COMP 348 assignment 1

Name: Hualin Bai ID:40053833

Question1: Knowledge representation in prolog

```
individual(hualin, male, baishi, chunlan).
individual(huana, male, baishui, fang).
individual(tianshu, male, shixin, honghong).
individual(jane, female, jiang, bekery).
individual(bangkun, male, yinqiang, wangli).
individual(dahu, male, yinqiang, wangli).
individual(zijin, female, liuwen, yinyu).
individual(baishi, male, zjiu, qying).
individual(honghong, female, zjiu, qying).
individual(bekery, female, zjiu, qying).
individual(baishui, male, zjiu, qying).
individual(yinqiang, male, kjian, wyi).
individual(chunlan, female, kjian, wyi).
individual(yinyu, female, kjian, wyi).
individual(lulu, female, zijin, wuxian).
individual(kili, male, dahu, lida).

offspring(X,Y):- individual(X,_,Y,_); individual(X,_,_,Y).
niblings(X,Y):- offspring(X,Z),individual(Z,_,F,M), individual(Y,_,F,M), Z \= Y.
puncle(X,Y):- offspring(Y,P), individual(P,male,F,M), individual(X,male,F,M), P \= X.
modrige(X,Y):- offspring(Y,P), individual(P,female,F,M),individual(X,female,F,M),P \= X.
avuncle(X,Y):- offspring(Y,P), individual(P,female,F,M), individual(X,male,F,M).
```

offspring(X,Y):- individual(X,_,Y,_); individual(X,_,_,Y).

niblings(X,Y):- offspring(X,Z),individual(Z,_,F,M), individual(Y,_,F,M), Z \= Y.

puncle(X,Y):- offspring(Y,P), individual(P,male,F,M), individual(X,male,F,M), P \= X.

modrige(X,Y):- offspring(Y,P), individual(P,female,F,M),individual(X,female,F,M),P \= X.

avuncle(X,Y):- offspring(Y,P), individual(P,female,F,M), individual(X,male,F,M).

Question2: Unifications and resolutions in Prolog

Program

```
likes(jane,X) = likes(X, josh).
This pair cannot be unified because X is conflicted.

diSk(27, queens, sgt_pepper) = diSk(A, B, help).
This pair cannot be unified because help is not match sgt_pepper.

[a,b,c] = [X,Y,Z|T].
This pair cannot be unified because Z|T is conflicted..

ancestor(french(jean), B) = ancestor(A, irish(joe)).
This pair can be unified with A = french(jean), B = irish(joe).

characters(hero(luke), X) = characters(X, villain(vader)).
This pair cannot be unified because X is conflicted.

f(X, a(b,c)) = f(d, a(Z, c)).
This pair can be unified with X = d, Z = b.

s(x, f(x), z) = s(g(y), f(g(b)), y).
This pair cannot be unified because x,y,z are not variables and cannot match.

vertical( line(point(X,Y), point(X,Z))) = vertical(line(point(1,1),point(1,3))).
This pair can be unified with X = 1, Y = 1, Z = 3.

g(Z, f(A, 17, B), A+B, 17) = g(C, f(D, D, E), C, E).
This pair can be unified with Z=C, A=D, D=17,B=E, A+B=C,E=17.

f(c, a(b,c)) = f(Z, a(Z, c)).
This pair cannot be unified because Z is conflicted.
```

Question3: Queries in Prolog

```
1  building(engineering, ev).
2  building(business, mb).
3  building(library, lb).
4  building(classes, h).
5  building(hr, fg).
6  department(electrical, engineering).
7  department(civil, engineering).
8  department(finance, business).
9  department(ibm-exams, lb).
10 status(engineering, accredited).
11 faculty(smith, electrical).
12 faculty(walsh, electrical).
13 faculty(smith, computer).
14 faculty(jones, civil).
15 faculty(james, civil).
16 faculty(davis, civil).
17 faculty(X, Y) :- department(Z, Y), faculty(X, Z).
18 building(X, Y) :- department(X, Z), building(Z, Y).
19 status(X,Z) :- department(X,Y),status(Y,Z).
20 faculty(X) :- faculty(X,_).
```

1.? building(library,lb).

ground queries.

