VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.



LAB REPORT

Artificial Intelligence (23CS5PCAIN)

Submitted by

Nischal Kiran(1BM22CS182)

in partial fulfilment for the award of the degree of BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING



B.M.S. COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)

BENGALURU-560019

Sep-2024 to Jan-2025

B.M.S. College of Engineering,

Bull Temple Road, Bangalore 560019(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering



CERTIFICATE

This is to certify that the Lab work entitled "Artificial Intelligence (23CS5PCAIN)" carried out by Nischal Kiran(1BM22CS182), who is Bonafide student of B.M.S. College of Engineering. It is in partial fulfilment for the award of Bachelor of Engineering in Computer Science and Engineering of the Visvesvaraya Technological University, Belgaum. The Lab report has been approved as it satisfies the academic requirements in respect of an Artificial Intelligence (23CS5PCAIN) work prescribed for the said degree.

Swathi Sridharan	Dr. Kavitha Sooda
Assistant Professor	Professor & HOD
Department of CSE, BMSCE	Department of CSE, BMSCE

Index

Sl. No.	Date	Experiment Title	Page No.
1	24-9-2024	Implement Tic –Tac –Toe Game	1
2	1-10-2024	Implement vacuum cleaner agent	9
3	8-10-2024	Implement 8 puzzle problems using Depth First Search (DFS)	13
4	15-10-2024	Implement A* search algorithm Implement Iterative deepening search algorithm	18
5	22-10-2024	Simulated Annealing to Solve 8Queens problem	24
6	29-10-2024	Implement A* search algorithm for N queens Implement Hill Climbing search algorithm to solve N-Queens problem	27
7	12-11-2024	Create a knowledge base using propositional logic and show that the given query entails the knowledge base or not.	33
8	19-11-2024	Implement unification in first order logic	36
9	3-12-2024	Create a knowledge base consisting of first order logic statements and prove the given query using forward reasoning.	41
10	3-12-2024	Implement Min-Max Algorithm for Tic Tac Toe Implement Alpha-Beta Pruning for 8 queens	44

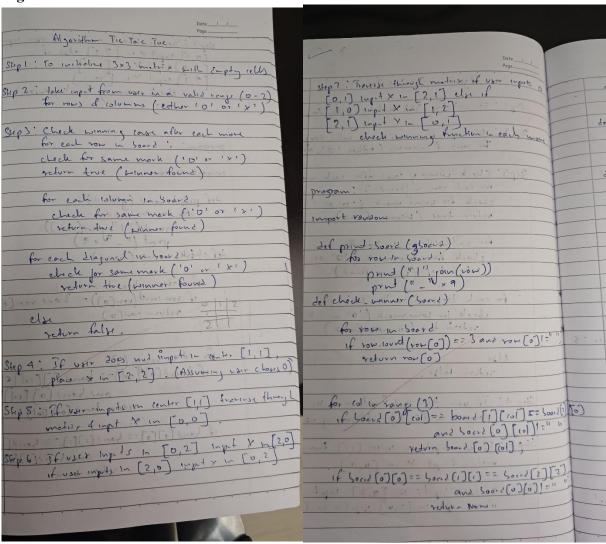
Github Link:

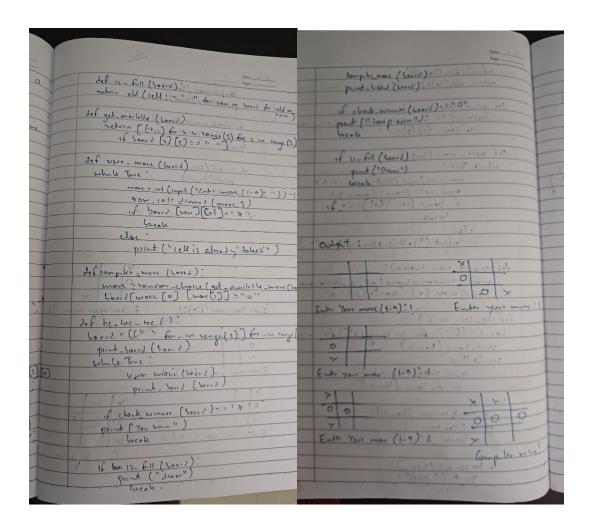
https://github.com/nischal-kiran/AI

Program 1

Implement Tic –Tac –Toe Game

Algorithm:





Code:

import random

```
def win(board): for row in board:
if row[0] == row[1] == row[2] != "":
    return True for col in range(3): if
board[0][col] == board[1][col] == board[2][col] != "":
    return True if board[0][0] == board[1][1]
== board[2][2] != "": return True if
board[0][2] == board[1][1] == board[2][0] != "":
    return True
return False
```

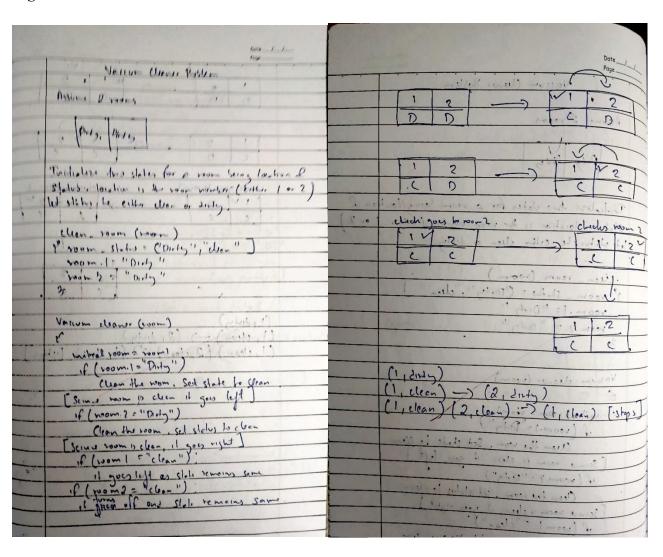
```
def printBoard(board):
  print("\n".join([" | ".join(row) for row in board]))
def draw(board):
  return all(cell != "" for row in board for cell in row)
def user move(board):
  while True:
try:
       move = int(input("Enter your move (1-9): ")) - 1
row, col = divmod(move, 3)
                                    if board[row][col]
                board[row][col] = "X"
== "":
                                                 break
else:
         print("That space is already taken. Try again.")
except (ValueError, IndexError):
       print("Invalid input. Please enter a number from 1 to 9.")
def computer move(board):
while True:
    move = random.randint(0, 8)
row, col = divmod(move, 3)
                                if
board[row][col] == "":
board[row][col] = "O"
break
def _main():
  board = [["" for _ in range(3)] for _ in range(3)]
  while True:
```

```
printBoard(board)
user_move(board)
                      if
win(board):
       printBoard(board)
print("You win!")
          if
break
draw(board):
       printBoard(board)
print("It's a draw!")
                          break
    computer_move(board)
if win(board):
       printBoard(board)
print("Computer wins!")
          if draw(board):
break
printBoard(board)
print("It's a draw!")
break
if __name__ == "__main__":
  _main()
```

```
Enter your move (1-9): 2
| X |
1 1
0
Enter your move (1-9): 9
| X |
0 1 1
1 0 | X
Enter your move (1-9): 1
X | X |
0 1 1
XIOIC
Enter your move (1-9): 5
X \mid X \mid
I X I C
X \mid O \mid C
You win!
```

Implement vacuum cleaner agent

Algorithm:



class Vegum (Legars in word in surprised) class Vegum (Legars	Dete		Otte Nage V
class (self convergence of the convergence of the class o	roje		I (link eminorment (selt):
cet - that = (self survivament). Stiff even war might & businers a company of self cland cells ! felf the parties of self cland cells ! felf the self cells ! felf ! felf ! felf the self cells ! felf	also Name (to a f		
Self. clane of relate 0 self. possition = (0,0): Jef clane (self): Johnie Tree: Ny : Self. possition If self elaboration and the self possition and the self (mithed self): Self. clane of the self possition and the self possition of the self (mithed self): Next possition = self. from the self possition facet possition of the self self (" self self self): Next possition = self. from the self possition facet possition of the self self self possition of the self self self self self the self self self self self self self sel			1/4 " ook (from))
self. position = (0,0). Jef clana (self): phase True. 1. Self. position. 1. Self. controlled [x] [y] = = 1 D! 1. Self. (controlled [x] [y] = = 1 D! 1. Self. (controlled [x] [y] = = 1 D! 1. Self. (controlled [x] [y] = = 1 D! 1. Self. (controlled [x] [y] = = 1 D! 1. Self. (controlled [x] [y] = = 1 D! 1. Self. (controlled [x] [y] = = 1 D! 1. Self. (controlled [x] [y] = = 1 D! 1. Self. position = self. from _ next = desty () 1. Self. position = self. from _ next = desty () 1. Self. position = self. from _ next = desty () 1. Self. position = self. from _ next = desty () 1. Self. position = self. from _ next = desty () 1. Self. position = self. from _ next = desty () 1. Self. position = self. (self. position = self. () 1. Self. (controlled [x] (self. position = self. () 1. Self. (controlled [x] (self. position = self. () 1. Self. (controlled [x] (self. position = self. () 1. Self. (controlled [x] (self. position = self. () 1. Self. (controlled [x] (self. position = self. () 1. Self. (controlled [x] (self. position = self. () 1. Self. (controlled [x] (self. position = self. () 1. Self. (controlled [x] (self. x] (self. position = self. () 1. Self. (controlled [x] (self. x] (self. position = self. () 1. Self. (controlled [x] (self. x] (s			punt to a total change cells ! of cell .
self. position = (0,0). Jef clana (self): phase True. 1. Self. position. 1. Self. controlled [x] [y] = = 1 D! 1. Self. (controlled [x] [y] = = 1 D! 1. Self. (controlled [x] [y] = = 1 D! 1. Self. (controlled [x] [y] = = 1 D! 1. Self. (controlled [x] [y] = = 1 D! 1. Self. (controlled [x] [y] = = 1 D! 1. Self. (controlled [x] [y] = = 1 D! 1. Self. (controlled [x] [y] = = 1 D! 1. Self. position = self. from _ next = desty () 1. Self. position = self. from _ next = desty () 1. Self. position = self. from _ next = desty () 1. Self. position = self. from _ next = desty () 1. Self. position = self. from _ next = desty () 1. Self. position = self. from _ next = desty () 1. Self. position = self. (self. position = self. () 1. Self. (controlled [x] (self. position = self. () 1. Self. (controlled [x] (self. position = self. () 1. Self. (controlled [x] (self. position = self. () 1. Self. (controlled [x] (self. position = self. () 1. Self. (controlled [x] (self. position = self. () 1. Self. (controlled [x] (self. position = self. () 1. Self. (controlled [x] (self. position = self. () 1. Self. (controlled [x] (self. x] (self. position = self. () 1. Self. (controlled [x] (self. x] (self. position = self. () 1. Self. (controlled [x] (self. x] (s			bing (4 loss than
Lef clam (self): place Tree. Ying : self possition. If self elaborationed [x][y] = : D! proof ("inhed foursament") self elaboration ("x][y] = : D! proof ("laboration (self possition for possition for proof ("fined Evaluationment")) next possition = self-strate next elaboration (self possition for possition) proof ("Mooning to rest elaboration for possition for possition) proof ("Mooning to rest elaboration (self possition) proof ("No diving now in (litering longitule")) proof ("No diving now in (litering longitule")) total ("No diving now in (litering longitule")) for all rest elaboration (0,0) Mooning to week diving position (0,0) there is no ready (left (self possitionment)): for it wenge (left (self possitionment)): for it wenge (left (self possitionment)): for it was a left for possition (0,1) **Chan(i,j) *	self. position = (0,0):		11. 12. 12.
ogend * Vaccum (beaut Agent (mithel entropy and in the entropy and in	[''a' 'a']		Initial initial
ogend * Vaccum (beaut Agent (mithel entropy and in the entropy and in	def clone (self):		7 (12.) (12.)
ogend · Varioum (beaut Agent (Inithel entropy of the land of the l	rabile True		1
fill (universal (1) [y] = 1. D. and tripley (universal) self cheaned (1) is a fill postion 3) print ("eleaned postion is self postion 3) print ("eleaned postion is self postion 3) print ("home postion is not divide postion for extent postion) print ("Mooning to west divide postion for extent postion) self postion = self-shade next divide postion for extent postion) self postion = west divide postion for extent postion) print ("Mooning to west divide postion for extent postion) print ("No divide now (trends (a-plat")) total ("No divide now (trends (a-plat")) (trend postion (0,0) Mooning to well divide (self postion with)) for i we read (len (self postion with)): for i weage (len (self postion with)):	xiy = self. position. man -		
fill (universal (1) [y] = 1. D. and tripley (universal) self cheaned (1) is a fill postion 3) print ("eleaned postion is self postion 3) print ("eleaned postion is self postion 3) print ("home postion is not divide postion for extent postion) print ("Mooning to west divide postion for extent postion) self postion = self-shade next divide postion for extent postion) self postion = west divide postion for extent postion) print ("Mooning to west divide postion for extent postion) print ("No divide now (trends (a-plat")) total ("No divide now (trends (a-plat")) (trend postion (0,0) Mooning to well divide (self postion with)) for i we read (len (self postion with)): for i weage (len (self postion with)):	(There were took destroit) being		
self catenard (call is in the self position of the self position of the self substituted ") prind ("steamed position that sight position of a gent display tensionment (") if next position = self-show next distry position funct position? prind ("Moone to west distry position of self position of the self self self self self self self sel	if (ill emironment LX) Ly) == D		
prind ("cleaned position (sidy position b) next-position = self-than next-day () if next-position prind ("known to next-day position facet position) self-position = next-day position self-position = next-position self-position = next-position for it is very position for i in very (self) for i in very (len(self-positionimal)) for i in very (len(self-positionimal)) if self-environment [if(j) = 'D' rebun((j)) That Environment ((n)) ((Self unironment LXJ LyJ = 3 1.01	_	agent display. (univolument ()
prind ("leave & position (self position 2) nut - position = self-struct and district position? if next position: prind ("Monay to most district position facet position? self position = Next position. DD D prind ("No district position. (break size of the self self self self self self self sel	self cheaned well to	_	aged clan ()
Next-position = self-frair next-duty () If next-position is next-duty position facet position) prind ("Monney to next-duty position facet position) self-position = vart-position. Thirtiel fencionament in the position of the position of the classes in the fencionament of the classes in the position (0,0). The first position (0,0). More position (0,0). More position (0,0). More position (0,0). When position (0,1). It self-position (0,1). The first position (0,1). The position of the position of the position of the position of the position (0,1). The position is the position of the positio	print (" cleaned position & self position b)		print ("fine Environment")
If next position is west diety position facut position? sell position 2 west diety position. sell position 2 west position. Defend "No diety rousin. (Herning loop let") pard ("No diety rousin. (Herning loop let") (break """ (bre		-	agent. display. Environment ()
Self mosther = next position. Clearly most (live ming lomple to ") (longer position (0,0) (longer p		-	
Self mosther = next position. Clearly most (live ming lomple to ") (longer position (0,0) (longer p	If next position.	-	011
Old ("No dishy room. (Genia; lompish") (break and dishy (suff)) The horse word adity (suff) (leave) position (0;0) (leave)	print ("Monny to ment dirty position query position)	-	Clil Comment
purd ("No dirty nown. (thering loople to") (break dirty (self)) The hord word dirty (self) and the self to the position (0,0) (there is no reage (len (self, particularly)): for i in range (len (self, particularly)): the deviation of the self particularly): the deletest recons : 0 the deletest recons : 0 the most dirty position (0,0) the most dirty position (0,0) the self self self self self self self sel	sel, position = hext. position.	-	D D
Dreak Continue of the position (0,0) The horal work activity (self) \(\text{ Movement by position (0,0)} \) For i in range (len (self, participant of [i])): For j'in says (len (self, participant of [i])): If the environment [ill] \(Position of the position of the position of the environment of the position of	else 1/1 to Chima langle L')		
About the series of the series	pried ("No diving book. Circular to a priest		Meney anything (Diss)
for i in rease (len(self parisonnet)); for in says (len (self parisonnet [i])); find survanist; if self continuous [i][j]= (D!) then (i,j) table lessiff cells: 2	Dreak solves		Mayor to wel July marthing (01)
for i in rease (len(self parisonnet)); for in says (len (self parisonnet [i])); find survanist; if self continuous [i][j]= (D!) then (i,j) table lessiff cells: 2	The same of the same of		
for i in range (len (self, parisonneral [i])); for in range (len (self, parisonneral [i])); if self, environment [ileg] = (D!) *elvin(i,j) total elevated cells: 2	Det And-WM-21mg (Sur)		No more dishi illy
of self-continuous tile 1 = 0!	for i us rence (len/self. purissume 4)).		
of self-territornerest (itt.) Lotel cleaned cells: 2 . he	Common (len (soll, portionine alt [i]));		final Europaint
return (1,1)	of self-environment [it[] = 'D'		1 , C [1] a junc a jul
and the money	return(i, i)		total elegate celli: 2" . 1
	geturn none		() is want

