For all assignments submitted via Blackboard:

I. **Put your name and both usernames (FSU/Blackboard and CS) on the paper.**
II. **Write the question/problem verbatim in your paper, then give your answer/solution.**
III. **After completing the paper, save as a pdf file.**
IV. **Submit the pdf file to Blackboard using the Assignments tab.**
V. **There is no need to name the file anything other than "hw?.pdf" [? = assignment number], the submission will be associated with your username automatically.**

COP4020 Homework Assignment 3 Prolog

1. 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`**. [To see subsequent hits, enter '`;`' until the pl command prompt is displayed.] Show how to modify the **classmates(X, Y)** rule so that a student is not considered a classmate of him or herself.

For the next questions, use the tic-tac-toe program shown in the lecture notes, not the one in the textbook, which is slightly different. You can copy the program using the command:

```
cp ~cop4020p/fall11/examples/tictactoe.pl .
```

2. 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)
```

On linprog, start Prolog with '`pl`' and load the program with '`[tictactoe].`' Enter '`move(X).`' to query the system. Use '`listing.`' to show the program database. Use '`trace.`' to activate the tracer before you enter a goal (use ? for help and hit ENTER to step (or "creep") through the trace).

Modify `tictactoe.pl` to represent the state illustrated above reload the program. Show a trace of the execution of sub-goals to solve the goal '`move(X).`'. The tracer will help you with this assignment, but in addition you are required to record (1) the trace depth (using indentation), (2) show failures, and (3) show redos.

3. Now change the position **o(7)** into **o(9)** in the program and trace '`move(X).`'.

   You will notice it takes more steps to find the winning spot for the `X`. Let's change the program to see if we can speed it up. Try changing the **move** clause definition into:

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

   Try '`move(X).`'. What happens and why?

4. Fix the program so that the new '**move**' clause works. Hint: you might want to change the definition of '**empty**' so that it instantiates the variable to the position of an empty spot.