Respond: True.

found by matching the database directly of line3: building(library,lb).

2.? status(finance, A).

non-ground queries.

Respond: False.

(1) go down to line 19 and find status(X,Z) :- department(X,Y),status(Y,Z).

unify X = finance, Z = A.

(2) go up to line 8 and find department(finance, business),return true,and unify Y = business.

(3) go down to line 19 and find status(X,Z):-department(X,Y),status(Y,Z).

unifiy X = business, Z = A.

(4) fail to find department(business,_), return false.

(5) fail to find status(business,_), return false.

(6) fail to find status(finance,A), return false.

3.? department(civil, Bussiness).

non-ground queries.

Respond: Bussiness = engineering.

(1) go down to line 7 and find department(civil,engineering).unify Bussiness = engineering.

(2) respond Bussiness = engineering.

4.? faculty(X, civil).

non-ground queries.

Respond: X = jones;

X = james;

X = davis;

First respond:

(1) go down to line 14 and find faculty(jones, civil).unify and respond X = jones.

Second respond:

(1) go down to line 15 and find faculty(james, civil).unify and respond X = james.

Third respond:

(1) go down to line 16 and find faculty(davis, civil).unify and respond X = davis.

Fourth respond:

(1) go down to line 17 and find faculty(X, Y) :- department(Z, Y), faculty(X, Z).

unify X = X, Y = civil.

(2) fail to find department(_,civil), return false.

(3) fail to find faculty(_,civil), respond false.

5.? faculty(smith, X).

non-ground queries.

Respond: X = electrical

X = computer

X = engineering

First respond:

(1) go down to line 11 and find faculty(smith, electrical).unify and respond X = electrical.

Second respond:

(1) go down to line 13 and find faculty(smith, computer).unify and respond X = computer.

Third respond:

(1) go down to line 17 and find faculty(X, Y) :- department(Z, Y), faculty(X, Z).

unify X = smith.

(2) go up to find department(electrical, engineering), unify Z = electrical, Y = engineering.

(3) go down to find faculty(smith, electrical), unify and return Y = engineering.

(4) unify X = Y = engineering, respond X = engineering.

Fourth respond:

(1) fail to find other cases to match faculty(X, Y) :- department(Z, Y), faculty(X, Z).

(2) respond false.

6.? department(X, Y).

non-ground queries.

Respond: X = electrical,Y = engineering

X = civil,Y = engineering

X = finance,Y = business

X = ibm-exams,Y = lb

First respond:

(1) go down to line 6 and find department(electrical, engineering).unify and respond X = electrical, Y = engineering.

Second respond:

(1) go down to line 7 and find department(civil, engineering).unify and respond X = civil, Y = engineering.

Third respond:

(1) go down to line 8 and find department(finance, business).unify and respond X = finance, Y = business.

Fourth respond:

(1) go down to line 9 and find department(ibm-exams, lb).unify and respond X = ibm-exams, Y = lb.

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

non-ground queries.

Respond: X = jones, Y = engineering

X = james, Y = engineering

X = davis, Y = engineering

First respond:

(1) go down to line 14 and find faculty(jones, civil),unify X = jones.

(2) go down to line 7 and find department(civil, engineering),unify Y = engineering.

(3) respond X = jones, Y = engineering.

Second respond:

(1) go down to line 15 and find faculty(james, civil),unify X = james.

(2) go down to line 7 and find department(civil, engineering),unify Y = engineering.

(3) respond X = james, Y = engineering.

Third respond:

(1) go down to line 16 and find faculty(davis, civil),unify X = davis.

(2) go down to line 7 and find department(civil, engineering),unify Y = engineering.

