Jacob Dunning

CS 2 Final Project

Program Specifications:

1. Problem definition: build an algorithm that reads in an array of 9 characters representing the squares on a tic tac toe board, evaluates all the legal moves it could make to terminal positions (game is won, or cat game), and returns the strongest possible move (the move with the highest score where terminal positions are scored as loss=-1, tie=0, and win=1)
2. System Requirements:
   1. Board interface: the program must be able to output a tic tac toe board
   2. Board functions: the board must update to user-inputted and AI moves in real time, must be able to tell if an inputted move is legal, and must be able to return a Boolean if the board has reached a terminal position
   3. Main function: prompt user for their choice of X’s or O’s, then offer them the ability to move accordingly, allowing one turn for each player

UML Diagram:

|  |
| --- |
| **Class Board** |
| -char spaces[9] |
| +Board()  +bool isLegalMove(char x)  +int AImove()  +bool isTerminal()  +void printBoard()  +void makeMove(char x)  +void foutBoard() |

Behavior:

* Spaces array contains the values of the 9 spaces on the tic tac toe board
* Board() constructor creates a board object, initializes spaces[i] to i+1
* bool isLegalMove(char x) takes a move argument, and tests if the move is legal (if it is beyond the scope of the spaces array, or if the space is already taken, return false. Else, return true)
* int aiMove() returns the index that corresponds to the best move that the AI can take. The return value is found by computing the values of the terminal outcomes that result from each of the legal moves the AI could take given the state of the board at the time of the function’s invocation.
* bool isTerminal() returns true if the board has reached a terminal position, returns false if not
* void printBoard() prints the spaces array in between two pairs of parallel lines to show the state of the board
* void makeMove(char x) takes a char argument, and updates the spaces array accordingly, given that the move is legal and the game is not already in a terminal position
* void foutBoard() writes the state of the board to a separate text file