1. Each tree’s root is a MAX node. For each tree, (i) circle all leaves examined in alphabeta search (ii) label nodes A, B, C, D with their final minimax value (e.g. 13) or minimax bound (e.g. \( \geq 5, \leq 9 \)).

![Diagram of trees with labeled nodes]

2. Modify tic-tac-toe: 1st player \( x \) wins if she gets 3-in-a-row, otherwise 2nd player \( o \) wins. Give the first-player minimax value for this game: \( \text{(circle one)} \) win lose . For this game, draw a proof tree for \( x \) to play from the position below: use the position as the root node in your tree. (The proof tree will show how the winning player can win.)

```
....
..o..
 x o x
```
3. Here are two options for finding nim values: A) minimax the tree of all possible continuations of the starting position. B) minimax the tree reduced by grouping isomorphic positions (e.g. (3 4 5) and (5 4 3)) into one equivalence class.

For nim(2, 2), for both A), B),
(i) draw the next two levels of the search tree (root’s children and grandchildren) and
(ii) give the total (not just in the top 3 levels) number of nodes in the tree

<table>
<thead>
<tr>
<th>option A)</th>
<th>option B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>top level</td>
<td>(2 2)</td>
</tr>
<tr>
<td>next level</td>
<td></td>
</tr>
<tr>
<td>next level</td>
<td></td>
</tr>
<tr>
<td>ii) total number nodes</td>
<td>_______</td>
</tr>
</tbody>
</table>

4. At right, show the output from winning for the nim position (1, 2, 3).

```python
def winning(nim_psn, sd, depth): # OUTPUT HERE:
    if nim_psn in sd: return sd[nim_psn] #
    print(' ' * depth, nim_psn) #
    if all(p == 0 for p in nim_psn): #
        sd.update({ nim_psn: False }) #
        return False #
    psn = tuple(sorted(nim_psn)) #
    for j in range(len(psn)): #
        for k in range(psn[j]): #
            child = tuple(sorted(psn[:j] + (k,) + psn[j+1:])) #
            if not winning(child, sd, depth+1): #
                sd.update({ nim_psn: True }) #
                if depth == 0: print('\nwinning: move to ',child) #
                return True #
            sd.update({ nim_psn: False }) #
    if depth == 0: print('\nlosing') #
    return False #
```
solution by Sahir  Each MAX node is $\Delta$. 
2. In modified ttt, notice that the game ends only if all cells are filled, or if x gets three in a row. If x plays first, x can win by starting in a corner. For the given position, the player-to-move x can win, so at levels 0, 2, 4, . . . of the proof tree, each node has exactly one child.

3. ii) 33 nodes with option A, 15 nodes with option B

4.

(1, 2, 3)
(0, 2, 3)
(0, 0, 3)
(0, 0, 0)
(0, 1, 3)
(0, 0, 1)
(0, 1, 1)
(0, 0, 2)
(0, 1, 2)
(0, 2, 2)
(1, 1, 3)
(1, 1, 2)
(1, 2, 2)

losing