

COP4020 Spring 2011 Homework Assignment 4

1. Consider the following tic-tac-toe board positions:

X	O	O
	X	X
O		

That is, the program database holds the following facts (in this order):

x(1) .

x(5) .

x(6) .

o(2) .

o(3) .

o(7) .

Show a trace of the execution of sub-goals to solve the goal **'move(X) .'**:

Use the tic-tac-toe program shown in the lecture notes, not the one in the textbook, which is slightly different. You can download the program from:

<http://www.cs.fsu.edu/~engelen/courses/COP4020/tictactoe.gz>

Change the source so that the X and O are set as shown in the figure. On linprog, start Prolog with **'pl'** (or **'swipl'** or **'xpce'** when available) and load the program with **'[tictactoe] .'** (don't forget the period to terminate the command). Type **'move(X) .'** to query the system, but notice that the starting board positions are different in this demo. Use **'listing .'** to show the program database. Use **'trace .'** to activate the tracer before you enter a goal (use ? for help and type ENTER to creep through the trace). The tracer will help you with this assignment, but in addition you have to write down the trace depth (indentation), show failures, and redos.

2. Now change the position `o(7)` into `o(9)` in the program, reload the program, and trace `'move(X) .'`. You will notice it takes more steps to find the winning spot for the X. How many steps are needed to find the move compared to the previous case?

3. Let's change the program to see if we can speed it up. Change `move` into:

```
move(A) :- empty(A), good(A), !.
```

Trace `'move(X) .'`. What happens? Explain why this does not work. i.e. what is required for `'empty(A)'` to use as a test?

4. Fix this problem so that the modified `'move'` clause above works. Hint: you need to change the definition of `'empty'` so that it tries `'A'` for each grid cell we want to check.

5. Consider the Prolog program:

```
takes(jane_doe, his201).  
takes(jane_doe, cs254).  
takes(ajit_chandra, art302).  
takes(ajit_chandra, cs254).  
classmates(X, Y) :- takes(X, Z), takes(Y, Z).
```

The query

```
?- classmates(jane_doe, X).
```

will succeed three times: twice with `X = jane_doe` and once with `X = ajit_chandra`. Modify the `classmates(X, Y)` rule so that a student is not considered a classmate of him or herself.

6. Write a Prolog predicate that takes a list of numbers and returns the list of squares of only those numbers that are positive (ignoring the non-positive values):

```
squares([], []).
```

```
squares([X|Xs], [Y|Ys]) :- (check if X is positive, Y=X2, compute Ys)
```

```
squares([X|Xs], Ys) :- (check if X is non-positive, compute Ys)
```

Complete the above. Is this predicate relational, i.e. can in/output parameters reversed?