First board will be created using game\_board function.

We intiliaze win\_game, count,temp with respectively '-' , 0 and 1

Then we declare two more variables name human and ai and get their values using a move2 functions

move2 function :- we ask users if he want to play with 'X' or 'O' if selects other than them then program expect a error and again call itself untill user select right

We declare one more variabe name player\_turn and intiliaze it by user using a function turn2 function

turn2 function :- we ask users if he want to decide if he wants to play first or not so he have to select ("Y or N") if selects other than them then program expect a error and again call itself untill user select right

Now we display a 3x3 matrix to user which has number 1-9 using two for loops and temp variable which we increment each time

1 2 3

4 5 6

7 8 9

exact like that so he/she can select there move by selecting a number from them

Now we run a loop until game is over or no moves left in the board(in this situation count will be 9 --> no moves left situation )

if it is player\_turn then we call players\_turn function

else we call findBestMove function

then we increment count by 1

in both conditions we check for winnner using check\_winner function and displays the board using show\_board

players\_turn function :- We ask user to select number between 1-9 untill user select between them it will call itself recursively

and after geting value we fill the boards that position with human

findBestMove function :- computer find its best move to play using minmax algorithim which is implemented using minmax function

minmax function :-

pseudo code for function

function minimax(node, depth, maximizingPlayer) is

if depth = 0 or node is a terminal node then

return the heuristic value of node

if maximizingPlayer then

value := −∞

for each child of node do

value := max(value, minimax(child, depth − 1, FALSE))

return value

else (\* minimizing player \*)

value := +∞

for each child of node do

value := min(value, minimax(child, depth − 1, TRUE))

return value