CSCE 420 - Fall 2016
Programing Assignment #5
due: Tues, Dec 6, 2016 (by start of class, 11:10am; using Turnin on CSNet)

This project will give you some practice writing Prolog programs.
On the departmental unix servers (like compute.cs.tamu.edu),
you should able to access installations of SWI
Prolog via the command 'swipl', or you can download and install a free version
for yourself on your own machine, such as http://www.swi-prolog.org/.

PROLOG

1. Write rules in Prolog to define the following kinship relationships
in terms of basic predicates like parent(X,Y) and female(X) and male(Y).

Input the following facts about family relationships in Prolog.

```
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).

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).
```

Use your rules to answer the following queries:

```
brother(lisa,bart).
brother(rod,tod). % but not bart
```

```
sister(marge,patty).
sister(bart,X). % answers should be lisa and maggie
aunt(lisa,patty).
uncle(bart,X). % should be herb
grandfather(maggie,abraham).
granddaughter(jackie,lisa).
ancestor(jackie,bart).
relative(lisa,selma).
unrelated(bart,smithers).
unrelated(tod,bart).
```

2. Write a prolog program to compute the day of the week for any given date (in the Gregorian calendar - don't forget leap years!).

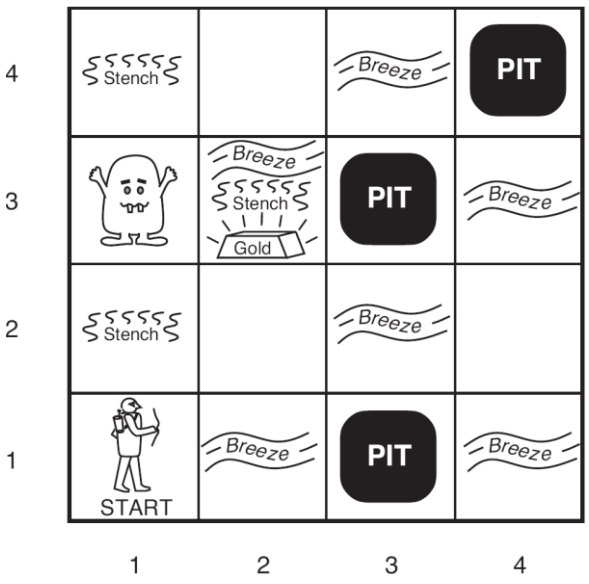
```
| ?- day_of_week(2016,10,31,D). % halloween
D = mon

| ?- day_of_week(2016,12,6,D). % due date of project
D = tues

| ?- day_of_week(2000,3,10,D). % peak of dot-com bubble
D = fri

| ?- day_of_week(1980,1,20,D). % superbowl XIV
D = sun
```

Hint: One way to solve this is to determine the number of days to a reference date. For example, Jan 1, 2016 was a Friday. Once you know the number of days to the reference date (positive or negative), it is easy to figure out the days of the week by calculating mod 7.

3. Write rules in Prolog to determine the best move in Tic-Tac-Toe for any given board configuration. Assume that the position of pieces is given by a predicate 'p'.

```
x  .  x
.  .  .
o  .  o

% assert these as facts
p(x,1,1).
p(x,1,3).
p(o,3,1).
p(o,1,3).

?- move(x,R,C).
go for win!
R = 1, C = 2 ;

x  .  .
.  .  x
o  .  o

p(x,1,1).
p(x,2,3).
p(o,1,3).
p(o,3,3).

?- move(x,R,C).
move to block opponent!
R = 3, C = 2 ;

?- move(o,R,C).
go for win!
R = 3, C = 2 ;
```

4. Write decision-making rules in Prolog for the Wumpus World. Assume the world is described in terms of facts about sense inputs (*stench* and *breeze*), along with a predicate called *visited*. Regardless of the current location of the agent, the available actions are to go to any room adjacent to one that has been visited. Primarily, the agent would prefer to explore rooms it can infer to be safe. (also assume that a wumpus, pit, and gold will never be in the same room together). There might be more than one room with gold or pits, but only one wumpus (which doesn't move). Note that your rules should work for any 4x4 cave, regardless of the location of the pits and wumpus and gold (don't just assume they are like below).




```
coordinate system:(X,Y) = (column,row)
```

example scenario.

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

?- candidate(X,Y). % just unvisited rooms adjacent to visited ones
X = 1, Y = 3 ;
X = 2, Y = 2 ;
X = 3, Y = 1 ;
?- move(X,Y). % the best choice, because it can be inferred to be safe
X = 2, Y = 2 ;
```

(Note, your implementation might give solutions in different order, or the same solution multiple times; that is OK).

Now suppose we go to (2,2) and observe no stench or breeze, but glitter.

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

?- candidate(X,Y).
X = 1, Y = 3 ;
X = 3, Y = 1 ;
X = 2, Y = 3 ;
X = 3, Y = 2 ;
?- move(X,Y).
X = 3, Y = 2 ; % because it's safe
```

(2,3) might also be a solution too, but inferring one is sufficient

5. Write Prolog rules to define the pickup() and puton() operators for the Blocksworld using Situation Calculus. You will have to include frame axioms. Note that all blocks are identical, the gripper can only pick up one block at a time (from the top of a stack), and the table is always clear (can put any number of blocks on the table).

An initial state will be described by a set of facts, such as follows.

```
   a
   b    d
   c    e
---------------
```

```
block(a).
block(b).
block(c).
block(d).
block(e).
on(a,b,init).
on(b,c,init).
on(c,table,init).
on(d,e,init).
on(e,table,init).
clear(a,init).
clear(d,init).
ge(init).
clear(table,init).
```

A sequence of actions will be represented in a nested way using *result* functions (abbreviated 'res') from Situation Calculus. For example, the state achieved by picking up d and putting it on the table will be represented as a situation argument like this: 'res(puton(d,table),res(pickup(d,e),init)).

Use your rules to make inferences about sequences of actions are possible, and what facts would be true after a given sequence of actions. For example, after picking up a and putting it on table, is it possible to pickup b? (the '_' acts like an unnamed variable that will bind to c).

```
?- poss(pickup(b,_),res(puton(a,table),res(pickup(a,b),init))).

Yes
```

Is a on the table as a result?

```
?- on(a,table,res(puton(a,table),res(pickup(a,b),init))).

Yes
```

What things would be clear after those 2 actions?

```
?- clear(Z,res(puton(a,table),res(pickup(a,b),init))).
Z = a ;
Z = b ;
Z = d ;
Z = table ;
No
```

If I then pickup d and put it on c, would I be able to put d on e?

```
?- poss(puton(d,e),res(puton(d,b),res(pickup(d,c),res(puton(a,table),
res(pickup(a,b),init))))).

No
```

After the 4 steps, list what each block is on.

```
?- on(P,Q,res(puton(d,b),res(pickup(d,e),
                            res(puton(a,table),res(pickup(a,b),init))))).
P = d, Q = b ;
P = a, Q = table ;
P = b, Q = c ;
P = c, Q = table ;
P = e, Q = table ;
```

What to Turnin

1. Submit your Prolog **source code** via the Turnin program on CSNet.
2. Also include a **transcript** showing your solutions to the problems above.