CSPC 54 : Prolog Assignment-5

Name : Rajneesh Pandey,

Roll no. 106119100,

Class : CSE-B

## Instructions



Code :

:- dynamic board/1.

play\_tic\_tac\_toe:-

    asserta(board([-,-,-,-,-,-,-,-,-])), nl,

    write('Lets play Tic-Tac-Toe!'), nl, nl,

    write('Rules:'), nl,

    write('You will be player O.'), nl,

    write('Give an integer between 1 and 9 for your move'), nl,

    write('

           1 = upper left

           2 = upper middle

           3 = upper right

           4 = center left

           5 = center middle

           6 = center right

           7 = bottom left

           8 = bottom middle

           9 = bottom right

           '), nl,

    human\_turn.

human\_turn:-

    board(Pos),

    print\_board,

    nl,

    user\_input(Turn),

    ins(o, Pos, Turn, NewPos),

    retract(board(Pos)),

    asserta(board(NewPos)),

    check\_winner.

check\_winner:-

    board(Pos),

    staticval(Pos, Val),

    outcome(Val),

    retract(board(\_)).

check\_winner:-

    board(Pos),

    max\_to\_move(Pos),

    computer\_turn.

check\_winner:-

    board(Pos),

    min\_to\_move(Pos),

    human\_turn.

computer\_turn:-

    board(Pos),

    alphabeta(Pos, -50, 50, NewPos, \_),

    retract(board(Pos)),

    asserta(board(NewPos)),

    check\_winner.

outcome(Val):-

    Val = +1,

    print\_board,

    nl,

    write('You lose.').

outcome(Val):-

    Val = -1,

    print\_board,

    nl,

    write('You win.').

outcome(Val):-

    Val = 0,

    print\_board,

    nl,

    write('Its a draw.').

user\_input(X):-

    write('Your Turn: '),

    read(X),

    integer(X),

    X > 0,

    X < 10.

user\_input(X):-

    write('Not a valid Turn.'),

    nl,

    user\_input(X).

moves(Pos, PosList):-

    member(-, Pos),

    player\_turn(Pos, Symbol),

    get\_moves(Pos, PosList, 1, Symbol).

get\_moves(Pos, [List], 9, Symbol):-

        nth0(8, Pos, -),

    ins(Symbol, Pos, 9, List).

get\_moves(\_, [], 9, \_).

get\_moves(Pos, [Head| Tail], Number, Symbol):-

    nth1(Number, Pos, -),

    ins(Symbol, Pos, Number, Head),

    NewNumber is Number+1,

    get\_moves(Pos, Tail, NewNumber, Symbol).

get\_moves(Pos, PosList, Number, Symbol):-

    NewNumber is Number+1,

    get\_moves(Pos, PosList, NewNumber, Symbol).

ins(Symbol, List, N, Result):-

    length([\_| T], N),

    append(T, [\_| R], List),

    append(T, [Symbol| R], Result).

player\_turn(Pos, Symbol):-

    max\_to\_move(Pos),

    Symbol = x.

player\_turn(Pos, Symbol):-

    min\_to\_move(Pos),

    Symbol = o.

% min\_to\_move/1 checks if the human is at turn.

min\_to\_move(Pos):-

    countO(Pos, O),

    countX(Pos, X),

    O =< X.

% max\_to\_move/1 checks if the computer is at turn.

max\_to\_move(Pos):-

    countO(Pos, O),

    countX(Pos, X),

    X < O.

% countO/2 checks the occurrences of the amount of o in the current

% board.

countO( [], 0).

countO([o | Tail], O):-

    countO(Tail, NewO),

    O is NewO + 1.

countO([Head | Tail], O):-

    Head \= 'o',

    countO(Tail, O).

% countX/2 checks the occurrences of the amount of x in the current

% board.

countX( [], 0).

countX([x | Tail], X):-

    countX(Tail, NewX),

    X is NewX + 1.

countX([Head | Tail], X):-

    Head \= 'x',

    countX(Tail, X).

