.

4) X = jones,

   Y = civil and department(civil, _) is true, so this is valid.

5) When X = james.

   faculty(X, Y) is instantiated to Y = civil. and department(civil, _) is true, so this is valid.

   When X = davis, Y = civil and department(civil, _) is true, so this is valid.

6) When using the rule faculty(X, Y) :- department(Z, Y), faculty(X, Z).

   a) Z = electrical, Y = engineering

   It succeeds in this case.

   ?- faculty(X, electrical). So **X = smith, or X = walsh**. **And Y = electrical**

   b) Z = civil, Y = engineering

   ?- faculty(X, civil). **X = jones, or X = james or X = davis**. **And Y = civil**

   c) Z = finance, Y = business

   faculty(X, finance). No such a fact. And also fails the rule.

   d) Z = ibm-exams, Y = lb

   faculty(X, ibm-exams). No such a fact. And also fails the rule.

13. faculty(X), faculty(X, Y), !, department(Y, Z). % note there is a cut (!) here

    Non-ground query.

    X = smith,

    Y = electrical,

    Z = engineering

    First, use rule faculty(X) :- faculty(X,_)

    ?- faculty(X,_). We find the fact faculty(smith, electrical). so, **X = smith**

    Then ?- faculty(smith, Y). So **Y = electrical**.

    Then ?- department(electrical, Z). **Z = engineering.**

    Because there is a ! sign, we don’t go further search and stop.

14. faculty(X), !, faculty(X, _). % note there is a cut (!) here

    Non-ground query.

X = smith

X = smith

X = smith

First use the rule faculty(X) :- faculty(X,_). So X = smith.

Then plug in and check the second query.

?- faculty(smith, _).

There are two facts and one rule can unify this query.

faculty(smith, electrical).

faculty(smith, computer).

In these two facts, X = smith.

In the rule faculty(X, Y) :- department(Z, Y), faculty(X, Z).

X = smith, Y can be anything.

?- faculty(smith, Y) :- department(Z, Y), faculty(smith, Z).

a) Z = electrical, Y = engineering

   It succeeds in this case. So **X = smith**.

b) Z = civil, Y = engineering

   faculty(smith, civil). No such a fact. Try the rule faculty(X, Y) :- department(Z, Y), faculty(X, Z). X = smith, Y = civil. But no department(Z, civil). So it fails.

c) Z = finance, Y = business

   faculty(smith, finance). No such a fact. And also fails the rule.

d) Z = ibm-exams, Y = lb

   faculty(smith, ibm-exams). No such a fact. And also fails the rule.

15. department(X, _), \+ faculty(_, X).

Non-ground query.

X = finance

X = ibm-exams

1) X = electrical

   ?- faculty(_, electrical). is true. So \+ faculty(_, electrical) is false.

2) X = civil

   ?- faculty(_,civil). is true. So \+ faculty(_,civil) is false.

3) **X = finance**

   ?- faculty(_,finance). is false. So \+ faculty(_,finance) is true.

4) **X = ibm-exams**

   ?- faculty(_,ibm-exams). is false. So \+ faculty(_,ibm-exams) is true.

Q4:

1. exists(P), dateofbirth(P, date(_,_,Y)), Y<1963, salary(P, Salary), Salary<15000.

   ?- exists(P) :- husband(P); wife(P); child(P).

   ?- husband(P).

   ?- family(P, _, _).

   a) P = person(john, cohen, date(17,may,1990), unemployed)

      ?- dateofbirth(person(john, cohen, date(17,may,1990), unemployed), date(_,_,Y)).

      person(_, _, Date, _) = person(john, cohen, date(17,may,1990), unemployed).

      Date = date(17,may,1990).

      Y = 1990.

      ?- Y < 1963. Return **false**.

   b) P = person(john, armstrong, date(7,may,1988), unemployed)

      ?- dateofbirth(person(john, armstrong, date(7,may,1988), unemployed), date(_,_,Y)).

      person(_, _, Date, _) = person(john, armstrong, date(7,may,1988), unemployed).

      Date = date(7,may,1988).

      date(7,may,1988) = date(_,_,Y)

      Y = 1988.

      ?- Y < 1963. Return **false**.

   c) P = person(eric, baily, date(7,may,1963), works(bbc, 2200)).

      ?- dateofbirth(person(eric, baily, date(7,may,1963), works(bbc, 2200)), date(_,_,Y)).

person(_, _, Date, _) = person(eric, baily, date(7,may,1963), works(bbc, 2200))

Date = date(7,may,1963)

date(7,may, 1963) = date(_,_,Y)

Y = 1963

?- Y < 1963. Return **false**.

d) P = person(eric, baily, date(7,may,1963), works(acc, 21200))

?- dateofbirth(person(eric, baily, date(7,may,1963), works(acc, 21200))

, date(_,_,Y)).

person(_, _, Date, _) = person(eric, baily, date(7,may,1963), works(acc, 21200)).