(3) respond X = davis, Y = engineering.

Fourth respond:

(1) go down to line 17 and find faculty(X, Y) :- department(Z, Y), faculty(X, Z).unify Y = civil.

(2) fail to find department(_,civil), respond false.

8.? faculty(Smith).

non-ground queries.

Respond: Smith = smith

Smith = walsh

Smith = smith

Smith = jones

Smith = james

Smith = davis

Smith = smith

Smith = walsh

Smith = jones

Smith = james

Smith = davis

First respond:

(1) go down to line 20 and find faculty(X) :- faculty(X,_),unify X = Smith.

(2) go down to line 11 and find faculty(smith, electrical),unify X = smith.

(3) respond Smith = smith.

Second respond:

(1) go down to line 20 and find faculty(X) :- faculty(X,_),unify X = Smith.

(2) go down to line 12 and find faculty(walsh, electrical),unify X = walsh.

(3) respond Smith = walsh.

Third respond:

(1) go down to line 20 and find faculty(X) :- faculty(X,_),unify X = Smith.

(2) go down to line 13 and find faculty(smith, computer),unify X = smith.

(3) respond Smith = smith.

Fourth respond:

(1) go down to line 20 and find faculty(X) :- faculty(X,_),unify X = Smith.

(2) go down to line 14 and find faculty(jones, civil).,unify X = jones.

(3) respond Smith = jones.

Fifth respond:

(1) go down to line 20 and find faculty(X) :- faculty(X,_),unify X = Smith.

(2) go down to line 15 and find faculty(james, civil).,unify X = james.

(3) respond Smith = james.

Sixth respond:

(1) go down to line 20 and find faculty(X) :- faculty(X,_),unify X = Smith.

(2) go down to line 16 and find faculty(davis, civil).,unify X = davis.

(3) respond Smith = davis.

Seventh respond:

(1) go down to line 20 and find faculty(X) :- faculty(X,_),unify X = Smith.

(2) go down to line 17 and find faculty(X, Y) :- department(Z, Y), faculty(X, Z).

(3) find department(electrical, engineering).and faculty(smith, electrical).unify Z = electrical, Y = engineering, X = smith.

(4) respond Smith = smith.

Eighth respond:

(1) go down to line 20 and find faculty(X) :- faculty(X,_),unify X = Smith.

(2) go down to line 17 and find faculty(X, Y) :- department(Z, Y), faculty(X, Z).

(3) find department(electrical, engineering).and faculty(walsh, electrical).unify Z = electrical, Y = engineering, X = walsh.

(4) respond Smith = walsh.

9th respond:

(1) go down to line 20 and find faculty(X) :- faculty(X,_),unify X = Smith.

(2) go down to line 17 and find faculty(X, Y) :- department(Z, Y), faculty(X, Z).

(3) find department(civil, engineering).and faculty(jones, civil)..unify Z = civil, Y = engineering, X = jones.

(4) respond Smith = jones.

10th respond:

(1) go down to line 20 and find faculty(X) :- faculty(X,_),unify X = Smith.

(2) go down to line 17 and find faculty(X, Y) :- department(Z, Y), faculty(X, Z).

(3) find department(civil, engineering).and faculty(james, civil).unify Z = civil, Y = engineering, X = james.

(4) respond Smith = james.

11th respond:

(1) go down to line 20 and find faculty(X) :- faculty(X,_),unify X = Smith.

(2) go down to line 17 and find faculty(X, Y) :- department(Z, Y), faculty(X, Z).

(3) find department(civil, engineering).and faculty(davis, civil).unify Z = civil, Y = engineering, X = davis.

(4) respond Smith = davis.

9.? building(_, X).

non-ground queries.

Respond: X = ev

X = mb

X = lb

X = h

X = fg

X = ev

X = ev

X = mb

First respond:

(1) go down to line 1 and find building(engineering, ev).,unify X = ev.

(2) respond X = ev.