thorn, to not dirty position (0,1) cleaned position (0,1) cleaned position (0,1) thornes in next dirty position (1,0) cleaned position (1,14) land 10. The more diffy noting of all 1. Final Enumer ment are 1. Complete the second of the se		Date/
total closes d'illo; o cleaned position (0,0) Mooring to with disty position (0,1) cleaned position (0,1) "		Page
deaned partial (0,0) Through the partial (0,1) cleaned partial (0,1) through the partial (0,1) cleaned partial (0,1) through the more dury partial (1,0) through the more dury many that the through t		
total clone ? ich : 0 clone ? position (0,0)? moving to next dirty position (0,1) clone ! position (101) !	_	, p, p
thorn, to next dirty position (0,1) cleaned position (0,1) cleaned position (0,1) cleaned position (0,1) cleaned position (1,1) no more dirty notions of and all the production of the position (1,0) cleaned position (1,1) no more dirty notions of and all the company of and all the company of the position of the company of the position (0,1) cleaned of position (1,1)		D D C T S O T A LE DOUBLE
cleaned position (0,1) cleaned position (0,1) thousage in next during position (1,0) cleaned position (1,11) no more duply resemy of addition Final Enumenment and it is all and cleaned to the position of the position	3")	
cleaned position (10,11) Product (10) Cleaned position (1,11) and (1,0) Cleaned position (1,11) and (1,0) No more disty mining of addition Final Enumerorations and all (1,0) Considerations and (1	_	
House he west during position (1,0) clearly position (1,11) and the self of the more during in solution of the self of the se		
Final Enumerations of the Company of		cleaned position (-0,1)
Final Enument and one of the control	_	Moving to west along position (10)
Final Enumerous and one of the Company of the Compa	-	cleaned youther (1(1))
Confirmation of the state of th	_	no more girth rooms
Confirmation of the state of th		Produce to
Tiplifords		TIMAL ENGINEERS
Tiplifords	-	(CO - F 1 F. 9 HAD?
Ry on the same of the control of the	-	Pinto Variety
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0		
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0		at he had now no south of the dead and
1 2 h		Here the water as it produced water
11 15 (1 to 1		
11 15 (1 to 1	_	457 457
2 1 2 C 1 V		0 3 3 (- 3 0 3
and the street of the street o	-	0.1.1
the go is made all a light of the		1 -11 . It was it is a could be paid
		1 av 11 and who will a local of
	1	E-27:10
	-	
The second secon		
		The second secon