Date = date(7,may,1963).

date(7,may,1963) = date(_,_,Y)

Y = 1963

?- Y < 1963. Return **false**.

e) P = person(eric, fox, date(27,may,1970), works(bbc, 25200)),

?- dateofbirth(person(eric, fox, date(27,may,1970), works(bbc, 25200)), date(_,_,Y))

person(_, _, Date, _) = person(eric, fox, date(27,may,1970), works(bbc, 25200))

Date = date(27,may,1970),

date(27,may,1970) = date(_,_,Y)

Y = 1970

?- Y < 1963. Return **false**.

f) P = person(tom, cohen, date(7,may,1960), works(bcd, 15200)).

?- dateofbirth(person(tom, cohen, date(7,may,1960), works(bcd, 15200)), date(_,_,Y))

person(_, _, Date, _) = person(tom, cohen, date(7,may,1960), works(bcd, 15200))

Date = date(7,may,1960)

date(7,may,1960)= date(_,_,Y)

Y = 1960

?- Y < 1963. True.

?- salary(person(tom, cohen, date(7,may,1960), works(bcd, 15200)), Salary).

salary(person(_, _, _, works(_, S)), S) = salary(person(tom, cohen, date(7,may,1960), works(bcd, 15200)), Salary)

S = 15200

Salary = 15200

?- Salary<15000. Return **false**.

g) P = person(bob, armstrong, date(12,oct,1977), works(ntnu, 12000))

?- dateofbirth(person(bob, armstrong, date(12,oct,1977), works(ntnu, 12000)), date(_,_,Y))

person(_, _, Date, _) = person(bob, armstrong, date(12,oct,1977), works(ntnu, 12000))

Date = date(12,oct,1977)

date(12,oct,1977) = date(_,_,Y)

Y = 1977

?- Y < 1963.

Return **false**.

h) P = person(tony, oliver, date(7,may,1960), works(bbc, 35200))

?- dateofbirth(person(tony, oliver, date(7,may,1960), works(bbc, 35200)), date(_,_,Y))

person(_, _, Date, _) = person(tony, oliver, date(7,may,1960), works(bbc, 35200))

Date = date(7,may,1960)

date(7,may,1960) = date(_,_,Y)

Y = 1960

?- Y < 1963. True.

?- salary(person(tony, oliver, date(7,may,1960), works(bbc, 35200)), Salary).

salary(person(_, _, _, works(_, S)), S) = salary(person(tony, oliver, date(7,may,1960), works(bbc, 35200)), Salary)

S = 35200

Salary = 35200

?- Salary<15000. Return **false**.

i) **P = person(jack, fox, date(27,may,1940), unemployed)**

?- dateofbirth(person(jack, fox, date(27,may,1940), unemployed), date(_,_,Y))

person(_, _, Date, _) = person(jack, fox, date(27,may,1940), unemployed)

Date = date(27,may,1940)

date(27,may,1940) = date(_,_,Y)

Y = 1940

?- Y < 1963. True.

?- salary(person(jack, fox, date(27,may,1940), unemployed)
, Salary).

salary(person(_, _, _, unemployed), 0) = salary(person(jack, fox, date(27,may,1940), unemployed), Salary)

Salary = 0

?- Salary<15000. Return **true**.

?- wife(P).

?- family(_, P, _).

a) P = person(lily, cohen, date(9,may,1990), unemployed)

?- dateofbirth(person(lily, cohen, date(9,may,1990), unemployed), date(_,_,Y)).

person(_, _, Date, _) = person(lily, cohen, date(9,may,1990), unemployed)

Date = date(9,may,1990)

date(9,may,1990) = date(_,_,Y)

Y = 1990

?- Y < 1963. Return **false**.

b) **P = person(lily, armstrong, date(29,may,1961), unemployed)**

?- dateofbirth(person(lily, armstrong, date(29,may,1961), unemployed), date(_,_,Y))

person(_, _, Date, _) = person(lily, armstrong, date(29,may,1961), unemployed)

Date = date(29,may,1961)

date(29,may,1961) = date(_,_,Y)

Y = 1961

?- Y < 1963. True.