Second respond:

(1) go down to line 2 and find building(business, mb).,unify X = mb.

(2) respond X = mb.

Third respond:

(1) go down to line 3 and find building(library, lb).,unify X = lb.

(2) respond X = lb.

Fourth respond:

(1) go down to line 4 and find building(classes, h).,unify X = h.

(2) respond X = h.

Fifth respond:

(1) go down to line 5 and find building(hr, fg).,unify X = fg.

(2) respond X = fg.

Sixth respond:

(1) go down to line 17 and find building(X, Y) :- department(X, Z), building(Z, Y).,unify Y = X.

(2) find department(electrical, engineering).and building(engineering, ev)..unify Z = engineering,X = electrical, Y = ev.

(3) respond X = ev.

Seventh respond:

(1) go down to line 17 and find building(X, Y) :- department(X, Z), building(Z, Y).,unify Y = X.

(2) find department(civil, engineering).and building(engineering, ev)..unify Z = engineering,X = civil, Y = ev.

(3) respond X = ev.

Eighth respond:

(1) go down to line 17 and find building(X, Y) :- department(X, Z), building(Z, Y).,unify Y = X.

(2) find department(finance, business).and building(business, mb).unify Z = business,X = finance, Y = mb.

(3) respond X = mb.

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

non-ground queries.

Respond: X = engineering, Y = ev

X = electrical,Y = ev

X = civil,Y = ev

First respond:

(1) go down to line 10 and find status(engineering, accredited).,unify X = engineering.

(2) find building(engineering, ev). unify Y = ev.

(2) respond X = engineering, Y = ev.

Second respond:

(1) go down to line 19 and find status(X,Z) :- department(X,Y),status(Y,Z).,unify Z = accredited.

(2) find department(electrical, engineering),and status(engineering, accredited). unify Y = engineering, X = electrical.

(3) respond X = electrical.

(4) find building(X, Y) :- department(X, Z), building(Z, Y).unify X = electrical.

(5) find department(electrical, engineering), and building(engineering, ev).unify Z = engineering,X = electrical, Y = ev.

(6) respond X = electrical, Y = ev.

Third respond:

(1) go down to line 19 and find status(X,Z) :- department(X,Y),status(Y,Z).,unify Z = accredited.

(2) find department(civil, engineering).,and status(engineering, accredited). unify Y = engineering, X = civil.

(3) respond X = civil.

(4) find building(X, Y) :- department(X, Z), building(Z, Y).unify X = civil.

(5) find department(civil, engineering), and building(engineering, ev).unify Z = engineering,X = civil, Y = ev.

(6) respond X = civil, Y = ev.

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

non-ground queries.

Respond: False.

First respond:

(1) find status(engineering, accredited).unify X = accredited.

(2) fail to find building(accredited,_), respond false.

Second respond:

(1) find status(X,Z) :- department(X,Y),status(Y,Z).unify Z = X.

(2) find status(engineering, accredited).unify Z = accredited, Y = engineering.

(3) find department(electrical, engineering).unify X = electrical.

(4) fail to find building(electrical,engineering), respond false.

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

non-ground queries.

Respond: 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

First respond:

(1) find faculty(smith, electrical),unify X = smith, Y = electrical.

(2) find department(electrical, engineering).. unify Y = electrical.

(3) respond X = smith, Y = electrical.

Second respond:

(1) find faculty(walsh, electrical).,unify X = walsh, Y = electrical.

(2) find department(electrical, engineering). unify Y = electrical.

(3) respond X = walsh, Y = electrical.

Third respond:

(1) find faculty(smith, computer),unify X = smith.

(2) find faculty(smith, electrical),unify X = smith, Y = electrical.

(3) respond X = smith, Y = electrical.

Fourth respond:

(1) find faculty(jones, civil).,unify X = jones, Y = civil.

(2) find department(civil, engineering).. unify Y = civil.

(3) respond X = jones, Y = civil.

