EECS 649: PROBLEM SET #6

Reading:

- R&N 7.1-6; 7.7.1, 7.7.3

Total Points: 100

Format:

- Use standard sheets of paper (8.5 by 11 inches) for scanning
- Perform all work neatly. When asked to write a paragraph, it should be typed (e.g., using a computer and LaTeX or Word).

Notes:

- **There is no need to typeset any of your answers for PS#6**
- **Problem 6.6 only requires minimax and alpha-beta**
- **Problems 6.1 through 6.4 require only material from the lecture "Knowledge Representation and Predicate Logic"**
- **Problem 6.5 requires lecture on "Automated Reasoning and Logical Agents"**

Problem 6.1 [10 points] *Wumpus World*
Suppose the Wumpus Wolrd agent has progressed to the point shown in Figure 7.4(a), having perceived nothing in [1,1], a breeze in [2,1], and a stench in [1,2], and is now concerned with the contents of [1,3], [2,2], and [3,1]. Each of these can contain a pit, and at most one can contain a wumpus. Following the example of Figure 7.5, construct the set of possible worlds. (You should find 32 of them.) Mark the worlds in which the KB is true and those in which each of the following sentences is true:
α_2 = “There is no pit in [2,2].”
α_3 = “There is a wumpus in [1,3].”
Hence show that KB |= α_2 and KB |= α_3.

If you like, you can use these linked grids, kindly provided by former student Joseph Zarycki.
Note that the Wumpus can be in the same square as a pit; also, remember that there may be more than one pit.

Problem 6.2 [10 points] *Four Propositions*
Consider a vocabulary with just four propositions: A, B, C, D. How many models are there for each of the following sentences?
a. $B \vee C$.
b. $\neg A \vee \neg B \vee \neg C \vee \neg D$.
c. $(A \Rightarrow B) \wedge A \wedge \neg B \wedge C \wedge D$.
Find the number of models for which each sentence <u>is true.</u>

Problem 6.3 [10 points] *Verifying Equivalences*
Using a method of your choice (e.g., a truth table), verify the equivalences for (a) contraposition and (b,c) De Morgan's two laws in Figure 7.11

Problem 6.4 [15 points] *Valid, Unsatisfiable, or Neither*
Decide whether each of the following sentences is valid, unsatisfiable, or neither. Verify your decisions using truth tables or the equivalence rules of Figure 7.11.
a. Smoke ⇒ Smoke
b. Smoke ⇒ Fire
c. (Smoke ⇒ Fire) ⇒ (¬Smoke ⇒ ¬Fire)
d. Smoke ∨ Fire ∨ ¬Fire
e. ((Smoke ∧ Heat) ⇒ Fire) ⇔ ((Smoke ⇒ Fire) ∨ (Heat ⇒ Fire))
f. (Smoke ⇒ Fire) ⇒ ((Smoke ∧ Heat) ⇒ Fire)

Problem 6.5 [20 points] *Forward Chaining and DPLL*
Trace the behavior of DPLL on the knowledge base in Figure 7.16 when trying to prove Q, and compare this behavior with that of the forward-chaining algorithm.

Hints:

1. **We did forward chaining (and backward chaining) in class;**
2. **For DPLL, put KB in CNF -- see the caption of Figure 7.12 for the pattern: A ^ B => C becomes ~A v ~B v C;**
3. **Wikipedia's DPPL entry may be easier to understand than R&N's description:**
   https://en.wikipedia.org/wiki/DPLL_algorithm#The_algorithm
   There is also an example (using ' for negation and + for or).

The last problem requires programming to solve a non-trivial game: 4x4 Tic-Tac-Toe.
Specifically, it will involve modifying a supplied Python program.
Hence, the problem is easiest if you use Python.
It is especially straightforward if you use Google Colaboratory, as explained next.
To use Google Colaboratory, you need a Google account (create if you don't have one).
Then:

- https://drive.google.com THEN NAVIGATE +New, More >, Google Colaboratory (sometimes you have to add this as an extension if not available)
- MORE INFO AT https://colab.research.google.com/notebooks/welcome.ipynb

Problem 6.6 [35 points] *Solving games in practice*
In this linked Tic-Tac-Toe Python notebook, we provide programs for both minimax and alpha-beta search to solve Tic-Tac-Toe (on a 3x3 board).

a. Start with the program given or translate it to another language of your choice. Test it on the examples from the lecture notes to see if you get the same results. **Do not turn anything in.**

b. Copy the program into a file called TicTacToe4. Modify it to play 4x4 Tic-Tac-Toe (where you win with 4 marks across any row, column, or main diagonals). For ease in debugging and reporting, modify the printing function as well. For reference, this took me 10-15 minutes, noting patterns and applying cut-and-paste liberally. Note that there are a couple tricky places that I have marked with "**`BEWARE`**".

