CPSC 352 Artificial Intelligence

Programming Project 2

March 22, 2021

Due Friday April 9, 2021 (11:59 pm)

For this project you will write an implementation of Tic-Tac-Toe using MINIMAX.

Tic-Tac-Toe is played on a 3 x 3 grid. The two players are X and O. X always goes first. On each turn, a player marks its letter (X or O) on an empty square of the grid. The game ends when either:

(a) one player has succeeded in filling a row, column, or diagonal with its letter, or
(b) as a tie when there are no more empty squares and neither player has won.
We'll consider the score to be +1 for a win, -1 for a loss, and 0 for a draw.

Your program will be X and the user will play O. ***On each turn***, your program will

(a) Run MINIMAX to determine the best move for X and update the game accordingly
(b) Prompt the user for its turn and update the game accordingly.

Identify the 9 squares as numbered 1-9 in row-major order. A session ought to go something like this (user response in **red**):

Welcome to TTT! My move is

```
_ _ _
_ X _
_ _ _
```

Make your move (row-major order):

$ **1**

```
O _ _
_ X _
_ _ _
```

My move is

```
O _ X
_ X _
_ _ _
```

…

X is the winner!

For a slightly higher grade (5%) modify the basic program so that the user first chooses X or O, so that either the program or the user can go first.

I have provided a variant statement of the algorithm below. It is similar to the one used in the YouTube video; it uses a single recursive function rather than a pair of mutually recursive functions as is presented in the text. Feel free to use either.

```
function MiniMax(state, maxPlayer)
     // state is current grid; Boolean maxPlayer true when X's turn
if Term-Test(state)
   return Value(state)
else if MaxPlayer
   maxEval ← −∞
   for each child c of state
      eval ← MiniMax(c, false)
      maxEval ← max(maxEval, eval)  // also need to track maxMove
   return maxEval
else
   minEval ← ∞
   for each child c of state
      eval ← MiniMax(c, true)
      minEval ← max(minEval, eval)
   return minEval
```

Grading Rubric:

Design/Clarity/Style:	20%
Correctness:	75%
Plays-first extra	5%

To hand in:

- Source Code
- One sample run of the program
- A README file with any instructions for compilation or input, and any comments you have about outstanding issues.