CISC449 Project README

Synopsis: In this project, we designed a player versus computer tic tac toe game using C. The computer accepts a user input specifying a mark on a 3x3 square board. After this, the computer marks a spot on the board. The board will be marked by X's and O's. The computer states who wins once each square on the board is marked. If there is no winner, the computer states that there is a tie. A certain amount of functions were used to implement this game. The following list contains a description of each function, specifications, and what was able to be proven.

Functions:

initialize: This function is used to initialize the game. It takes in a two dimensional 3x3 board (matrix) and sets it up with empty chars. The board is proven valid along with empty spaces. The loop invariants of this function say that the conditions are true immediately before and after each iteration of the loops. The assigns condition assigns the board matrix which says that those expressions are the only variables in which the pre and post state may differ. The loop assigns condition assigns i, j, and the board matrix. This has the same effect, but with the variables in the loop. Loop variant N-i and loop variant N-j both prove that the procedure terminates and that the post conditions hold.

print_board: This function simply prints a 3x3 two dimensional board on the screen. We were able to prove that the values in the board may differ in the pre and post states. In addition, for every integer i, it was proven that its pre and post states may change and that the function will terminate with the postconditions held.

end_game: The end_game function is used to show the final result which is reported by the computer. For example, if the computer wins, the outcome shows "Winner is: Computer." This function assigns no variable. If the computer were to win the game, it is proven that the result will show the string "Winner is: Computer." If it were to be a tie game, then it is proven that the result will show the string "Tie game." Now, if the computer did not win, and it wasn't a tie game, the program is verified to compile.

comp_turn: The comp_turn function ensures that the computer's move is valid. It uses the minimax algorithm (specified later) which assures that the computer can never lose. It returns the end_game function that searches the board for matching marks.

player_turn:

gridTurn:

coordTurn:

win_check:

diag_check:

tie_check:

minNum:

maxNum:

new_board_check:

minimax: