**TIC TAC TOE**

import random

print("1BM22CS028 AKASH")

def print\_board(b):

for r in b:

print(" | ".join(r))

print("-"\*10)

def check\_win(b):

for row in b:

if row.count(row[0]) == len(row) and row[0] != ' ':

return True

for col in range(len(b)):

check = []

for row in b:

check.append(row[col])

if check.count(check[0]) == len(check) and check[0] != ' ':

return True

if b[0][0] == b[1][1] == b[2][2] and b[0][0] != ' ':

return True

if b[0][2] == b[1][1] == b[2][0] and b[0][2] != ' ':

return True

return False

def check\_draw(b):

for row in b:

if ' ' in row:

return False

return True

def computer\_move(b):

best\_score = -float('inf')

best\_move = None

for i in range(len(b)):

for j in range(len(b[0])):

if b[i][j] == ' ':

b[i][j] = 'X'

score = minimax(b, 0, False)

b[i][j] = ' '

if score > best\_score:

best\_score = score

best\_move = (i, j)

b[best\_move[0]][best\_move[1]] = 'X'

def minimax(b, depth, is\_maximizing):

if check\_win(b):

if is\_maximizing:

return -1

else:

return 1

elif check\_draw(b):

return 0

if is\_maximizing:

best\_score = -float('inf')

for i in range(len(b)):

for j in range(len(b[0])):

if b[i][j] == ' ':

b[i][j] = 'X'

score = minimax(b, depth + 1, False)

b[i][j] = ' '

best\_score = max(score, best\_score)

return best\_score

else:

best\_score = float('inf')

for i in range(len(b)):

for j in range(len(b[0])):

if b[i][j] == ' ':

b[i][j] = 'O'

score = minimax(b, depth + 1, True)

b[i][j] = ' '

best\_score = min(score, best\_score)

return best\_score

def human\_move(b):

while True:

move = input("Enter your move (row and column number, separated by space): ")

move = move.split()

move = (int(move[0]), int(move[1]))

if b[move[0]][move[1]] != ' ':

print("Invalid move, try again.")

else:

b[move[0]][move[1]] = 'O'

break

def play\_game():

board = [[' ' for \_ in range(3)] for \_ in range(3)]

while True:

print\_board(board)

human\_move(board)

if check\_win(board):

print\_board(board)

print("You win!")

break

elif check\_draw(board):

print\_board(board)

print("It's a draw!")

break

computer\_move(board)

if check\_win(board):

print\_board(board)

print("Computer wins!")

break

elif check\_draw(board):

print\_board(board)

print("It's a draw!")

break

play\_game()

VACCUM

print("Akash ks ")

print("1BM22CS028")

class VacuumCleaner:

def \_\_init\_\_(self, room\_a\_dirt, room\_b\_dirt, starting\_room):

self.current\_state = (room\_a\_dirt, room\_b\_dirt, starting\_room)

def is\_goal\_state(self):

return self.current\_state[0] == 0 and self.current\_state[1] == 0

def clean(self):

if self.current\_state[0] == 1:

self.current\_state = (0, self.current\_state[1], self.current\_state[2])

print("Cleaned room A.")

elif self.current\_state[1] == 1:

self.current\_state = (self.current\_state[0], 0, self.current\_state[2])

print("Cleaned room B.")

def move(self):

if self.current\_state[2] == 'A':

self.current\_state = (self.current\_state[0], self.current\_state[1], 'B')

print("Moved to room B.")

else:

self.current\_state = (self.current\_state[0], self.current\_state[1], 'A')

print("Moved to room A.")

def run(self):

while not self.is\_goal\_state():

print(f"Current state: {self.current\_state}")

self.clean()

if not self.is\_goal\_state():

self.move()

print("Both rooms are clean!")

def get\_initial\_state():

room\_a\_dirt = int(input("Is room A dirty? (1 for yes, 0 for no): "))

room\_b\_dirt = int(input("Is room B dirty? (1 for yes, 0 for no): "))

starting\_room = input("Which room is the vacuum cleaner in? (A or B): ").strip().upper()

if starting\_room not in ['A', 'B'] or room\_a\_dirt not in [0, 1] or room\_b\_dirt not in [0, 1]:

print("Invalid input. Please enter the correct values.")

return get\_initial\_state()

return room\_a\_dirt, room\_b\_dirt, starting\_room

initial\_state = get\_initial\_state()

vacuum = VacuumCleaner(\*initial\_state)

vacuum.run()