Table of Contents

Sl. No.	Title	Page No.
1.	Tic Tac Toe	1 – 4
2.	8 Puzzle Breadth First Search Algorithm	5 - 6
3.	8 Puzzle Iterative Deepening Search Algorithm	7 - 8
4.	8 Puzzle A* Search Algorithm	9_11
5.	Vacuum Cleaner	12 – 15
6.	Knowledge Base Entailment	16 – 18
7.	Knowledge Base Resolution	19 – 21
8.	Unification	22 – 25
9.	FOL to CNF	26 – 28
10.	Forward reasoning	29 – 30

Program 1: Implement Tic Fac Tee

Code:

```
board = [' ' for x in range(10)]
def insertLetter(letter, pos):
    board[pos] = letter
def spaceIsFree(pos):
    return board[pos] == ' '
def printBoard(board):
    print(' | |')
    print(' ' + board[1] + ' | ' + board[2] + ' | ' + board[3])
    print(' | |')
    print('----')
    print(' | |')
    print(' ' + board[4] + ' | ' + board[5] + ' | ' + board[6])
    print(' | |')
    print('----')
    print(' | |')
    print(' ' + board[7] + ' | ' + board[8] + ' | ' + board[9])
    print(' | |')
def isWinner(bo, le):
    return (bo[7] == le and bo[8] == le and bo[9] == le) or <math>(bo[4] == le
and
    bo[5] == le \ and \ bo[6] == le) \ or \ (bo[1] == le \ and \ bo[2] == le \ and
bo[3] == le) or (bo[1] == le and
    bo[4] == le \ and \ bo[7] == le) \ or (
    bo[2] == le \ and \ bo[5] == le \ and \ bo[8] == le) \ or \ (
    bo[3] == le \ and \ bo[6] == le \ and \ bo[9] == le) \ or \ (
    bo[1] == le \ and \ bo[5] == le \ and \ bo[9] == le) \ or \ (bo[3] == le)
    le and bo[5] == le and bo[7] == le)
def playerMove():
    run = True
    while run:
        move = input('Please select a position to place an \'X\' (1-9):
')
        try:
            move = int(move)
            if move > 0 and move < 10:
                 if spaceIsFree(move):
```

```
run = False
                    insertLetter('X', move)
                else:
                    print('Sorry, this space is occupied!')
            else:
                print('Please type a number within the range!')
        except:
            print('Please type a number!')
def compMove():
    possibleMoves = [x for x, letter in enumerate(board) if letter == ' '
and \boldsymbol{x}
    != 0]
    move = 0
    for let in ['0', 'X']:
        for i in possibleMoves:
            boardCopy = board[:]
            boardCopy[i] = let
            if isWinner(boardCopy, let):
                move = i
                return move
    cornersOpen = []
    for i in possibleMoves:
        if i in [1, 3, 7, 9]:
            cornersOpen.append(i)
    if len(cornersOpen) > 0:
        move = selectRandom(cornersOpen)
        return move
    if 5 in possibleMoves:
        move = 5
        return move
    edgesOpen = []
    for i in possibleMoves:
        if i in [2, 4, 6, 8]:
            edgesOpen.append(i)
    if len(edgesOpen) > 0:
        move = selectRandom(edgesOpen)
        return move
def selectRandom(li):
    import random
    ln = len(li)
    r = random.randrange(0, ln)
    return li[r]
def isBoardFull(board):
    if board.count(' ') > 1:
```

```
return False
   else:
       return True
def main():
   print('Welcome to Tic Tac Toe!')
   printBoard(board)
   while not (isBoardFull(board)):
       if not (isWinner(board, '0')):
           playerMove()
           printBoard(board)
       else:
           print('Sorry, 0\'s won this time!')
   if not (isWinner(board, 'X')):
       move = compMove()
       if move == 0:
           print('Tie Game!')
       else:
           insertLetter('0', move)
           print('Computer placed an \'0\' in position', move, ':')
           printBoard(board)
   else:
       print('X\'s won this time! Good Job!')
   if isBoardFull(board):
       print('Tie Game!')
while True:
   answer = input('Do you want to play again? (Y/N)')
   if answer.lower() == 'y' or answer.lower() == 'yes':
       board = [' ' for x in range(10)]
       print('----')
       main()
   else:
       break
```