Fifth respond:

(1) find faculty(james, civil).,unify X = james, Y = civil.

(2) find department(civil, engineering).. unify Y = civil.

(3) respond X = james, Y = civil.

Sixth respond:

(1) find faculty(davis, civil).,unify X = davis, Y = civil.

(2) find department(civil, engineering).. unify Y = civil.

(3) respond X = davis, Y = civil.

7th respond:

(1) find faculty(X, Y) :- department(Z, Y), faculty(X, Z). unify Z = electrical;civil.

(2) respond

X = smith, Y = electrical

X = walsh, Y = electrical

X = jones, Y = civil

X = james, Y = civil

X = davis, Y = civil

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

non-ground queries.

Respond: X = smith, Y = electrical, Z = engineering.

(1) find faculty(smith, electrical),unify X = smith, Y = electrical.

(2) find department(electrical, engineering). unify Y = electrical.

(3) find department(electrical, engineering).unify Z = engineering.

(4) respond X = smith, Y = electrical, Z = engineering.

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

non-ground queries.

Respond: X = smith

X = smith

X = smith

(1) find faculty(smith, electrical).unify X = smith.

(2) find faculty(smith, electrical).unify X = smith.

(3) find faculty(smith, computer).unify X = smith.

(4) respond X = smith, X = smith, X = smith.

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

non-ground queries.

Respond: X = finance

X = ibm-exams

First respond:

(1) find department(finance, business).unify X = finance.

(2) fail to prove faculty(_,finance), respond true.

(3) respond X = finance.

Second respond:

(1) find department(ibm-exams, lb).unify X = ibm-exams.

(2) fail to prove faculty(_,ibm-exams), respond true.

(3) respond X = ibm-exams.

Question 4: Using Prolog for a Search Problem

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

First respond:
(1) find exists(Persons) :- husband(Persons); wife(Persons); child(Persons).unify Person = P.
(2) find husband(Persons) and person(jack, fox, date(27,may,1940), unemployed),
match Y = 1940 <1963, Salary = 0 <15000. Unified by dateofbirth(person(_, _, Date, _), Date).and salary(person(_, _, _, unemployed), 0).
(3) respond P = person(jack, fox, date(27, may, 1940), unemployed),
Salary = 0,
Y = 1940

Second respond:
(1) find exists(Persons) :- husband(Persons); wife(Persons); child(Persons).unify Person = P.
(2) find wife(Persons) and person(lily, armstrong, date(29,may,1961), unemployed),
match Y = 1961 <1963, Salary = 0 <15000. Unified by dateofbirth(person(_, _, Date, _), Date).and salary(person(_, _, _, unemployed), 0).
(3) respond P = person(lily, armstrong, date(29, may, 1961), unemployed),
Salary = 0,
Y = 1961

Third respond:

(1) find exists(Persons) :- husband(Persons); wife(Persons); child(Persons).unify Person = P.

(2) find wife(Persons) and person(ann, cohen, date(29,may,1961), unemployed),

match Y = 1961 <1963, Salary = 0 <15000. Unified by dateofbirth(person(_, _, Date, _), Date).and salary(person(_, _, _, unemployed), 0).

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

Salary = 0,

Y = 1961

Fourth respond:

(1) find exists(Persons) :- husband(Persons); wife(Persons); child(Persons).unify Person = P.

(2) find wife(Persons) and person(anny, oliver, date(9,may,1961), unemployed),

match Y = 1961 <1963, Salary = 0 <15000. Unified by dateofbirth(person(_, _, Date, _), Date).and salary(person(_, _, _, unemployed), 0).

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

Salary = 0,

Y = 1961

Fifth respond:

(1) find exists(Persons) :- husband(Persons); wife(Persons); child(Persons).unify Person = P.

(2) find wife(Persons) and person(jane, fox, date(9,aug,1941), works(ntu, 13050)),

match Y = 1941 <1963, Salary = 13050 <15000. Unified by dateofbirth(person(_, _, Date, _), Date).and salary(person(_, _, _, works(_, S)), S).

