The University of Adelaide
School of Computer Science

Artificial Intelligence
Tutorial 2

Question 1
Figure 1 shows a game tree to be searched based on the minimax algorithm. The root of the tree corresponds to Max's turn.

a) Fill in the minimax value at each node.

b) If *alpha-beta pruning* is to be used to search the game tree, clearly circle the branches that are pruned (i.e, branches that do *not* need to be searched). Note: at each node, search the child nodes from left to right.

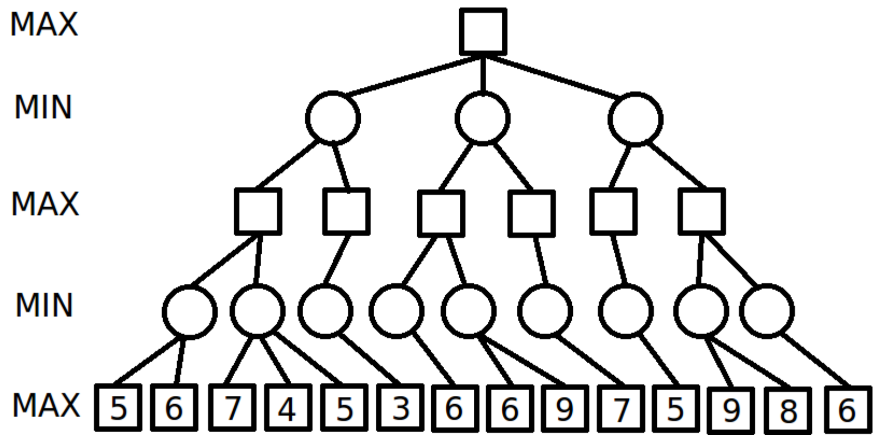

Figure 1: Game tree

Question 2 (Question 6.1 of AIMA 2ed)
Take the game of tic-tac-toe. Draw the game tree to a depth of two (shows the starting board at root and then has two more levels). *Use symmetry to avoid drawing states that are essentially equivalent.* Now annotate the "leaf nodes" (leaves of this partial tree) with utility according to the following. We define X_n as the number of rows, columns or diagonals with nothing else in them but n X's. Likewise define O_n for the noughts.

For a complete tree, the utility function assigns +1 to any position with $X_3 = 1$ (the game has been won by the X player) and -1 to any position with $O_3 = 1$ (the game has been won by the O player). All other terminal nodes would have utility zero.

Non-terminal leaf nodes (i.e., leaves of a partial tree for a certain lookahead) are given a utility by a linear relation $Eval(s) = 3X_2(s) + X_1(s) - (3O_2(s) + O_1(s))$.

Annotate your leaf nodes with their cost and then use Min-Max analysis to annotate all other nodes.

a) What is the best starting move according to this analysis?

b) Circle the nodes that would not be evaluated using alpha-beta pruning *assuming an ordering of the leaf nodes that maximises the benefit of such pruning*.

Question 3 (Question 6.3 of AIMA 2ed)
Consider the two-player game based on the board with 4 squares as shown in Figure 2. The starting position is as shown in the figure. Player A moves first. The two players take turns moving, and each player must move his token to an open adjacent space in either direction. If the opponent occupies an adjacent space, then a player may jump over the opponent to the next open space if any. (For example, if A is on 3 and B is on 2, then A may move back to 1.) The game ends when one player reaches the opposite end of the board. If player A reaches space 4 first, then the value of the game to A is +1; if player B reaches space 1 first, then the value of the game to A is -1.

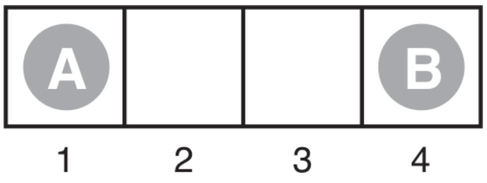

Figure 2: Board for simple two-player game.

(a) Draw the complete game tree, using the following conventions:

1. Write each state as (sA, sB), where sA and sB denote the token locations.
2. Put each terminal state in a square box and write its game value in a circle.
3. Put loop states (states that already appear on the path to the root) in double square boxes. Since their value is unclear, annotate each with a "?" in a circle.

(b) Now mark each node with its backed-up minimax value (also in a circle). Explain how you handled the "?" values and why.

(c) Explain why the standard minimax algorithm would fail on this game tree and briefly sketch how you might fix it, drawing on your answer to (b). Does your modified algorithm give optimal decisions for all games with loops?

(d) This 4-square game can be generalized to n squares for any $n > 2$. Prove that A wins if n is even and loses if n is odd.