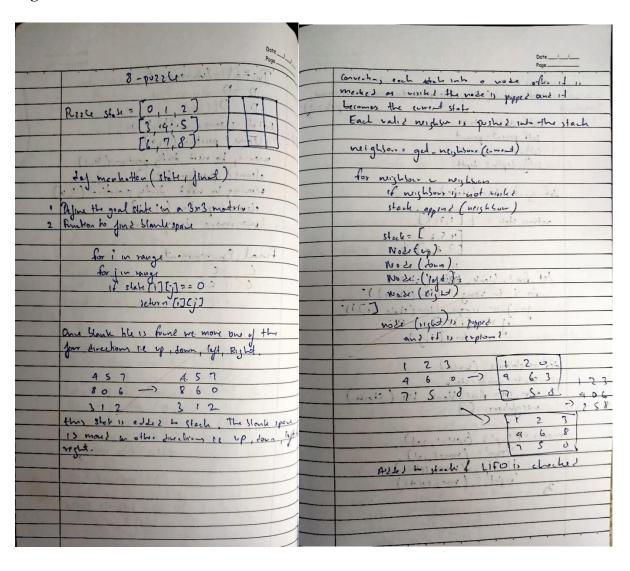
Code:

```
def printArr(arr):
for row in arr:
print(row)
print() def
clean(arr, x, y):
if arr[x][y] == 1:
arr[x][y] = 0 def
check(arr): for
                if
row in arr:
1 in row:
return True
return False
# Directions: right (0,1), down (1,0), left (0,-1), up (-1,0)
directions = [(0, 1), (1, 0), (0, -1), (-1, 0)] direction index
= 0 # Start moving right
# Get room status print("Enter the status of the rooms (0 for
clean; 1 for dirty):") arr1 = [] for i in range(2):
  row = [] for j in range(2):
int(input(f"Status of room ({i}, {j}): "))
row.append(a) arr1.append(row)
x, y = 0, 0 #Start cleaning from the first room
while True: printArr(arr1) if not
check(arr1):
     break
clean(arr1, x, y)
  #Move to the next room in the current direction
dx, dy = directions[direction index] new x,
new y = x + dx, y + dy
```

```
Enter the status of the rooms (0 for clean; 1 for dirty):
Status of room (0, 0): 1
Status of room (0, 1): 0
Status of room (1, 0): 1
Status of room (1, 1): 0
[1, 0]
[1, 0]
[0, 0]
[1, 0]
[0, 0]
[1, 0]
[0, 0]
[1, 0]
[0, 0]
[0, 0]
All rooms are cleaned!
```

Implement 8 puzzle problems using Depth First Search (DFS)

Algorithm:



limits / / When more remains involved the property of the property a class Noze . def - mit - (self; stelle, persent = Mone; Moverno for more (new saw, were (al) to moves them; () rell stife = state ... self. parent = perent Mr Hele [ver] [col) neighbours append (Note (nevertate, note) self. more simone. del- dis almost (stock state, depth a hard); stock = [Mose (stock - state)] def god state (state) Visited - set () return state = [[1,2,3] while stack : [4,5,6] current rete = stack. pop () del find - Stonk talks (stold) If is good (correct mose state):
return reconnect poth (correct mose) for i in rough (lin (state)) for j in reagn (lin (state (i)));

Jethan (i, j) visited (t-ple (map (tople, corrent-node state))). of enred more depth < cepth - limit ! neighbour = get-neighbour (current-node) def neighbours (noce) for neighbour in hurghbours (common noce)
for neighbour in hurghbours shle)) not in wild
stack append (heighbour) She he = note . Stack " vor, 101= find Stonk, hile (Stack) return prove moves = 1 ' vp' : (70 v -1, 101),

1 com = (vom +1, 101),

1 [[f] = (vom + (01-1),

2 reght = (vom + (01-1)) def. recorned - yeth (node): while note porent is not mure peth. opport (byose, mar nose = node . perent return peth [-1] intelestate [[12,3] many [4,0,6] [7,5,8] depth limit = 1000 forest orale solhon = If - limit (spite of - state , soft Solution: ["right" Hown " left" lep! "right" do