   Do not turn in any code. But comment at a high level on what you modified, how long it took, etc.

c. Run minimax on the ten test boards at the end of this problem set (as printed by my modified program), **as your available computer memory and time allow**.

   Notes:

   - The boards are arranged in order of time/space complexity and all **may not be solvable** in reasonable time/memory on your (or Google's) machines. Hence, you **do not need to solve every board to receive full credit**. But comment.
   - You may need or want to cutoff search at some point, say at 100 million nodes (YMMV) expanded for time or if the `seen_set` gets too large (or maxes out your RAM). This can be done with if statements in the `new_node` function.
   - To check your program: note that Board 1 is an immediate win, Board 2 is an immediate loss; Boards 3-6 are wins.

   Report your results (please delete any intermediate results printed!), including the games value, optimal move (minimax only), nodes examined, terminals examined, unique states encountered, and time taken.

d. Repeat the above using alpha-beta.

e. **Compare** your minimax and alpha-beta results. How much more efficient was it in terms of nodes examined (etc.) and time? **Did you weakly solve** 4x4 Tic-Tac-Toe? What is the **full game's value** for the first player ('X')?

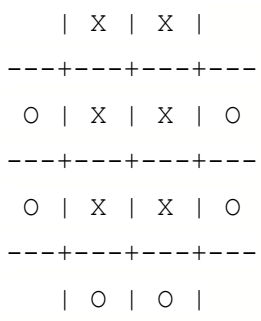
Optional, Extra Credit Problem [20 points]

Use transposition tables (aka cache-ing or memo-ization) to speed either of minimax or alpha-beta. Report and interpret your results.

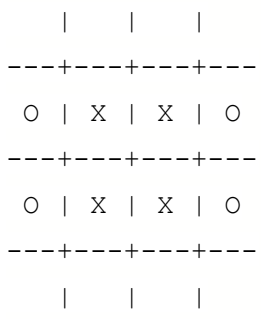
TEST BOARDS FOR 4x4 TIC-TAC-TOE

```
BOARD 1
 X | X | X | X
---+---+---+---
 O | O | O |
---+---+---+---
   |   |   |
---+---+---+---
   |   |   |

BOARD 2
 O | O | O | O
---+---+---+---
 X | X | X |
---+---+---+---
   |   |   |
---+---+---+---
   |   |   |

BOARD 3
 X | X | X |
---+---+---+---
 O | X | X | O
---+---+---+---
 O | X | X | O
---+---+---+---
   | O | O | O
```

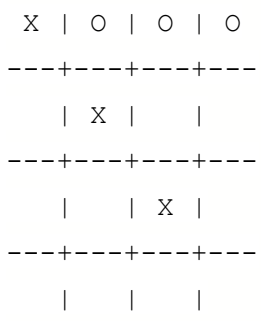
BOARD 4



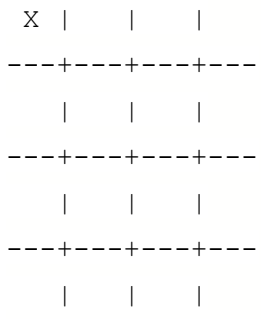
BOARD 5



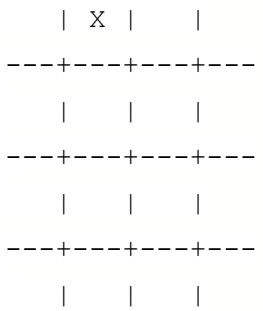
BOARD 6



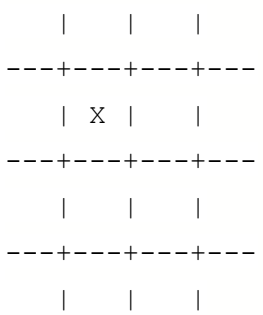
BOARD 7



BOARD 8 (aka Elon Musk’s favorite)



BOARD 9



BOARD 10