1. Consider three versions of tic-tac-toe.

U: usual tic-tac-toe,
A: game ends when x gets 3-in-a-row (in which case x wins) or when board full (in which case x wins if and only if x has 3-in-a-row, otherwise o wins),
B: game ends when x gets 3-in-a-row (in which case x wins) or when o gets 3-in-a-row (in which case o wins) or when board full (in which case x wins if and only if x has 3-in-a-row, otherwise o wins).

For each version, for each case, give the minimax outcome:
(a) if the first move is x to the middle
(b) if the first move is o to the middle
(c) if the first move is x to a side (not the middle, not a corner)
(d) if the first move is o to a side (not the middle, not a corner).

2. For any tic-tac-toe position \( P \), define \( Z(P) \) as the position obtained from \( P \) by changing every x to an o and vice versa. Here is an example:

\[
\begin{array}{ccc}
P & x & . . \\
. . & o & . x \\
o & x & o . \\
\end{array}
\]

For any tic-tac-toe position \( P \), define \( P_x \) as the game that starts at position \( P \) with x moving next. Define \( P_o \) similarly.

For each of I, II, give a proof or a counterexample. Hint: consider strategy stealing.

Claim I: for version B tic-tac-toe, if x has a winning strategy for \( P_x \), then o has a winning strategy for \( Z(P)_o \).

Claim II: for version B tic-tac-toe, if o has a winning strategy for \( P_o \), then x has a winning strategy for \( Z(P)_x \).

3. For Hex on the \( n \times n \) board, give Nash’s proof that the first player has a winning strategy.

4. For Hex on any rectangular board (so, any number \( m \) of rows, and any number \( n \) of columns), give Pierce’s proof that the game cannot end in a draw.
5. (i) In the simple Hex solver `hex3.py` in the class repo, explain the difference between `has_win()` and `can_win()`.

(ii) In `can_win()`, explain each line of code.

(iii) How long do you think this program should take to solve 3x3 positions? Explain briefly.

(iv) Do you think that a similar program would solve 5x5 positions in a reasonable amount of time? Why, or why not? What about 4x4 positions?

6. Find all winning first moves on the 3×3 Hex board. Check your answer using `hex3.py`.

7. Explain why the 4.3.2 side connection is a safe connection. (See chapter 9 in *Hex the full story*.

8. Explain how three semiconnections form the 7.6.5.2 side connection.

9. In figure 9.5 in *Hex the full story*, explain how we know that Black can win from this position, even if White plays next. In your own words, define *virtual connection*. Explain the difference between *virtual connection* and *semiconnection*.

10. For these 3x3 Hex positions, with * to play next, find all winning moves. Explain how you know each move is winning or losing.

   * * *   * * *   * * *
   o * o . o   o . . o o   o . . . o
   o . . . o   o . . . o   o . o . o
   o . . . o   o . . . o   o . . . o
   * * *   * * *   * * *

   * * *
   o * o . o
   o . . . o
   o . . . o
   * * *

   * * *
   o * o . o
   o . . . o
   o . . . o
   * * *

   * * *
   o * o . o
   o . . . o
   o . . . o
   * * *