Code:

```
class PuzzleState:
                     def init (self, board,
moves=0, previous=None):
     self.board = board
                              self.moves
= moves
              self.previous = previous
self.empty pos = self.find empty()
  def find_empty(self):
                              for
i in range(3):
                      for j in
range(3):
                    if
self.board[i][j] == 0:
            return (i, j)
  def manhattan distance(self):
             for i in range(3):
dist = 0
for j in range(3):
          tile = self.board[i][j]
if tile != 0:
             target x = (tile - 1) // 3
target y = (tile - 1) \% 3
                                      dist += abs(i -
target_x) + abs(j - target_y)
                                  return dist
  def generate moves(self):
moves = []
                 x, y =
self.empty_pos
     directions = [(1, 0), (-1, 0), (0, 1), (0, -1)]
     for dx, dy in directions:
```

```
new x, new y = x + dx, y + dy
if 0 \le \text{new } x \le 3 \text{ and } 0 \le \text{new } y \le 3:
          new board = [row[:] for row in self.board]
                                                                new board[x][y],
       new board[new x][new y] =
                                             new board[new x][new y], new board[x][y]
          moves.append(PuzzleState(new board, self.moves + 1, self))
     return moves
 def dfs(start board, max depth):
= [PuzzleState(start_board)] visited =
       goal state = [[1, 2, 3], [4, 5, 6], [7,
set()
8, 0]]
  while stack:
     current state = stack.pop()
                                     if
current state.board == goal state:
       return current state
visited.add(tuple(map(tuple, current state.board)))
if current state.moves < max depth:
       for next state in current state.generate moves():
          if tuple(map(tuple, next_state.board)) not in visited:
if next state.manhattan distance() < 10:
               stack.append(next state)
return None
def print solution(solution):
  path = []
while solution:
     path.append(solution.board)
solution = solution.previous for step in
reversed(path):
                     for row in step:
print(row)
                         print(f"Total
                print()
```

```
moves taken to reach the final state:

{len(path) - 1}") initial_board = [[1, 2, 3], [4, 0, 5], [7, 8, 6]] max_depth = 10

solution = dfs(initial_board, max_depth)

if solution:

print("Solution found:")

print_solution(solution) else:

print("No solution found.")
```

```
Solution found:
[1, 2, 3]
[4, 0, 5]
[7, 8, 6]

[1, 2, 3]
[4, 5, 0]
[7, 8, 6]

[1, 2, 3]
[4, 5, 6]
[7, 8, 0]

Total moves taken to reach the final state: 2
```

Implement A* search algorithm

Algorithm

Code:

```
def H n(state, target):
  return sum(x != y for x, y in zip(state, target))
def F n(state with lvl, target): state, lvl =
state with lvl return H n(state, target) + lvl def
possible moves(state with lvl, visited states):
  state, lvl = state with lvl b =
state.index(0)
                  directions = []
pos moves = [] if b \le 5:
directions.append('d')
                        if b >= 3:
                        if b % 3 >
directions.append('u')
0: directions.append('l')
                          if b % 3 <
2: directions.append('r')
                           for move
in directions:
     temp = gen(state, move, b)
if temp not in visited states:
       pos moves.append([temp, lvl + 1])
pos moves def gen(state, move, b): temp = state.copy()
if move == 'l': temp[b], temp[b - 1] = temp[b - 1], temp[b]
if move == 'r': temp[b], temp[b + 1] = temp[b + 1], temp[b]
```

```
if move == 'u': temp[b], temp[b - 3] = temp[b - 3], temp[b]
if move == 'd': temp[b], temp[b + 3] = temp[b + 3], temp[b]
print(state[i:i+3]) print() def
for i in range(0, 9, 3):
                                   visited states = []
astar(src, target): arr = [[src, 0]]
iterations = 0 while arr:
    iterations += 1
                       current = min(arr,
key=lambda x: F n(x, target)
arr.remove(current)
                       display state(current[0])
if current[0] == target:
       return f'Found with {iterations} iterations'
visited states.append(current[0])
arr.extend(possible moves(current, visited states))
return 'Not found' src = [1, 2, 3, 8, 0, 4, 7, 6, 5] target
= [2, 8, 1, 0, 4, 3, 7, 6, 5] print(astar(src, target))
```

```
Current State:
[1, 3, 4]
[0, 8, 2]
[7, 6, 5]
Current State:
[8, 1, 0]
[2, 4, 3]
[7, 6, 5]
Current State:
[8, 0, 1]
[2, 4, 3]
[7, 6, 5]
Current State:
[0, 8, 1]
[2, 4, 3]
[7, 6, 5]
Current State:
[2, 8, 1]
[0, 4, 3]
[7, 6, 5]
Found with 40 iterations
```

