COMPUTER NETWORKS

PYTHON SOCKET PROGRAMMING

Title: Tic-Tac-Toe

A MULTIPLAYER TIC-TAC-TOE GAME:

Tic-tac-toe is a two player game in which the objective is to take turns and mark the correct spaces in a 3x3 grid. Two players seek alternate turns to complete a row, a column, or a diagonal with either three O's or three X's drawn in a grid of nine squares. There is no universally-agreed rule as to who plays first, but in this article the convention that X plays first is used. Players soon discover that the best play from both parties leads to a draw. Hence, tic-tac-toe is often played by young children who may not have discovered the optimal strategy. Because of the simplicity of tic-tac-toe, it is often used as a pedagogical tool for teaching the concepts of good sportsmanship and the branch of artificial intelligence that deals with the searching of game trees. It is straightforward to write a computer program to play tic-tac-toe perfectly or to enumerate the 765 essentially different positions (the state space complexity) or the 26,830 possible games up to rotations and reflections (the game tree complexity) on this space.

Multiclient is not possible in this game because only 2 people can play this game.

Name: Sanmat Sanjayakumar Payagoudar

SRN: PES1UG20CS385

Name: Sanjay K N

SRN: PES1UG20CS379

PROTOCOL: TCP

SERVER SIDE CODE:

```
import socket
import threading

class TicTacToe:
    def __init__(self):
        self.board = [[" ", " ", " "], [" ", " ", " "], [" ", " ", " "]]
        self.turn = "X"
        self.you = "X"
        self.opponent = "O"
        self.winner = None
        self.game_over = False

        self.counter = 0

    def host_game(self, host, port):
        server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        server.bind((host, port))
        server.listen(5)

        client, addr = server.accept()

        self.you = "X"
        self.opponent = "O"
        threading.Thread(target=self.handle_connection,
args=(client,)).start()
        server.close()

    def connect_to_game(self, host, port):
```

