Q1.

individual(adam, male, tom, sandra).

individual(helen, female, tom, sandra).

individual(andrew, male, michel, eve).

individual(john, male, michel, eve).

individual(mark, male, andrew, helen).

individual(roger,male,mark,judy).

individual(jim,male,mark,judy).

individual(janis,male,mark,judy).

individual(daphne,female,peter,janis).

individual(bryan,male,roger,olivia).

individual(oliver,male,roger,olivia).

individual(lily,female,roger,olivia).

individual(noah,male,jim,sophia).

individual(harry,male,jim,sophia).

individual(emily,female,jim,sophia).

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

siblings(X,Y):-individual(X,\_,A,B),individual(Y,\_,A,B),X\=Y.

niblings(X,Y):-siblings(Y,Z),offspring(X,Z).

puncle(X,Y):-siblings(X,Z),individual(Y,\_,Z,\_),individual(X,male,\_,\_).

modrige(X,Y):-siblings(X,Z),individual(Y,\_,\_,Z),individual(X,female,\_,\_).

avuncle(X, Y):-siblings(X,Z),individual(Y,\_,\_,Z),individual(X,male,\_,\_).

Q2.

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

False, use the same variable .it should be likes(jane, X) = likes(Y, josh).

1. diSk(27, queens, sgt\_pepper) = diSk(A, B, help).

False, help is not a variable,it should be diSk(27, queens, sgt\_pepper) = diSk(A, B, C).

1. [a,b,c] = [X,Y,Z|T].

True,X=a,Y=b,Z=c,T=[].

4. ancestor(french(jean), B) = ancestor(A, irish(joe)).

True, A = french(jean),B = irish(joe)

5. characters(hero(luke), X) = characters(X, villain(vader)).

False, use the same variable .it should be characters(hero(luke), X) = characters(Y, villain(vader)).

6. f(X, a(b,c)) = f(d, a(Z, c)).

True,X=d,Z=b.

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

False, lowercase x,z,y,b is not variable

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

X=1,Y=1,Z=3

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

A = 17, B = 17, C = 34,  D = 17, E = 17, Z = 34

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

False, Z cannot both =c and=b

Q3.

? building(library, lb). .

Ground. True, because it is one of database

? status(finance, A).

Non-Ground. False .

Unify status(X, Z) :- department(X, Y), status(Y, Z). X=finance

Unify department(X, Y),X=finance,Y=business

Unify status(Y, Z),Y=business, no such Z.

? department(civil, Bussiness).

Ground. False .The database is department(civil, engineering).

? faculty(X, civil).

Non-Ground

jones faculty(jones, civil).

james faculty(james, civil).

davis faculty(davis, civil).

? faculty(smith, X).

Non-Ground

electrical faculty(smith, electrical).

computer faculty(smith, computer).

Engineering faculty(X, Y) :- department(Z, Y), faculty(X, Z).Unify faculty(X,Z) ,Z=electrical/computer unify department(Z, Y),Y=engineering.

? department(X, Y).

Non-ground,

X=electrical Y=engineering department(electrical, engineering).

X=civil Y=engineering department (civil, engineering).

X=finance Y=business department(finance, business).

X=Ibm-exams Y=lb department(ibm-exams, lb).

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

Non-ground

X=jones,Y=engineering faculty(jones, civil). department(civil, engineering).

X=james. Y=engineering faculty(james, civil). department(civil, engineering).

X=davis Y=engineering faculty(davis, civil). department(civil, engineering).

? faculty(Smith).

if it is smith Ground True Unify faculty(X) :- faculty(X, \_).

If it is Smith(variable) Non-Ground

Smith= smith

Smith= walsh

Smith=jones

Smith=james

Smith=davis

In short,Smith could be every student cause faculty(X) :- faculty(X, \_). We don’t care what is”\_”

? building(\_, X).

Non-Ground

X=ev building(engineering, ev).

