LAB-1

Date: 01/10/2024:-

Q. Implement tic- tac toe game.

(#). Program title:
Implement tic- tac toe game.

(#). Algorithm:

function print_board(board):
  Print board format

function check_winner(board):
  for each row in board:
    if all elements in row are same and not empty:
      return the element
  for each column:
    if all elements in column are same and not empty:
      return the element
  if diagonal checks yield same element and not empty
    return the element
  return None

function is_board_full(board):
  return True if no empty spaces, else False

function main():
  initialize board with empty spaces
  iteration = 0
  winner = None

  while winner is None:
    if iteration is even:
      print_board(board)
      get user input for row and column

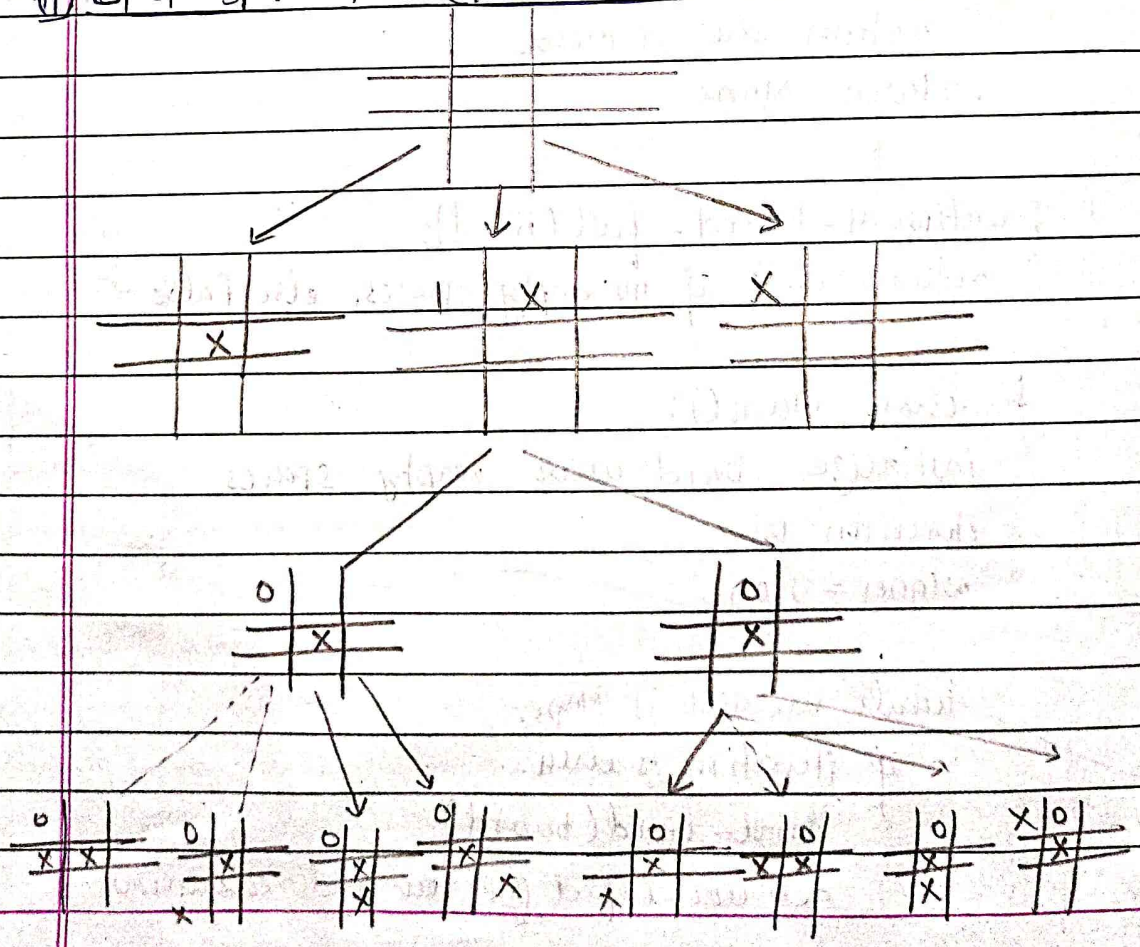
validate input
place 'O' in board
else:
randomly select empty position for 'X'.

Iteration += 1
winner = check_winner(board)

if winner is not None:
print winner message
break

print game introduction
main().

#) State space tree (Tic Tac Toe):



Mangal
Date
Page

Date: 01/10/2024

(#). Program title:
Implement Vaccum Cleaner Agent

(#). Algorithm:

```
function VaccumCleanerAgent(environment):
    position = (0,0)
    cleaned_cells_count = 0

    while True:
        if environment[position] is dirty:
            clean(environment[position])
            cleaned_cells_count += 1
            print("cleaned position:", position)

        next_position = findNextDirty(environment)
        if next_position exists:
            position = next_position
        else:
            print("No more dirty cells found. Cleaning complete")
            break

function findNextDirty(environment):
    for each cell in environment:
        if cell is dirty:
            return cell's position.
    return None.
```

1/10/24

Mangal
Date
Page

#) State space Tree (Vaccum cleaner):