?- salary(person(lily, armstrong, date(29,may,1961), unemployed), Salary).

salary(person(_, _, _, unemployed), 0) = salary(person(jack, fox, date(27,may,1940), unemployed), Salary)

Salary = 0

?- Salary<15000. **Return true**.

c) P = person(grace, baily, date(9,may,1965), works(ntu, 1000))

?- dateofbirth(person(grace, baily, date(9,may,1965), works(ntu, 1000)), date(_,_,Y))

person(_, _, Date, _) = person(grace, baily, date(9,may,1965), works(ntu, 1000))

Date = date(9,may,1965)

date(9,may,1965) = date(_,_,Y)

Y = 1965

?- Y < 1963. Return **false**.

d) P = person(grace, baily, date(9,may,1965), works(ntnu, 12000))

?- dateofbirth(person(grace, baily, date(9,may,1965), works(ntu, 1000)), date(_,_,Y))

person(_, _, Date, _) = person(grace, baily, date(9,may,1965), works(ntu, 1000))

Date = date(9,may,1965)

date(9,may,1965) = date(_,_,Y)

Y = 1965

?- Y < 1963. Return false.

e) P = person(grace, fox, date(9,may,1971), works(ntbu, 13000))

?- dateofbirth(person(grace, fox, date(9,may,1971), works(ntbu, 13000)), date(_,_,Y))

person(_, _, Date, _) = person(grace, fox, date(9,may,1971), works(ntbu, 13000))

Date = date(9,may,1971)

date(9,may,1971) = date(_,_,Y)

Y = 1971

?- Y < 1963. Return **false**.

f) P = person(ann, cohen, date(29,may,1961), unemployed)

?- dateofbirth(person(ann, cohen, date(29,may,1961), unemployed), date(_,_,Y))

person(_, _, Date, _) = person(ann, cohen, date(29,may,1961), unemployed)

Date = date(29,may,1961)

date(29,may,1961) = date(_,_,Y)

Y = 1961

?- Y < 1963. True.

?- salary(person(ann, cohen, date(29,may,1961), unemployed) , Salary).

salary(person(_, _, _, unemployed), 0) = salary(person(jack, fox, date(27,may,1940), unemployed), Salary)

Salary = 0

?- Salary<15000. **Return true.**

g) P = person(liz,armstrong, date(6,oct,1975), unemployed)

?- dateofbirth(person(liz,armstrong, date(6,oct,1975), unemployed), date(_,_,Y))

person(_, _, Date, _) = person(liz,armstrong, date(6,oct,1975), unemployed)

Date = date(6,oct,1975)

date(6,oct,1975) = date(_,_,Y)

Y = 1975

?- Y < 1963

Return **false**.

h) P = person(anny, oliver, date(9,may,1961), unemployed)

?- dateofbirth(person(anny, oliver, date(9,may,1961), unemployed), date(_,_,Y))

person(_, _, Date, _) = person(anny, oliver, date(9,may,1961), unemployed)

Date = date(9,may,1961)

date(9,may,1961) = date(_,_,Y)

Y = 1961

?- Y < 1963. True.

?- salary(person(anny, oliver, date(9,may,1961), unemployed) , Salary).

salary(person(_, _, _, unemployed), 0) = salary(person(anny, oliver, date(9,may,1961), Salary)

Salary = 0

?- Salary<15000. Return **true**.

i) P = person(jane, fox, date(9,aug,1941), works(ntu, 13050))

?- dateofbirth(person(jane, fox, date(9,aug,1941), works(ntu, 13050)), date(_,_,Y))

person(_, _, Date, _) = person(jane, fox, date(9,aug,1941), works(ntu, 13050))

Date = date(9,aug,1941)

date(9,aug,1941) = date(_,_,Y)

Y = 1941

?- Y < 1963. True.

?- salary(person(jane, fox, date(9,aug,1941), works(ntu, 13050)) , Salary).

salary(person(_, _, _, works(_, S)), S) = salary(person(jane, fox, date(9,aug,1941), works(ntu, 13050)), Salary)

Salary = 13050

?- Salary<15000. Return **true**.

?- child(P)

?- family(_, _, P), member(X, P).

Because all the children's year of birth are larger than 1963, we omit the tracking here.

So, the results are

P = person(jack, fox, date(27, may, 1940), unemployed),

Salary = 0,

Y = 1940

P = person(lily, armstrong, date(29, may, 1961), unemployed),

Salary = 0,

Y = 1961

P = person(ann, cohen, date(29, may, 1961), unemployed),

Salary = 0,

Y = 1961

P = person(anny, oliver, date(9, may, 1961), unemployed),

Salary = 0,

Y = 1961

P = person(jane, fox, date(9, aug, 1941), works(ntu, 13050)),

Salary = 13050,

Y = 1941

false

2. exists(P), dateofbirth(P,date(_,_,Y)), !, Y<1998, salary(P,Salary), Salary<20000.

   ?- exists(P) :- husband(P); wife(P); child(P).

   ?- husband(P).

   ?- family(P, _, _).

   a) **P = person(john, cohen, date(17,may,1990), unemployed)**

      ?- dateofbirth(person(john, cohen, date(17,may,1990), unemployed),date(_,_,Y)).

      person(_, _, Date, _) = person(john, cohen, date(17,may,1990), unemployed)

      Date = date(17,may,1990)

      date(17,may,1990) = date(_,_,Y)

      Y = 1990

      Cutting here!