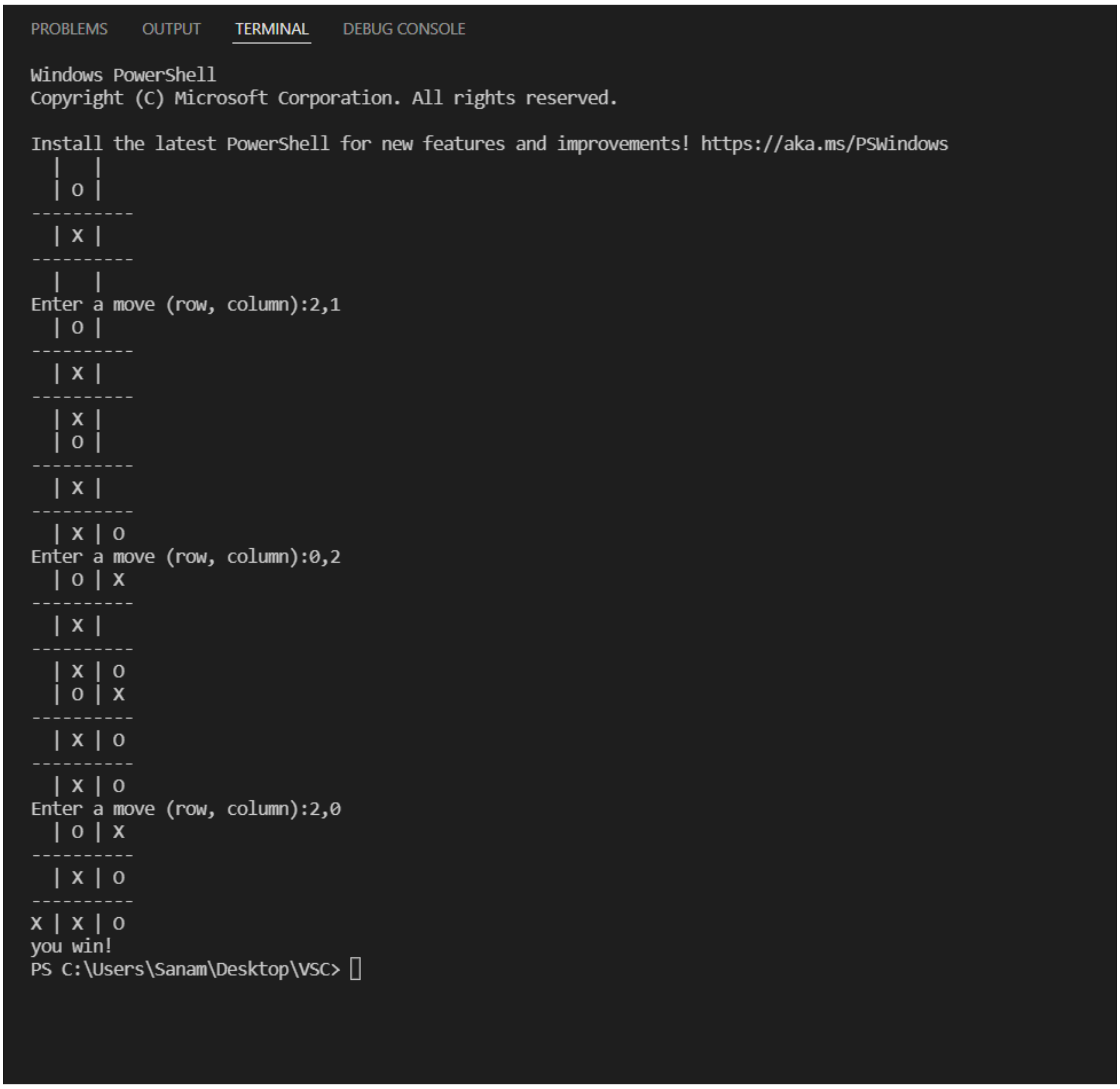
```python
        client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        client.connect((host, port))

        self.you = "O"
        self.opponent = "X"
        threading.Thread(target=self.handle_connection,
args=(client,)).start()

    def handle_connection(self, client):
        while not self.game_over :
            if self.turn == self.you:
                move = input("Enter a move (row, column):")
                if self.check_valid_move(move.split(',')):
                    client.send(move.encode('utf-8'))
                    self.apply_move(move.split(','), self.you)
                    self.turn = self.opponent

                else:
                    print("invalid move")
            else:
                data = client.recv(1024)
                if not data:

                    break
                else:
                    self.apply_move(data.decode('utf-8').split(','),
self.opponent)
                    self.turn = self.you
        client.close()

    def apply_move(self, move, player):
        if self.game_over:
            return
        self.counter += 1
        self.board[int(move[0])][int(move[1])] = player
        self.print_board()
        if self.check_if_won():
            if self.winner == self.you:

                print("you win!")
                exit()
            elif self.winner == self.opponent:
                print("you lose.")
                exit()
            else:
                if self.counter == 9:
                    print("It is a Tie!")
                    exit()
    def check_valid_move(self, move):
        return self.board[int(move[0])][int(move[1])] == " "

    def check_if_won(self):
        for row in range(3):
            if self.board[row][0] == self.board[row][1] == self.board[row][2]
!= " ":
                self.winner = self.board[row][0]
                self.game_over = True
                return True

        for col in range(3):
            if self.board[0][col] == self.board[1][col] == self.board[2][col]
!= " ":
                self.winner = self.board[0][col]
                self.game_over = True
                return True

        if self.board[0][0] == self.board[1][1] == self.board[2][2] != " ":
            self.winner = self.board[0][0]
            self.game_over = True
            return True

        if self.board[0][2] == self.board[1][1] == self.board[2][0] != " ":
            self.winner = self.board[0][2]
            self.game_over = True
            return True

        return False
    def print_board(self):
        for row in range(3):
            print(" | ".join(self.board[row]))
            if row != 2:
                print("----------")

game =  TicTacToe()
game.host_game("10.5.17.106",9999)
```