born board (pos) = deller	Pa agram a: Implement Tic Tac Toe
def invert (.letter, pos): born board (pos) = deter def space-fru (pos): retain board (pos) == '' def print (board): print ('1' + bord [1] +'1' + board [2]+' ib) print ('' + board [4] +'1' + board [5]+ board [5]) print ('' + board [7] +'1' + [4b board [6]+ board [6]) print ('' + board [7] +'' + [4b board [6]+ board [6]) print ('' + board [7] +'' + [4b board [6]+ board [6])	unipoet random
born board (pos) = dutler def Space-fru (pos): retain board [pos] == '' def print (board): print (' 1') print (' 1' + board [1] + ' 1' + board [2] + ' 1' + [board [2] + ' 1' + [b	board = [' for i in range (10)]
def Space-fru(pos): retien board[pos] == '' def print(board): print(' ' + board[1] +' ' + board[2]+' '*board[2]+' '*board[2]+' '*board[2]+'' ' print('' + board[4] +' ' + board[2]+'' ' board[2]) print(' + board[2] + ' ' + [b board[2]+'' ' print(' + board[2] + ' ' + [b board[2]+'' ' print(' + board[2] + ' ' + [b board[2]+'' ' print(' + board[2] + ' ' + [b board[2]+'' ' print(' + board[2] + '' ' + [b board[2]+'' ' print('' + board[2] + '' ' + [b board[2]+'' ' + [b board[2]+'']+'' ' print('' + board[2]+''' + [b board[2]+''']+''' print('' + board[2]+'''' + [b board[2]+''''''''''''''''''''''''''''''''''''	def insert (letter, pos):
def print (board): print ('11') print ('11')	born board (pos) = detter
def print (board): print ('11')	det space-fru (pos):
print(' 1') print(' + bold[1]+' + board[2]+' +b print('') print(''+ board[4]+' +board[6]+ board[6]) print(' + board[7]+' +Ebboard[board[7]+' +Ebboard[board[9]) print(' 1')	retain board[pos] ==
beint('') faint('' + board[4] +'1'+ board[5]+ board[6]) faint ("'') print (' + board[7] + ' + [6] board[board[3]) faint ('1 1')	brint ('11') Asint ('11' + bold [1] + '1' + board [2]+'1'+br
beard[c]) fxint ("") frint (' ' board[7] + ' + [6 board[ubeard [3]) frint (' 1')	Seint('')
fxint ("") print (' ' + board[7] + ' ' + [b board[ubeard [3]) fxint (' 1')	
print (' ') print (' ' + board[7] + ' ' + [6 board[J
	A 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
	hand [9] + Hoboard
	print ('11')
	The state of the s

_	
	def is winner (board, len):
9	def is_winner (board, len): notwisheard [i] == len and bo.[2] == len and board [3]:
	(beard[4) == den and board[5) = den and board[6] == Gen
-	(iboard[7) == len and board[8) == len and board[0)= len
	[board[1] = = len and iboard[4] == len and board [7]=1
	(board (2)= Len and board (5)=len and board (8)== le
	(bord [3] == sen and board(6)=len and board[9]==0
	The state of the s
le	f player ():
	run= True
	while run:
	move = input l'évous a position to place anxilis
	try:
	move = int(move)
	if move 76 and move 210:
	if space-free (move):
	run= False
	uiseit ('X', move)
	elx:
	freint ("Occapied")
	eux:

Date: 17-11-2023 des comproves: Sossible - [of for of with in enumeral move def comprious (): Run = True while run: move = random. nandint(1,16) if (move 70 and move (10): if space & free (move): hun= false inject tollu ('O' more) elx: Continue eus: continue if not (board count (") \$10): player Move () bintBoard (board) if (is winner (board, 1x1)): faint ('you won') Usear etse een: compMove () fru Board (board) if (is winner (board, '0')): yseint ("Computer woon") deleax een: print (" Tie Hus is")

	NI .
	Algorithm: Tic Tac Toe
-	Create a 3×3 board consisting of emptyspace Create function insert () to insert a water to the board and Space fee () to check if the
-	Create function insert () to insert if totter
	the board and space fue () to
	1/03/110.1
-	first allow blayer to play
	- If the board is free, insert & - If the board is free, insert & - Then check if more deads to the player to
	- wen check of
	win or not trins, give computer
	- It the player dog not with
	the chance to play
-	Continue till the board is compty.
	200 00 941.

Output:







