(1) Tic - Taa - Toc Game Algorithm:

(i) Initialization of Game Board:
Create a 3x3 matrix using 2-D array to represent the board, initialized to empty states.

(ii) Display the Board to show the current state of the board.

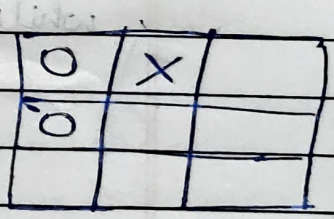
(iii) User Input : allow the user to input their move (row and column)

(iv) Check for Win : check if a player has won the game.

(v) Check for Draw : check if the board is full and there is no winner.

(vi) ~~Suitable~~ Algorithm to be implemented for computer's move ⇒

→ ① Winning Move
→ ② Blocking Move
→ ③ Center Move
→ ④ Corner move
→ ⑤ Side Move



(vii) Check for win or draw again : Once the computer moves, check for a win or draw.

(viii) Repeat Steps 3-6 alternating between user and computer until the game ends.

DATE: PAGE:

```
import random

board = [[" ", " ", " "], [" ", " ", " "], [" ", " ", " "]]

def print_board():
    for row in board:
        print("|".join(cell if cell != " " else " " for cell in row))
        print("-" * 9)

def start():
    print("Welcome to Tic-Tac-Toe!")
    random_num = random.randint(0,1)
    if random_num == 0:
        print("Player plays first.")
    else:
        print("Computer plays first.")
        computer_plays()

while True:
    if random_num == 0:
        player_plays()
    if check_win('X'):
        print_board()
        print("Player won!")
    return
random_num = 1

    else:
        computer_plays()
        if check_wins('O'):
            print_board()
            print("Computer Won!")
            return
```

```
random_num = 0

if board_full():
    print_board()
    print("Draw!")
    return

def check_win(player):
    wins = [
        [(0,0),(0,1),(0,2)],
        [(1,0),(1,1),(1,2)],
        [(2,0),(2,1),(2,2)],
        [(0,0),(1,0),(2,0)]
        [(0,1),(1,1),(2,1)]
        [(0,2),(1,2),(2,2)],
        [(0,0),(1,1),(2,2)],
        [(0,2),(1,1),(2,0)],
    ]

for win in wins:
    if all(board[x][y] == player for x,y in win):
        return True
return false

def player_plays():
    printboard()
    while True:
        try:
            a,b = map(int, input("Enter the co-ordinates
            (row and column 0-2, eg: '01'): ").split())

            if board[a][b] == ' ':
                board[a][b] = 'X'
                break
```

```
else:
    print("Cell already taken. Try again")

except(ValueError, IndexError):
    print("Invalid input. Please enter coordinates in the
        format 'row column'.")

def computer_plays():
    for win in check_possible_wins('O'):
        if board[win[0]][win[1]] == '':
            board[win[0]][win[1]] = 'O'
            return

for win in check_possible_wins('X'):
    if board[win[0]][win[1]] == '':
        board[win[0]][win[1]] = 'O'
        return

while True:
    a, b = random.randint(0, 2), random.randint(0,2)
if board[a][b] == '':
    board[a][b] = 'O'
    break.

def check_possible_wins(player):
    """Return possible winning positions that can be
taken by the specified player."""
    possible_moves = []
    wins = [
        [(0,0), (0,1), (0,2)],
        [(1,0), (1,1), (1,2)],
        [(2,0), (2,1), (2,2)],
```

DATE: PAGE:

```
[(0,0),(1,0),(2,0)],
[(0,1),(1,1),(2,1)],
[(0,2),(1,2),(2,2),],
[(0,0),(1,1),(2,2)]
[(0,2),(1,1),(2,0)],
]

for win in wins:
    if sum(1 for x,y in win if board[x][y]==player)
        ==2:
        for x,y in win:
            if board[x][y]=='':
                possible_moves.append((x,y))

return possible_moves

def board_full():
    return all(cell != '' for row in board for
                                cell in row)

start()
```