CLIENT SIDE CODE:

```python
import socket
import threading

class TicTacToe:
    def __init__(self):
        self.board = [[" ", " ", " "], [" ", " ", " "], [" ", " ", " "]]
        self.turn = "X"
        self.you = "X"
        self.opponent = "O"
        self.winner = None
        self.game_over = False

        self.counter = 0

    def host_game(self, host, port):
        server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        server.bind((host, port))
        server.listen(5)

        client, addr = server.accept()

        self.you = "X"
        self.opponent = "O"
        threading.Thread(target=self.handle_connection,
args=(client,)).start()
        server.close()

    def connect_to_game(self, host, port):
        client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        client.connect((host, port))

        self.you = "O"
        self.opponent = "X"
        threading.Thread(target=self.handle_connection,
args=(client,)).start()

    def handle_connection(self, client):
        while not self.game_over :
            if self.turn == self.you:
                move = input("Enter a move (row, column):")
                if self.check_valid_move(move.split(',')):
                    client.send(move.encode('utf-8'))
                    self.apply_move(move.split(','), self.you)
                    self.turn = self.opponent

                else:
                    print("invalid move")
            else:
                data = client.recv(1024)
                if not data:

                    break
                else:
                    self.apply_move(data.decode('utf-8').split(','),
self.opponent)
                    self.turn = self.you
        client.close()

    def apply_move(self, move, player):
        if self.game_over:
            return
        self.counter += 1
        self.board[int(move[0])][int(move[1])] = player
        self.print_board()
        if self.check_if_won():
            if self.winner == self.you:

                print("you win!")
                exit()
            elif self.winner == self.opponent:
                print("you lose.")
                exit()
        else:
            if self.counter == 9:
                print("It is a Tie!")
                exit()

    def check_valid_move(self, move):
        return self.board[int(move[0])][int(move[1])] == " "

    def check_if_won(self):
        for row in range(3):
            if self.board[row][0] == self.board[row][1] == self.board[row][2]
!= " ":
                self.winner = self.board[row][0]
                self.game_over = True
                return True

        for col in range(3):
            if self.board[0][col] == self.board[1][col] == self.board[2][col]
!= " ":
                self.winner = self.board[0][col]
                self.game_over = True
                return True
        if self.board[0][0] == self.board[1][1] == self.board[2][2] != " ":
            self.winner = self.board[0][0]
            self.game_over = True
            return True

        if self.board[0][2] == self.board[1][1] == self.board[2][0] != " ":
            self.winner = self.board[0][2]
            self.game_over = True
            return True

        return False
    def print_board(self):
        for row in range(3):
            print(" | ".join(self.board[row]))
            if row != 2:
                print("----------")



game = TicTacToe()
game.connect_to_game("192.168.78.134",9999)
```