% Value of each terminal node.

staticval(Pos, +1):-

    Pos = [x,x,x,\_,\_,\_,\_,\_,\_];

    Pos = [\_,\_,\_,x,x,x,\_,\_,\_];

    Pos = [\_,\_,\_,\_,\_,\_,x,x,x];

    Pos = [x,\_,\_,x,\_,\_,x,\_,\_];

    Pos = [\_,x,\_,\_,x,\_,\_,x,\_];

    Pos = [\_,\_,x,\_,\_,x,\_,\_,x];

    Pos = [x,\_,\_,\_,x,\_,\_,\_,x];

    Pos = [\_,\_,x,\_,x,\_,x,\_,\_].

staticval(Pos, -1):-

    Pos = [o,o,o,\_,\_,\_,\_,\_,\_];

    Pos = [\_,\_,\_,o,o,o,\_,\_,\_];

    Pos = [\_,\_,\_,\_,\_,\_,o,o,o];

    Pos = [o,\_,\_,o,\_,\_,o,\_,\_];

    Pos = [\_,o,\_,\_,o,\_,\_,o,\_];

    Pos = [\_,\_,o,\_,\_,o,\_,\_,o];

    Pos = [o,\_,\_,\_,o,\_,\_,\_,o];

    Pos = [\_,\_,o,\_,o,\_,o,\_,\_].

staticval(Pos, 0):-

    \+ moves(Pos, \_).

% This prints out the board in a nice readable format.

print\_board:-

    write('The current board:'),

    nl,

    board([A,B,C,D,E,F,G,H,I]),

    write('\_\_\_\_\_'), nl,

    write('|' ), write(A), write(B), write(C), write('|'), nl,

    write('|' ), write(D), write(E), write(F), write('|'), nl,

    write('|' ), write(G), write(H), write(I), write('|'), nl,

    write('-----'), nl.

% alphabeta/5 predicate, this is the alphabeta algorithm.

alphabeta( Pos, Alpha, Beta, GoodPos, Val)  :-

    moves( Pos, PosList), !,

    boundedbest( PosList, Alpha, Beta, GoodPos, Val);

    staticval( Pos, Val).

boundedbest( [Pos | PosList], Alpha, Beta, GoodPos, GoodVal)  :-

    alphabeta( Pos, Alpha, Beta, \_, Val),

    goodenough( PosList, Alpha, Beta, Pos, Val, GoodPos, GoodVal).

goodenough( [], \_, \_, Pos, Val, Pos, Val)  :-  !.

goodenough( \_, Alpha, Beta, Pos, Val, Pos, Val)  :-

    min\_to\_move( Pos),

    Val > Beta, !;

    max\_to\_move( Pos), Val < Alpha, !.

goodenough( PosList, Alpha, Beta, Pos, Val, GoodPos, GoodVal)  :-

    newbounds( Alpha, Beta, Pos, Val, NewAlpha, NewBeta),

    boundedbest( PosList, NewAlpha, NewBeta, Pos1, Val1),

    betterof( Pos, Val, Pos1, Val1, GoodPos, GoodVal).

newbounds( Alpha, Beta, Pos, Val, Val, Beta)  :-

    min\_to\_move( Pos), Val > Alpha, !.

newbounds( Alpha, Beta, Pos, Val, Alpha, Val)  :-

    max\_to\_move( Pos), Val < Beta, !.

newbounds( Alpha, Beta, \_, \_, Alpha, Beta).

betterof( Pos, Val, \_, Val1, Pos, Val)  :-

    min\_to\_move( Pos), Val > Val1, !;

    max\_to\_move( Pos), Val < Val1, !.

betterof( \_, \_, Pos1, Val1, Pos1, Val1).

**INPUT:** play\_tic\_tac\_toe.

**OUTPUT:**

Lets play Tic-Tac-Toe!  
  