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

Salary = 13050,

Y = 1941

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

Respond:

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

Salary = 0,

Y = 1990

(1) find exists(Persons) :- husband(Persons); wife(Persons); child(Persons).unify Person = P.

(2) find husband(X) :- family(X, _, _).unify X = person(john, cohen, date(17,may,1990), unemployed), Y = 1990.

(3) match Y = 1990 < 1998,and salary(person(_, _, _, unemployed), 0).unify Salary = 0 < 20000.

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

Salary = 0,

Y = 1990

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

Resopnd: FamilyName = baily, GivenName = grace

FamilyName = baily, GivenName = grace

FamilyName = fox, GivenName = grace

FamilyName = fox, GivenName = jane

First respond:

(1) find wife(X) :- family(_, X, _). unify X = person(GivenName, FamilyName, _, works(_,_)).

(2) find person(grace, baily, date(9,may,1965), works(ntu, 1000)), unify and respond FamilyName = baily, GivenName = grace.

Second respond:

(1) find wife(X) :- family(_, X, _). unify X = person(GivenName, FamilyName, _, works(_,_)).

(2) find person(grace, baily, date(9,may,1965), works(ntnu, 12000)), unify and respond FamilyName = baily, GivenName = grace.

Third respond:

(1) find wife(X) :- family(_, X, _). unify X = person(GivenName, FamilyName, _, works(_,_)).

(2) find person(grace, fox, date(9,may,1971), works(ntbu, 13000)), unify and respond FamilyName = fox, GivenName = grace.

Forth respond:

(1) find wife(X) :- family(_, X, _). unify X = person(GivenName, FamilyName, _, works(_,_)).

(2) find person(jane, fox, date(9,aug,1941), works(ntu, 13050)), unify and respond FamilyName = fox, GivenName = jane.

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

Respond:

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)

First respond:

(1) find child(X) :- family(_, _, Children), member(X, Children).unify X = X.

(2) find dateofbirth(person(_, _, Date, _), Date).unify X = person(_, _, Date, _), Date = 1983.

(3) find [person(louie, baily, date(25,may,1983), unemployed)], unify X.

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

Second respond:

(1) find child(X) :- family(_, _, Children), member(X, Children).unify X = X.

(2) find dateofbirth(person(_, _, Date, _), Date).unify X = person(_, _, Date, _), Date = 1983.

(3) find [person(louie, baily, date(25, may, 1983), unemployed)], unify X.

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

Third respond:

(1) find child(X) :- family(_, _, Children), member(X, Children).unify X = X.

(2) find dateofbirth(person(_, _, Date, _), Date).unify X = person(_, _, Date, _), Date = 1983.

(3) find [person(pat, cohen, date(5, may, 1983), works(bcd, 15200))], unify X.

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

Forth respond:

(1) find child(X) :- family(_, _, Children), member(X, Children).unify X = X.

(2) find dateofbirth(person(_, _, Date, _), Date).unify X = person(_, _, Date, _), Date = 1983.

(3) find [person(jim, cohen, date(5, may, 1983), works(bcd, 15200))], unify X.

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

Fifth respond:

(1) find child(X) :- family(_, _, Children), member(X, Children).unify X = X.

(2) find dateofbirth(person(_, _, Date, _), Date).unify X = person(_, _, Date, _), Date = 1983.

(3) find [person(jimey, oliver, date(5, may, 1983), unemployed)], unify X.

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

Question 5: Lists and Backtracking

(1). Write a prolog rule totalIncome/2 to compute the total income of a family.

totalIncome([], 0).

totalIncome([Person|List],Sum) :- salary(Person, S), totalIncome(List, Sum1), Sum is S + Sum1.

```
totalIncome([], 0).
totalIncome([Person|List],Sum) :- salary(Person, S), totalIncome(List, Sum1), Sum is S + Sum1.
```

(2). Write a prolog query to print total income of each family.

