1  Minimax (10 points)

Do problem 5.7 from the textbook.

2  Node Ordering (10 points)

Consider figure 5.2 from the textbook.  a) Provide a node expansion ordering that maximizes the amount of pruning possible with alpha-beta and show which nodes are pruned.  b) Provide a node expansion order that minimizes the amount of pruning possible with alpha-beta and show which nodes (if any) are pruned.

3  Search Implementation (20 points)

Implement alpha-beta pruning for tic-tac-toe using the evaluation function from problem 5.9 from the text.  Your implementation of the search algorithm itself should be generic and should be easily adapted to other games by changing the evaluation function and node expansion function.  Demonstrate that your alpha-beta pruning code is correct by showing that it produces the same values for the root for each move as what you get when you run minimax to the same depth (depth 2 as suggested in problem 5.9).  Do this by having your program play against itself and report the root values for the first player with first alpha-beta turned on, and then with alpha-beta turned off for an entire game.

4  k-ary constraints in CSPs (10 points)

Do problem 6.6 from the text.

5  CSP example and representation choices (10 points)

Do problem 6.7 from the text.  For the part that asks about different representations, the key question is what you pick for variables and what the domains of these variables are.  There are two obvious choices here.  Discuss the pros and cons of these choices.

6  Knight Tours (10 points)

A knight tour is a sequence of moves for a single knight on a chess board (different versions vary the dimensions of the board) such that every square is visited exactly once.  If you’re unfamiliar with
the rules of chess, just Google or ask a friend/the TA/the prof. You don’t need to know the full
rules of chess to answer this question - just how knights can move. A *closed tour* involves moving
the knight to a final position where it is exactly one move away from the start position. These are
more difficult to find. We will focus on *open tours* where any tour that hits every square exactly
once suffices.  
a) Describe how to use search to find open knight tours. Be sure to explain how your
solution ensures that board positions are visited *exactly* once.  
b) Describe how to formulate the open knight tour problem as a CSP. What are the variables and what are the constraints? Explain
how your solution is satisfiable iff there exists a valid knight tour.