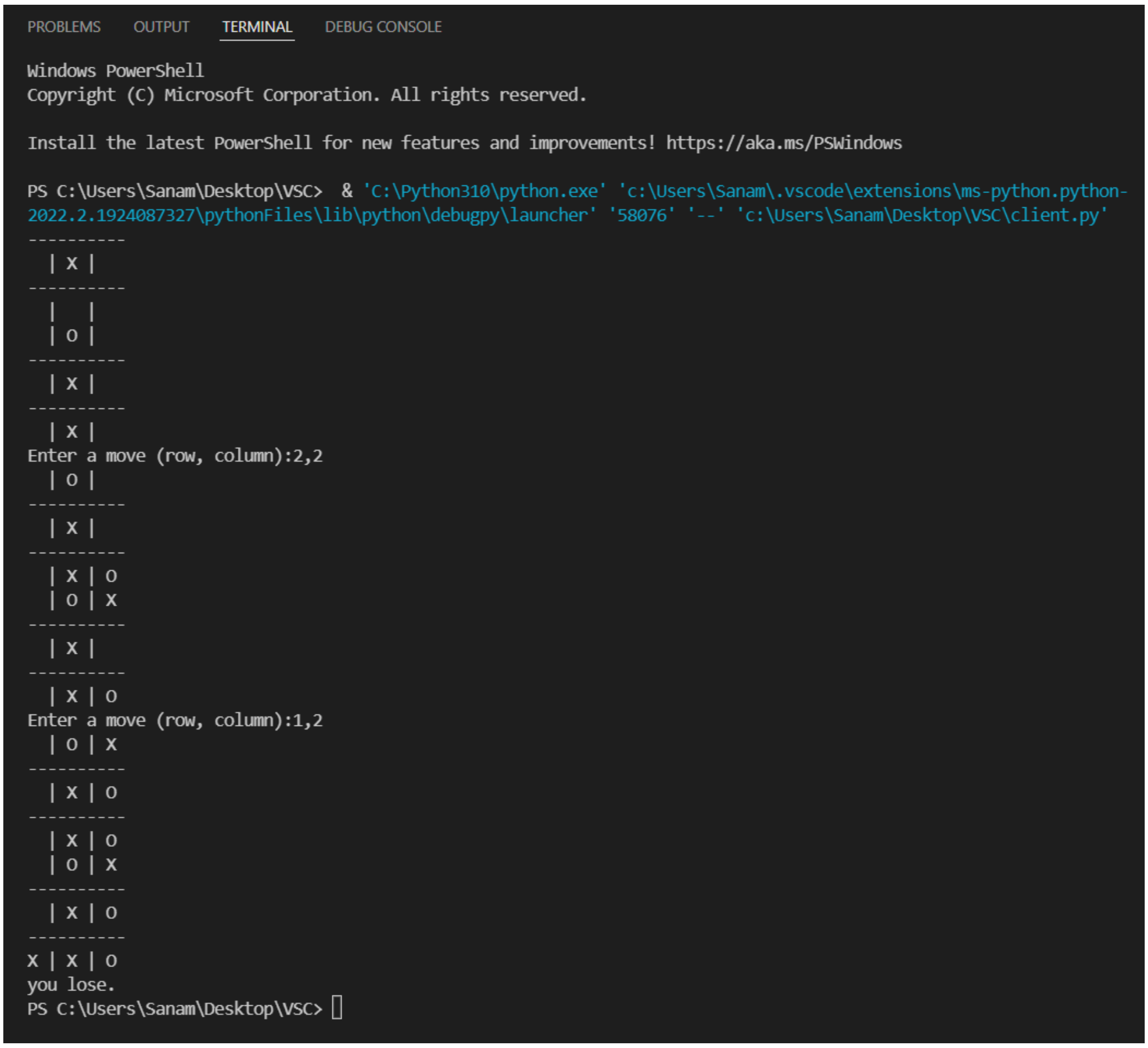
OUTPUT:

SERVER SIDE OUTPUT:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows
  |   |
  | O |
----------
  | X |
----------
  |   |
Enter a move (row, column):2,1
  | O |
----------
  | X |
----------
  | X |
  | O |
----------
  | X |
----------
  | X | O
Enter a move (row, column):0,2
  | O | X
----------
  | X |
----------
  | X | O
  | O | X
----------
  | X | O
----------
  | X | O
Enter a move (row, column):2,0
  | O | X
----------
  | X | O
----------
X | X | O
you win!
PS C:\Users\Sanam\Desktop\VSC> 
```

CLIENT SIDE OUTPUT:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Sanam\Desktop\VSC>  & 'C:\Python310\python.exe' 'c:\Users\Sanam\.vscode\extensions\ms-python.python-
2022.2.1924087327\pythonFiles\lib\python\debugpy\launcher' '58076' '--' 'c:\Users\Sanam\Desktop\VSC\client.py'
----------
  | X |
----------
  |   |
  | O |
----------
  | X |
----------
  | X |
Enter a move (row, column):2,2
  | O |
----------
  | X |
----------
  | X | O
  | O | X
----------
  | X |
----------
  | X | O
Enter a move (row, column):1,2
  | O | X
----------
  | X | O
----------
  | X | O
  | O | X
----------
  | X | O
----------
X | X | O
you lose.
PS C:\Users\Sanam\Desktop\VSC> 
```