Connect3

Write a program that plays “connect three” (a simplified version of connect four) against a human. You do not have to implement any Graphical User Interface. A command line interface that outputs the board state as a sequence of characters is fully sufficient.

The game is played on a 4×4 board and players alternately put a game piece of their color (either white or black) on the bottom-most empty field of one of the 4 columns. At each point in the game a player can thus choose among at most 4 moves (putting one game piece into one of the columns that are not filled yet). The objective for each player is to achieve a configuration where three game pieces of the player’s color line up (either horizontally, vertically, or along a diagonal). Whichever player achieves this first wins.

You have to write a program that (starting from an initially empty board) plays against a user. The program should first ask if the human or the computer has the first move and then start the game. Whenever it is the user’s turn, the program should read a move from the keyboard. For moves by the computer you should implement an appropriate game search (this also requires the capability to identify terminal nodes, i.e. nodes in which either one of the players has 3 pieces in a row or where all the fields on the board are filled). Once the game terminates your program should indicate who won or if it is a draw.

To limit the search time of the computer (in most situations this problem is too complex to search it exhaustively) you have to implement a limit on the number of nodes that are opened before the search terminates. For this assignment your program should open no more than 400 nodes before choosing a move (if you make this parameter flexible you can use it to regulate the strength of your computer player). Not being able to exhaustively construct the search tree implies that you have to implement a heuristic evaluation function which replaces the actual utility in cases where the search stops before reaching terminal nodes (such an evaluation function could look at a variety of different features - e.g. how many times does either player have 2 pieces in a row? In which row are the pieces ? ...). Remember, the admissible criterion does not apply to evaluation functions in game searches.