Algorithm for Tic Tac Toe

tic tac Toe Algorithm ( ){

//input:-

//output:-

Step 1:-
   initialize board as 3x3 3D array
   with empty value
   int [ ] [ ] board = new int [3][3]
   for (int i=0; i < 3; i++){
      for (int j=0; j<3; j++){
         board [i] [j] = 0;
      }
   }

Step 2:-
   Set user player = "x" & AI = "0";

S3:
   while Game not over //S3
   if player turn {
         Display Board . ask player for
   input
   if board [row] [column] is empty
         board [row] [column] = 'x'
   else
         ask for another input
   }
}

```
1) AI turn
    if board [row] [column] is empty
        board [row] [column] = '0'
        // first move is AI
    else if player is suppose to win
        block the used player
    else
        place random of row column
            if board [row] [column] is empty
            board [row] [column] = '0'


S4:
    // game over condition
    if board [row] [1, 2, 3] = 'x'
        x win;
    else if board [1, 2, 3] [column] =
        'x'
            player win;
    else if diagonal = 'x'
            player win;
    else if board [row] = '0'
            AI win;
    else if board [column] = '0'
            AI win;
    else if diagonal. board [] = '0'
            AI win
    else
            draw
```
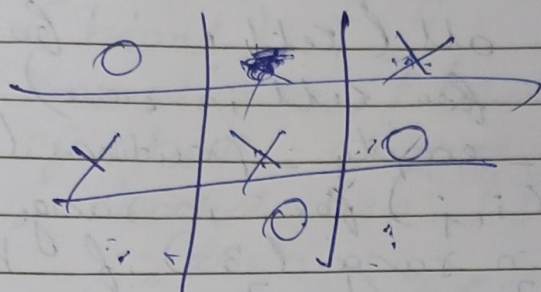
Code

```
import random

def print_board (board):
    for row in board:
        print (" 1 ".join (row))
        print (" - " * 9)

def check_winner (board):
    lines = []
    lines. extend (board)
    lines. extend ([[ board [i][j] for
        i in range (3)] for j in range (3)])
    lines. append ([ board [i][i] for
        i in range (3)])
    lines. append ([ board [i][2-i] for
        i in range (3)])

    for line in lines:
        if line [0] == line [1] == line [2]
            != " ":
            return line [0].
    return None
```

```python
def is_board_full(board):
    return all(cell != '' for row
    in board for cell in row)
def get_empty_position(board):
    return [(i,j) for i in range(3)
    for j in range(3) if board
    [i][j] == '']


def player_move(board):
    while True:
        try:
            row = int(input("Enter row
                (0-2): "))
            col = int(input("Enter column
                (0-2): "))
            if board[row][col] == '':
                board[row][col] = 'X'
                break
            else:
                print("Position already taken.
                    Try again.")
        except (ValueError, IndexError):
            print("Invalid input. Please
                enter numbers between 0 and 2")


def ai_move(board):
    for row, col in get_empty_position
        (board):
        board[row][col] = 'O'
        if check_winner(board) == 'O':
            return
        board[row][col] = ''
```

```python
    for row, col in get_empty_positions
        (board):
        board [row][col] = 'x'
        if check_winner (board) == 'x':
            board [row][col] = 'o'
            return
        board [row][col] = ' '

    row, col = random. choice (get_
    empty_positions (board))
    board [row][col] = 'o'


def main():
    board = [['x' for _ in range (3)]
    for _ in range (3)]
    current_player = 'x'

    while True:
        print_board (board)

        if current_player == 'x':
            player_move (board)
        else:
            ai_move (board)

        winner = check_winner (board)
        if winner:
            print_board (board)
            print (f" {winner} wins!")
            break
        elif is_board_full (board):
            print_board (board)
```

Print ("It's a draw!")
break

current - player = 'O' if current
player == 'x' else 'x'

if _name_ == "_ main _";
main()

Output

```
 |  |
-- --
 1 |  |
 1 -----
 1 |  |
-- ----
```

Enter row (0-2): 1
Enter column (0-2): 1
```
 |  |
 | x |
-- --
 1 | 1 |
  | 1
-- --
 | x |
0 | 1 |
```

Enter row (0-2): 0
Enter column (0-2): 0
```
 x |  |
 | x |
---- --
 0 | 1
-- --
 x | 1 |
```

```
  | X |
- - - - -
O |   | O
```

Enter row (0-2) : 2
Enter column (0-2) : 1

```
X |   |
- - - - -
  | X |
- - - - -
O | X | O
```

```
X | O |
- - - - -
  | X |
- - - - -
O | X | O
```

Enter row (0-2) : 1
Enter column (0-2) : 2

```
X | O | 
- - - - -
  | X | X
- - - - -
O | X | O
```

```
X | O |
- - - - -
O | X | X
- - - - -
O | X | O
```

Enter row (0-2) ≠ 0
Enter column (0-2) : 2

```
X | O | X
- - - - -
O | X | X
- - - - -
O | X | O
```

It's a draw!