X=mb building(business, mb).

X=lb building(library, lb).

X=h building(classes, h).

X=fg building(hr, fg).

X=ev building(X, Y) :- department(X, Z), building(Z, Y). Then for department(X, Z), we use data base department(electrical, engineering). X=electrical,Z=engineering. For building(Z, Y), we use database building(engineering, ev). Y=ev.

X=ev for same reason

X=mb for same reason

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

Nonground

X=engineering, status(engineering, accredited).

Y=ev, building(engineering, ev).

X=electrical, status(X, Z) :- department(X, Y), status(Y, Z).

Y=ev building(engineering, ev).

X=civil status(X, Z) :- department(X, Y), status(Y, Z).

Y=ev building(engineering, ev).

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

Non-ground

False status(engineering, accredited). X=arreciated. No suchY

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

Non-ground

X=smith,Y=electrical faculty(X) :- faculty(X, \_). faculty(smith, electrical). department(electrical, engineering).

X=walsh,Y=electrical faculty(X) :- faculty(X, \_). faculty(walsh, electrical). department(electrical, engineering).

X=smith,Y=electrical faculty(X) :- faculty(X, \_). faculty(X, Y) :- department(Z, Y), faculty(X, Z).. department(electrical, engineering).

X=jones,Y=civil faculty(X) :- faculty(X, \_). faculty(jones, civil). department(civil, engineering).

X=james,Y=civil faculty(X) :- faculty(X, \_). faculty(james, civil). department(civil, engineering).

X=davis,Y=civil faculty(X) :- faculty(X, \_). faculty(davis, civil). department(civil, engineering).

X=smith,Y=electrical faculty(X) :- faculty(X, \_). faculty(X, Y) :- department(Z, Y), faculty(X, Z).. department(electrical, engineering).

X=walsh,Y=electrical faculty(X) :- faculty(X, \_). faculty(X, Y) :- department(Z, Y), faculty(X, Z).. department(electrical, engineering).

X=jones,Y=civil faculty(X) :- faculty(X, \_). faculty(X, Y) :- department(Z, Y), faculty(X, Z).. department(civil, engineering).

X=james,Y=civil faculty(X) :- faculty(X, \_). faculty(X, Y) :- department(Z, Y), faculty(X, Z).. department(civil, engineering).

X=davis,Y=civil faculty(X) :- faculty(X, \_). faculty(X, Y) :- department(Z, Y), faculty(X, Z).. department(civil, engineering).

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

X = smith,Y = electrical,Z = engineering

faculty(X) :- faculty(X, \_). faculty(smith, electrical). department(electrical, engineering).

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

X = smith faculty(smith, electrical).

X= smith faculty(X, Y) :- department(Z, Y), faculty(X, Z).

X=smith faculty(X, Y) :- department(Z, Y), faculty(X, Z).

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

X = finance department(finance, business). And there is no such faculty(\_, finance).

X = ibm-exams department(ibm-exams, lb). And there is no such faculty(\_,ibm-exams).

Q4.

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

exists(P), Unify exists(Persons) :- husband( Persons); wife( Persons); child( Persons). To ensure this people is in database

dateofbirth(P, date(\_,\_,Y)), Y<1963.Unify dateofbirth(person(\_, \_, Date, \_), Date). P= person(\_, \_, Date, \_), date(\_,\_,Y)=Date. Y=year and Y <1963 .This step can find P whose

salary(P, Salary) ,Unify salary(person(\_, \_, \_, works(\_, S)), S). P= person(\_, \_, \_, works(\_, S)), Salary=S. and S<15000

salary(P, Salary) can also unify salary(person(\_, \_, \_, unemployed), 0). P= person(\_, \_, \_, unemployed). Salary=0 which is also <15000

Then we can find a person P whose Y in Date(\_,\_,Y) <1963 and whose S in works(\_,S) <15000 or who is unemployed.

Solution is

*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*

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

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

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