PROGRAMMING ASSIGNMENT 2: PROLOG

- ANIKET SANJIV BONDE

PROGRAMS:

Q1)

```
brother(X,Y) :- male(Y),parent(X,Z),parent(Y,Z),\+(X = Y).
sister(X,Y) :- female(Y),parent(X,Z),parent(Y,Z),\+(X = Y).
aunt(X,Y) :- parent(X,Z),sister(Z,Y).
uncle(X,Y) :- parent(X,Z),brother(Z,Y).
grandfather(X,Y) :- male(Y),parent(X,Z),parent(Z,Y).
granddaughter(X,Y) :- female(Y),parent(Y,Z),parent(Z,X).

ancestor(X,Y) :- parent(X,Y).
ancestor(X,Y) :- parent(X,Z),ancestor(Z,Y).

descendant(X,Y) :- ancestor(Y,X).

related(X,Y) :- brother(X,Y).
related(X,Y) :- sister(X,Y).
related(X,Y) :- ancestor(X,Z),related(Z,Y).
%related(X,Y) :- descendant(X,Z),related(Z,Y).
unrelated(X,Y) :- \+related(X,Y).

parent(bart,homer).
parent(bart,marge).
parent(lisa,homer).
parent(lisa,marge).
parent(maggie,homer).
parent(maggie,marge).
parent(homer,abraham).
parent(herb,abraham).
parent(tod,ned).
parent(rod,ned).
parent(marge,jackie).
parent(patty,jackie).
parent(selma,jackie).
```

```
female(maggie).
female(lisa).
female(marge).
female(patty).
female(selma).
female(jackie).
male(bart).
male(homer).
male(herb).
male(burns).
male(smithers).
male(tod).
male(rod).
male(ned).
male(abraham).
```

Q2)

```
job(X,surgeon) :- occupation(X,oral_surgeon).
job(X,surgeon) :- occupation(X,plastic_surgeon).
job(X,surgeon) :- occupation(X,heart_surgeon).
job(X,surgeon) :- occupation(X,brain_surgeon).

state(X,texas) :- address(X,houston).
state(X,texas) :- address(X,dallas).
state(X,texas) :- address(X,college_station).
state(X,texas) :- address(X,san_antonio).

occupation(joe,oral_surgeon).
occupation(sam,patent_laywer).
occupation(bill,trial_laywer).
occupation(cindy,investment_banker).
occupation(joan,civil_laywer).
occupation(len,plastic_surgeon).
occupation(lance,heart_surgeon).
occupation(frank,brain_surgeon).
occupation(charlie,plastic_surgeon).
occupation(lisa,oral_surgeon).
address(joe,houston).
address(sam,pittsburgh).
address(bill,dallas).
address(cindy,omaha).
address(joan,chicago).
address(len,college_station).
address(lance,los_angeles).
address(frank,dallas).
address(charlie,houston).
address(lisa,san_antonio).
salary(joe,50000).
salary(sam,150000).
salary(bill,200000).
salary(cindy,140000).
salary(joan,80000).
salary(len,70000).
salary(lance,650000).
```

```
salary(frank,85000).
salary(charlie,120000).
salary(lisa,190000).

%to get the query result query result(X).
result(X) :- job(X,surgeon),state(X,texas),salary(X,A),A>100000.
```

Q3)

```
remdups([],[]).
remdups([H | T], List) :- member(H, T), remdups( T, List).
remdups([H | T], [H|T1]) :- \+member(H, T), remdups( T, T1).
```

Q4)

```
divisible(N,X) :- M is N mod X, M=0.

factor(N, M) :- numlist(2, N, L),sieve(L, N, M).
sieve([X|[]],X,[X]).
sieve([X|Y], N, [X|M]) :- divisible(N, X), Q is div(N,X), numlist(2, Q, L), sieve(L,Q,M).
sieve([X|Y], N, M) :- \+divisible(N, X), sieve(Y,N,M).
```

Q5)

```
bit(0).
bit(1).
bitvec(1,[X]) :- bit(X).
bitvec(N, [X|Y]) :- N>1,M is N-1,bitvec(M, Y), bit(X).

count([1], 1).
count([0], 0).
count([X|Y], D) :- X=:=1, count(Y, C), D is C+1.
count([X|Y], C) :- X=:=0, count(Y, C).
code(N, M, X) :- bitvec(N, X), count(X, C), C =:= M.
```