Implement Iterative deepening search algorithm Algorithm:

Date	Oste Page
Iterative depening search Algorithm	when moved to 8- 1 2 3
19 0 O O O O O O O O O O O O O O O O O O	0.84
200	7 6 5
Deriver depth 10	10 feet = 2+ (+1+(+0+0+0+)
	Listance = 2+1+1+1+0+0+0+1
Q & depth: 1	· 1 · 26.3. (A)
	1 01.25 11.24 h(w)
(3 (digita: 2'	he know f(n)= , (n)+ h(n)
July July - Call	Wall of
6 0 0 0 0 0 0 0	J(n)= 2ph of note
platting	
Step 1: print the root note of = (1)	h(n) = 7 menters at the policy
y U : (v) =	h(n) = 1
Step 2: Print the children of the root note!	g(n) = 0
y had: Not fand Lepth: 0	\$(w) = 0+273] with with the 1 6 5
YPX God: Not found depth:	A series 12 sont
YPR'SXEH Goal : Duch found by this	
Return F	1 2 mm had he(in) + 16
F=311 = (1) 7 2 3 F	0 8 4 g(n)=146=7
Distal Hall problem	7 6 5 \$ (1)= (+6-1)
Inital Hali wo heal	- meldors elsery to the
1 2 30	after 2nd more h(n) = 6-
8 0 4 6 2 0 4 3	1 1 2 1
7 61.5 5 7 6 5	7 6 5 F(m)= 6+1=7
Charles & St. Co. Co. Co. Co. Co. Co. Co. Co. Co. Co	17 6 5 ((m)= 6+1=1
The Stank space can more in 4 disections	
which 15 2, 8, 6, 4.	102 more h(u) - 8
1=(-)- + 5 5	103
when more to & To 103	1 2 4 (m)=1
8 2 4	F(=/+www
765	3 3 12
Heuristic Litanu = 2 + 2 + 1+ 1+0+0+2	Character As a second of the
- 8	

1 2 3 1 2 3 1 2 3 1 0 3 0 8 6 8 2 6 8 2 6 8 2 6 8 8 8 8 8 8 8 8 8			
1 2 3 1 2 3 1 2 3 1 0 3		Date/_	
1 2 3 1 2 3 1 2 3 1 0 3			
123 123 123 123 123 103 103 108 4 8 4 8 4 0 8 6 4 82 4 7 6 5 7 6 7 6			
123 123 123 103 684 840 864 824 765 765 765 765 1(m)= 6 16, 16 16 16 16 16 16 16 16 16 16 16 16 16			
0 8 4 8 4 0 8 64 8 2 4 7 6 5 7 1 6 7 7 5 7 6 5 1 (m) = 1		1765	
0 8 4 8 4 0 8 64 82 4 7 6 5 7 6 5 7 6 5 7 6 5 h(m)= 1			
0 8 4 8 4 0 8 64 82 4 7 6 5 7 6 5 7 6 5 7 6 5 h(m)= 1			
0 8 4 8 4 0 8 64 824 7 6 5 7 6 7 7 0 5 7 6 5 (m)= 6 6 6 7 6 6 7 6 5 (m)= 1		123 123 123	103
1 1 2 1 1 2 1 2 2 0 2 1 2 3 4 1 2 3 4 1 1 2 3 1 1 1 1 1 1 1 2 3 1 1 1 1 1 1 1			
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1		765 765 705 7	65
123 123 1020 123 123 111 134 134 134 134 134 134 134 134 13		h(m)= 1 h()=1 1111=8	
123 123 1020 123 123 111 134 134 134 134 134 134 134 134 13		g(u)= 1 g(u)=1	I TO L
02 2 123 123 120 20 12 3601 1 8 4 78 4 78 4 78 6 78 8 4 7 6 0 1 1 1 2 78 16 16 7 1 16 16 17 18 16 17 18 18 18 18 18 18 18 18 18 18 18 18 18		f(n)=7 f(n)=7 hand	
0 2 3 123 20 20 12 22 3601 1 8 4 78 4 78 4 78 4 78 4 78 4 78 12 7			81
0 2 3 123 102 6 12.361 1 8 4 784 8 4 3 11 8 4 5 11 7 6 5 6 6 5 7 6 5 7 6 0 1 1 2 3 16 10 2 7 16 10 12 5 16 10 12 3 10 2 3 5 10 2 6 5 5 10 12 3 10 2 3 5 10 2 6 5 5 10 12 3 10 2 3 5 10 2 6 5 10 12 3 10 10 10 10 10 10 10 10 10 10 10 10 10			
1 5 4 784 8 4 7 16 8 4 7 16 7 17 17 18 18 18 18 18 18 18 18 18 18 18 18 18			
765 065 765 760 h(1)=7 h(n)=7			
1 = 2			
1 - 2 - 5 (m) = 2 - 5 (m) = 13 - 5 (m) = 2	1.6		
2 12 12 12 12 12 12 12 12 12 12 12 12 12	h(n(x) = 2 $n(x) = 2$ $n(x) = 2$	
2	3(4)=2 (-)(4/82/)(4/26: 5)(4/26:	
2	-		
1 1 10 10 10 10 10 10 10 10 10 10 10 10			
14 14 24 0 1 24 12 16 17 16 18 18 18 18 18 18 18 18 18 18 18 18 18	2.	- (1 1 2 2 8 a) -)	
14 14 24 01 24 14 14 14 14 14 14 14 14 14 14 14 14 14	5		
2-12-12-12-02-12-1 3-12-065	200	e - greater - houself him him	h(in)=
2-12-12-12-02-12-1 3-12-065			1
2-12-12-12-02-12-1 3-12-065	5 15 30	Mara 2004 460 . 1000 30 - 11 - 12 - 12	V
2.171210101111 743			312
		21 12 12 12 12 12 12 1	
6 2) Wale			
		40)	h(u)=S
7 65			
h(n) e S		h(n)es	