?-family(Husband,Wife,Children), totalIncome([Husband,Wife|Children], Income).

```
?- family(Husband,Wife,Children), totalIncome([Husband,Wife|Children],
   Income).
```

(3). Write a prolog query to print family details of each family that has income per family member less than 2000.

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

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

(4). Write a prolog query to print family details of each family where children's total income is more than their parents.

?- family(Husband,Wife,Children),totalIncome([Husband,Wife],Income), totalIncome(Children,Income2),Income2 > Income.

```
family(Husband,Wife,Children),totalIncome([Husband,Wife],Income),
totalIncome(Children,Income2),Income2 > Income.

Children = [person(pat, cohen, date(5, may, 1983), works(bcd, 15200)),
person(jim, cohen, date(5, may, 1983), works(bcd, 15200))],
Husband = person(tom, cohen, date(7, may, 1960), works(bcd, 15200)),
Income = 15200,
Income2 = 30400,
Wife = person(ann, cohen, date(29, may, 1961), unemployed)
false
```

```
?- family(Husband,Wife,Children),totalIncome([Husband,Wife],Income),
   totalIncome(Children,Income2),Income2 > Income.
```

Question 6: Graphs in Prolog

(a) prolog database

Database follows: flightPath(originatingAirport,destinationAirport,flightTime(hour),distance(miles))

```
flightPath(lax,nrt,12,5439).
flightPath(cdg,lax,12,5656).
flightPath(cdg,jfk ,8,3624).
flightPath(cdg,fco, 2,684).
flightPath(lju,cdg, 2,587).
flightPath(lju,fco,1,265).
flightPath(jfk,lax,7,2469).
flightPath(fco,jfk,10,4266).
flightPath(fco,sin,12,6245).
flightPath(sin,nrt,7,3329).
flightPath(jfk,nrt,14,6729).
```

(b) transferTime/2

```
transferTime(lax, 1).
transferTime(jfk, 1).
transferTime(fco, 2).
transferTime(cdg, 1).
transferTime(lju, 2).
```

```
transferTime(sin, 2).
transferTime(nrt, 3).
```

A.

(1)

```
connection(Start,Destination):- flightPath(Start,Destination,_,_).
connection(Start,Destination):- flightPath(Start,Indirect,_,_), connection(Indirect,Destination).
```

(2)

```
flightTime(Start,Destination,Time,[Start,Destination]) :- flightPath(Start,Destination,Time,_).
flightTime(Start,Destination,Time,[Start|Path]):- flightPath(Start,Indirect,T1,_),
        flightTime(Indirect,Destination,T2,Path),transferTime(Indirect,T3),
        Time is T1 + T2 + T3.
```

(3)

```
pathLength([Start,Destination],Length):- flightTime(Start,Destination,_,[Start,Destination]),
                    flightPath(Start,Destination,_,Length).
pathLength([Start,Destination|Path],Length):- flightTime(Start,Destination,_,[Start,Destination]),
            flightPath(Start,Destination,_,L1),
            pathLength([Destination|Path],L2), Length is L1 + L2.
```

(4)

```
shortestPath(Start,Destination):-
        findall(Length,(flightTime(Start,Destination,_,Path),pathLength(Path,Length)),List),
        min_list(List,Length),
        pathLength(Path,Length),
        print(Path).
```

B.