      ?- Y < 1998. True.

      ?- salary(person(john, cohen, date(17,may,1990), unemployed)
      , Salary).

      salary(person(_, _, _, unemployed), 0) = salary(person(john, cohen, date(17,may,1990), unemployed), Salary)

      Salary = 0

      ?- Salary<20000. Return **true**.

So, the result is

P = person(john, cohen, date(17, may, 1990), unemployed),

Salary = 0,

Y = 1990

3. wife(person(GivenName, FamilyName, _, works(_,_))).

   ?- family(_, person(GivenName, FamilyName, _, works(_,_)), _).

   The following person can be unified:

   a) person(grace, baily, date(9,may,1965), works(ntu, 1000))

FamilyName = baily,

GivenName = grace

b) person(grace, baily, date(9,may,1965), works(ntnu, 12000))

FamilyName = baily,

GivenName = grace

c) person(grace, fox, date(9,may,1971), works(ntbu, 13000)).

FamilyName = fox,

GivenName = grace

d) person(jane, fox, date(9,aug,1941), works(ntu, 13050))

FamilyName = fox,

GivenName = jane

4. child(X), dateofbirth(X, date(_,_,1983)).

?- family(_, _, Children), member(X, Children).

All the child in the list can unify the first query, so we omit here and only consider the second query: dateofbirth(X, date(_,_,1983)).

The following child can unify this query:

X = person(louie, baily, date(25,may,1983), unemployed)

X = person(louie, baily, date(25,may,1983), unemployed)

X = person(pat, cohen, date(5,may,1983), works(bcd, 15200))

X = person(jim, cohen, date(5,may,1983), works(bcd, 15200))

X = person(jimey, oliver, date(5,may,1983), unemployed)

Q5:

1) total([ ],0).

total([Person|List], Sum):-

salary(Person, S),

total(List, Rest),

Sum is S + Rest.

```
tot_income(family(Husband,Wife,Children),I) :- family(Husband,Wife,Children),
total([Husband, Wife | Children], I ).
```

2)

```
?- tot_income(family(Husband,Wife,Children) , I)
```

3)

```
?-
family( Husband, Wife, Children),
total( [ Husband, Wife | Children ] , Income ),
length( [ Husband, Wife | Children ] , N ),
Income / N < 2000.
```

4)

```
?-

family(Husband,Wife,Children),

total([Husband,Wife|[]],IncomeParrents ),

total([Husband, Wife|Children], IncomeAll),

IncomeChildren is IncomeAll-IncomeParrents,

IncomeChildren > IncomeParrents.
```

Q6:

a)

```
flightPath(lax,nrt,11,5439).
flightPath(cdg,jfk,8,3624).
flightPath(cdg,lax,11,5656).
flightPath(cdg,fco,2,684).
flightPath(lju,cdg,2,587).
flightPath(lju,fco,2,628).
flightPath(jfk,lax,6,2469).
flightPath(jfk,nrt,14,6729).
flightPath(fco,jfk,10,4266).
flightPath(fco,sin,12,6245).
```

```
    flightPath(sin,nrt,7,3329).
b)  transferTime(lax,2).
    transferTime(jfk,2).
    transferTime(fco,2).
    transferTime(cdg,2).
    transferTime(lju,2).
    transferTime(sin,2).
    transferTime(nrt,2).
```

A.

```
1)  connection(Start, Destination) :- flightPath(Start, Destination,_ , _).
    connection(Start, Destination) :- flightPath(A,C,_,_), connection(C,B).
2)  flightTime(Start,Destination,Time,Path):-

    flightPath(Start,Destination,FlightTime,_),

    transferTime(Start, TimeStart),

    transferTime(Destination, TimeDest),

     Time is FlightTime + TimeStart + TimeDest,

    append([Start],[Destination],Path).


    flightTime(Start,Destination,Time,Path):-

     flightPath(Start,X,FlightTime,_),

     transferTime(Start, TimeStart1),

     flightTime(X,Destination,TimeX,PathX),

     Time is FlightTime + TimeX +TimeStart1,

     append([Start],PathX,Path).
```

```
3)  pathLength([],0).

    pathLength([_],0).

    pathLength([X,Y|L],Length):-

    flightPath(X,Y,_,T),

    pathLength([Y|L], Length1),

    Length is Length1+ T.
```

```
4)  shortestPath(Start, Destination) :-

    findall(Path, flightTime(Start, Destination, _, Path), Paths),

    findall(Length, (member(MemberOfPaths, Paths),
    pathLength(MemberOfPaths, Length)), Lengths),

    min_list(Lengths, ShortestLength),

    member(ShortestPath, Paths),

    pathLength(ShortestPath, ShortestLength),

    print(ShortestPath).
```