**Before you start the assignment, remember that a tree is just a Node containing the pointer to the head of linked list/list of all its children nodes**

**Tic Tac Toe AI**

In this assignment, you will be implementing a basic tic tac toe game, complete with an AI agent that can play against a human. If implemented correctly, the AI will always draw the game at worst.

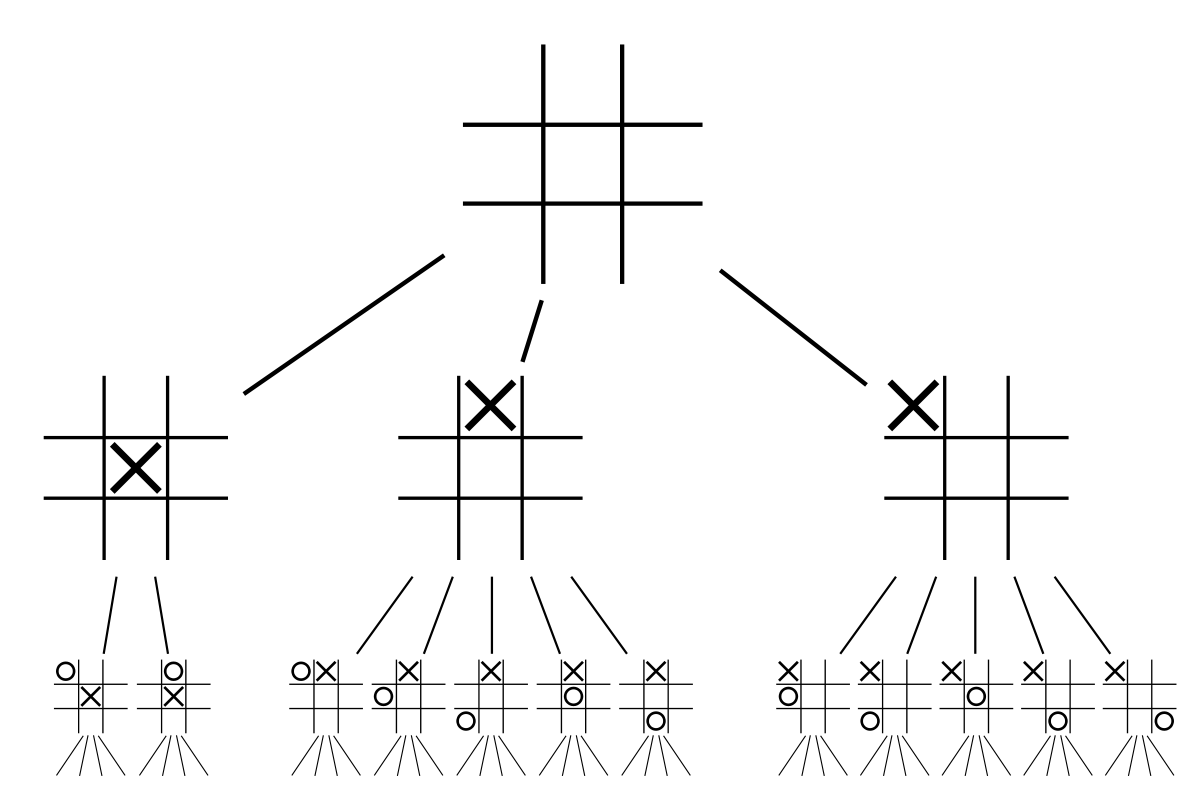
There are 4 files you have to edit to make this game work. The class structures and definitions have been defined in the corresponding header files with the appropriate comments, so there should not be any ambiguity.

**Part 1: (20 marks) Tic Tac Toe Board**

Implement all the functions in the board.h file. This file defines the Tic Tac Toe board class, and most of it is self explanatory.

**Part 2: (35 marks) Tree generation and scoring**

In this part, you will be populating a tree with all possible configurations of the tic tac toe board.

At run time, make a tree, and populate it according to the principle described in the following diagram (Each parent node has all valid moves as its children). Populate the whole tree and store it in memory. You will be primarily working with the node.h and tree.h files in this part. The primary function that you will be marked is the recursive function *generateTicTacToe* in the tree.h file, although you will also have to implement the other functions in these files too. The leaves of the tree will contain the final possible configurations of the board before a draw or winning condition, and you can use this result as a way of giving scores to nodes and sum it up to their parents. (experiment with different scores and go with the one you think is the best).

**Part 3: (10 marks) Main file and AI move**

In the main file, implement a basic Tic Tac Toe game, where the first move is made by the user, while the second move is made by the AI to update the board. The AI will make its move, by choosing the board with the highest score in the immediate next level in the tree. *MakeMove* function in Tree class.

**Part 4: (25 marks) Minimax Algorithm**

Instead of having the AI make a move on the simplistic algorithm in the last part, implement the minimax algorithm, as described in the following link. The header files do not have any functions for this part, so make any functions as needed.

<http://neverstopbuilding.com/minimax>

**Part 5: (10 marks) Analysis, Limitations**

Write comments at the top of your main file with some **limitations** of this approach of generating complete game trees like done in this assignment. Is this approach feasible for say, a 10x10 board? What would be an **alternative** solution considering the limitations you have mentioned.