Program

```
flightPath(lax,nrt,12,5439).
flightPath(cdg,lax,12,5656).
flightPath(cdg,jfk ,8,3624).
flightPath(cdg,fco, 2,684).
flightPath(lju,cdg, 2,587).
flightPath(lju,fco,1,265).
flightPath(jfk,lax,7,2469).
flightPath(fco,jfk,10,4266).
flightPath(fco,sin,12,6245).
flightPath(sin,nrt,7,3329).
flightPath(jfk,nrt,14,6729).
transferTime(lax, 1).
transferTime(jfk, 1).
transferTime(fco, 2).
transferTime(cdg, 1).
transferTime(lju, 2).
transferTime(sin, 2).
transferTime(nrt, 3).

connection(Start,Destination):- flightPath(Start,Destination,_,_).
connection(Start,Destination):- flightPath(Start,Indirect,_,_), connection(Indirect,Destination).

flightTime(Start,Destination,Time,[Start,Destination]) :- flightPath(Start,Destination,Time,_).
flightTime(Start,Destination,Time,[Start|Path]):- flightPath(Start,Indirect,T1,_),
                    flightTime(Indirect,Destination,T2,Path),transferTime(Indirect,T3),
                    Time is T1 + T2 + T3.

pathLength([Start,Destination],Length):- flightTime(Start,Destination,_,[Start,Destination]),
                                    flightPath(Start,Destination,_,Length).
pathLength([Start,Destination|Path],Length):- flightTime(Start,Destination,_,[Start,Destination]),
                          flightPath(Start,Destination,_,L1),
                          pathLength([Destination|Path],L2), Length is L1 + L2.

shortestPath(Start,Destination):- findall(Length,(flightTime(Start,Destination,_,Path),pathLength(Path,Length)),List),
                            min_list(List,Length),
                            pathLength(Path,Length),
                            print(Path).

```

C. write 4 queries

(1) connection(Start, Destination)

(a)test two ground quires (one with positive and one with negative answer):

```
connection(lax,nrt).
true                                                    1
Next | 10 | 100 | 1,000 | Stop
?- connection(lax,nrt).
```

```
connection(lax,fco).
false
?- connection(lax,fco).
```

(b) two non-ground quires:

```
connection(lax,Destination).
Destination = nrt
false
?- connection(lax,Destination).
```

```
connection(Start,sin).
Start = fco
Start = cdg
Start = lju
Start = liu
false
?- connection(Start,sin).
```

(2) flightTime(Start, Destination, Time, Path)

(a)test two ground quires (one with positive and one with negative answer):

```
flightTime(cdg,lax,12,[cdg,lax]).
true                                                    1
Next | 10 | 100 | 1,000 | Stop
?- flightTime(cdg,lax,12,[cdg,lax]).
flightTime(cdg,jfk,10,[cdg,jfk]).
false
?- flightTime(cdg,jfk,10,[cdg,jfk]).
```

(b) two non-ground quires:

```
flightTime(cdg,lax,Time,Path).

Path = [cdg, lax],
Time = 12
Path = [cdg, jfk, lax],
Time = 16
Path = [cdg, fco, jfk, lax],
Time = 22
false

?- flightTime(cdg,lax,Time,Path).
```

```
flightTime(fco,nrt,Time,Path).

Path = [fco, jfk, nrt],
Time = 25
Path = [fco, jfk, lax, nrt],
Time = 31
Path = [fco, sin, nrt],
Time = 21
false

?- flightTime(fco,nrt,Time,Path).
```

(3) pathLength(Path, Length)

(a)test two ground quires (one with positive and one with negative answer):

```
pathLength([cdg,lax],5656).

true                                    1

pathLength([cdg,lax],5000).

false
```

(b) two non-ground quires:

```
pathLength([cdg,lax],Length).

Length = 5656

pathLength([cdg,lax,nrt],Length).

Length = 11095
```

```
pathLength(Path,5656).
Path = [cdg, lax]
```

```
?- pathLength(Path, 5656).
```

(4) shortestPath(Start, Destination)

(a)test two ground quires (one with positive and one with negative answer):

```
shortestPath(fco,nrt).
[fco, sin, nrt]
true 1
```

```
shortestPath(cdg,lju).
false
```

(b) two non-ground quires:

```
shortestPath(cdg,Destination).
[cdg, fco]
true 1
```

```
shortestPath(Start,nrt).
[sin, nrt]
true 1
```