Q6)

```
sin_zero(X,X) :- abs(sin(X)) < 0.0001.
sin_zero(X,Y) :- abs(sin(X)) >= 0.0001, Z is (X - sin(X)/cos(X)), sin_zero(Z,Y).
```

Q7) NOTE: THIS QUESTION TAKES 2 MINUTES TO GET RESULT AFTER QUERY

```
solution(L) :- digit(S), S>0, digit(E), digit(N), digit(D),
                                digit(M), M>0, digit(O), digit(R), digit(Y),
                                L = [S,E,N,D,M,O,R,Y], diff(L),
                                1000*S+100*E+10*N+D+1000*M+100*O+10*R+E =:=
                                10000*M+1000*O+100*N+10*E+Y.

digit(0).
digit(1).
digit(2).
digit(3).
digit(4).
digit(5).
digit(6).
digit(7).
digit(8).
digit(9).

diff([]).
diff([X|R]) :- not(member(X,R)), diff(R).
```

Q8)

```
row(1).
row(2).
row(3).
col(1).
col(2).
col(3).
symbol(x).
symbol(o).

twoInRow(S,R) :- symbol(S), row(R), col(C1), col(C2), C1 < C2, p(S,R,C1), p(S,R,C2).
twoInCol(S,C) :- symbol(S), col(C), row(R1), row(R2), R1 < R2, p(S,R1,C), p(S,R2,C).
twoInDiagLeft(S) :- symbol(S), row(R1), row(R2), R1 < R2, p(S,R1,R1), p(S,R2,R2).
twoInDiagRight(S) :- symbol(S), row(R1), row(R2), R1 < R2, p(S,R1,4-R1), p(S,R2,4-R2).

canWin(S,R,C) :- symbol(S), row(R), col(C), twoInRow(S,R), \+p(x,R,C), \+p(o,R,C).
canWin(S,R,C) :- symbol(S), row(R), col(C), twoInCol(S,C), \+p(x,R,C), \+p(o,R,C).
canWin(S,R,R) :- symbol(S), row(R), twoInDiagLeft(S), \+p(x,R,R), \+p(o,R,R).
canWin(S,R,C) :- symbol(S), row(R), col(C), twoInDiagRight(S), \+p(x,R,C), \+p(o,R,C), C is 4-R.

forcedMove(x,R,C) :- row(R), col(C), canWin(o,R,C), write('move to block opponent!').
forcedMove(o,R,C) :- row(R), col(C), canWin(x,R,C), write('move to block opponent!').

move(x,R,C) :- row(R), col(C), canWin(x,R,C), write('go for win!').
move(o,R,C) :- row(R), col(C), canWin(o,R,C), write('go for win!').

move(x,R,C) :- row(R), col(C), \+canWin(x,R,C), forcedMove(x,R,C).
move(o,R,C) :- row(R), col(C), \+canWin(o,R,C), forcedMove(o,R,C).

p(x,1,1).
p(x,1,3).
p(o,3,1).
p(o,1,3).
```

Q9)

```
row(1).
row(2).
row(3).
row(4).
col(1).
col(2).
col(3).
col(4).

candidate(X,Y) :- col(X), row(Y), not(visited(X,Y)), Z is X-1, visited(Z,Y), col(Z).
candidate(X,Y) :- col(X), row(Y), not(visited(X,Y)), Z is X+1, visited(Z,Y), col(Z).
candidate(X,Y) :- col(X), row(Y), not(visited(X,Y)), Z is Y-1, visited(X,Z), row(Z).
candidate(X,Y) :- col(X), row(Y), not(visited(X,Y)), Z is Y+1, visited(X,Z), row(Z).

adjacent(X,Y,P,Y) :- col(X), row(Y), P is X-1, col(P).
adjacent(X,Y,P,Y) :- col(X), row(Y), P is X+1, col(P).
adjacent(X,Y,X,Q) :- col(X), row(Y), Q is Y-1, row(Q).
adjacent(X,Y,X,Q) :- col(X), row(Y), Q is Y+1, row(Q).

not_wompus(X,Y) :- adjacent(X,Y,P,Q), visited(P,Q), not(stench(P,Q)).
not_pit(X,Y) :- adjacent(X,Y,P,Q), visited(P,Q), not(breeze(P,Q)).

move(X,Y) :- candidate(X,Y), not_wompus(X,Y), not_pit(X,Y).

visited(1,1).
visited(2,1).
visited(1,2).
stench(2,1).
breeze(1,2).
```