Rules:  
You will be player O.  
Give an integer between 1 and 9 for your move  
1 = upper left 2 = upper middle 3 = upper right 4 = center left 5 = center middle 6 = center right 7 = bottom left 8 = bottom middle 9 = bottom right  
The current board:  
\_\_\_\_\_  
|---|  
|---|  
|---|  
-----  
  
Your Turn:

*2*

The current board:  
\_\_\_\_\_  
|-o-|  
|-x-|  
|---|  
-----  
  
Your Turn:

*3*

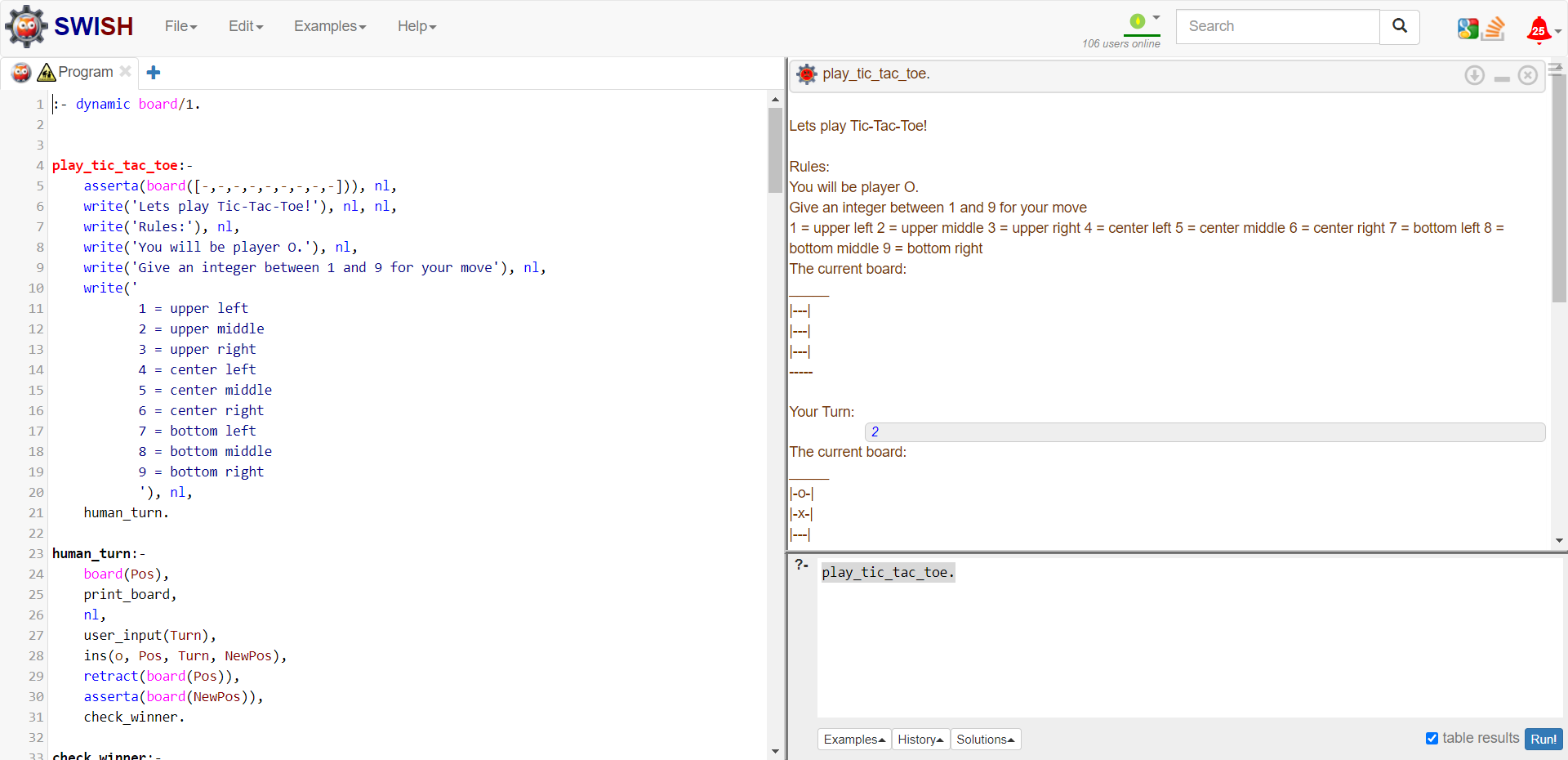
The current board:  
\_\_\_\_\_  
|-oo|  
|xx-|  
|---|  
-----  
  
Your Turn:

*1*

The current board:  
\_\_\_\_\_  
|ooo|  
|xx-|  
|---|  
-----  
  
You win.

*1***true**

**Screenshots:**

****