**CSE 5360 AI-1**

**Assignment-4**

**Read Me**

**Name: Sagar Sharma**

**UTA ID: 1001626958**

Programming Language used: **Python 2**

**Note: All the files must be in the same directory.**

The files included are:

* maxconnect4.py
* Maxconnect4Game.py
* Minimax.py
* Maxconnect4Game.pyc
* Minimax.pyc
* input1.txt
* input2.txt
* input3.txt
* input4.txt
* computerplayer.txt
* humanplayer.txt

**Note: For any other input file to be used to run the program besides the 4 input files given with the zip folder, should have the board state in the format as given by the professor.**

***Commands to run the program (works on omega):***

For the interactive mode the following arguments should be passed

**python maxconnect4.py interactive [input\_file] [computer-next/human-next] [depth]**

For the one-move mode the following arguments should be passed

**python maxconnect4.py one-move [input\_file] [output\_file] [depth]**

***Examples:***

***For interactive mode:***

python maxconnect4.py interactive input1.txt computer-next 5

python maxconnect4.py interactive input2.txt human-next 10

python maxconnect4.py interactive input3.txt computer-next 20

python maxconnect4.py interactive input4.txt human-first 44

***for the one-move mode:***

python maxconnect4.py one-move input1.txt output1.txt 8

python maxconnect4.py one-move input2.txt output2.txt 15

python maxconnect4.py one-move input3.txt output3.txt 30

python maxconnect4.py one-move input4.txt output4.txt 60

**Program Structure:**

Note: Since the scaffolding code for the maxconnect4.py and MaxConnect4Game.py is already given by the professor, I have just described the additional changes that I have made to those files as well as the *separate implementation of the depth-limited alpha-beta pruned minimax algorithm* class has been explained.

**maxconnect4.py** consist of the following methods added by me:

* **oneMoveGame()** has the depth-limited alpha-beta pruned minimax algorithm instead of the random play code given in the scaffolding code.
* **afterMoveOperations()** displays the game board state after the move played by either of the players.
* **interactiveGame()** implements the interactive mode for the maxconnect4 game.

**Minimax class** performs the decision making and has the following methods:

* **possibleMoves()** returns the moves that are possible on a state.
* **result()** gives a new state after taking a particular move, based on utility values returned by the depth-limited board states with the help of a eval function based on the scores belonging to those sub board states according to the rules of the game for either of the players.
* **makeDecision()** returns the minimax decision.
* **maxVal()** performs the alpha-beta pruned maximizing operation.
* **minVal()** performs the alpha-beta pruned minimizing operation.
* **utility()** returns the utility value that needs to be maximized or minimized by the maxVal() or minVal() methods and ultimately the decision would be taken by the makeDecision() method.

**COMPARISION WITH PROFESSOR’S PROGRAM**: Yes, the program has depth-limited, alpha-beta pruned minimax implemented and can be compared against the professor’s program.

**References**:

* Used the scaffolding code given by the professor of maxconnect4.py and Maxconnect4Game.py.
* Also made use of the pseudocode for alpha-beta pruned minimax provided in professor’s slides.