Code:

```
def iterative deepening search(graph, start, goal):
def depth limited search(node, goal, depth):
if depth == 0:
                     if node == goal:
return [node]
                     else:
         return None
elif depth > 0:
       for child in graph.get(node, []):
         result = depth limited search(child, goal, depth - 1)
if result is not None:
                                 return [node] + result
return None depth = 0
                          while True:
    result = depth limited search(start, goal, depth)
if result is not None:
       return result
depth += 1 def
get user input graph():
  graph = \{\}
               num edges = int(input("Enter the
number of edges: ")) print("Enter each edge in the
format 'node1 node2':") for in range(num edges):
node1, node2 = input().split()
                                  if node1 in graph:
       graph[node1].append(node2)
else:
       graph[node1] = [node2]
if node2 in graph:
graph[node2].append(node1)
else:
       graph[node2] = [node1]
return graph def main():
  graph = get user input graph() start node = input("Enter the
starting node: ") goal node = input("Enter the goal node: ")
path = iterative deepening search(graph, start node, goal node)
if path:
```

```
print(f"Path found: {' -> '.join(path)}")
else:
    print("No path found") if
    __name__ == "__main__":
main()
```

```
Enter the number of edges: 14
Enter each edge in the format 'node1 node2':
Y X
PR
P S
X F
ХН
R B
R C
SX
SZ
F U
F E
H L
H W
Enter the starting node: Y
Enter the goal node: F
Path found: Y -> X -> F
```

Simulated Annealing to Solve 8-Queens problem

Algorithm:

ug) I (UIIII).		
	Date//_		
	Page	180	No.
	Stimulated Annealing Algorithm		Page
			8 8)
	Sd current state: untial state		10:0
_2	choose ca withat temperature		1/2
3	Set Sest State = corred state		11/14/17
	set current Energy = evaluate (current state)		
	a many the same of the local of the same	60	1 2,5 1 2 1 23
1	while temp > 0 of interation (max - iteration	25	1 13 5
	for iteration = 1 to max Iteration do	3	1- 705 5 2/5 826
	new state = generate Neighbour Corrend state		\$ (w) d 2 - 6 1 2 (m) d
and a	new Energy = craduate (new state)		1 = (w)e 1 = (w)e
	energy difference = now Energy - Current Energy		Output [2(a)) [2(a)]
	if energy Pillimene 20 then		Enter un tral state: 50
	current stak = new state		united fraggreture: 20
	errent Every = new ewigg.		1 100 00 6 6 1 30 50 11
1	if current Energy & Sent Energy then		
-	Gest State = Corrent state		the no. of iderations 150 PBT PBT
	Sest Everyy = Correct Lorenzy		
1	Else		Revalor 1: Correct Stole = 49.0489
	1. Accept with a certain probability		Energy = 2401.7898 6 (M)
+	2 - Accept Prosobility = exp(- Enough Profession		1 2200
-	2. Accept Prosenting	5	2: 49.0481 : 2905.7898 = 5.00
-	3: If renson (v,1) & acceptance (no) last ty	0	3: 48.8662 ,= 2397.9069 = 2.5
-	then 111 to 2 new Hele	0	4: 48.8900 = 2351.2721=1.25
1	1. Current Steh = new Steh	:14	5.02 91 , = 2000922 = 0.625
-	2. (uring fining)		0.15
-	2. Correct Energy: her Energy. 11 1001 down temperature. temperature = temperature (ooling Roke		lest state : 48.0291 , Best Energy= 2306.70
1	temp interest - him	1	3
	Return Sest State.	13	
		80	
1		10/10	

Code:

import random import math

def energy(x):

return x ** 2 + 5 * math.sin(x) + math.exp(-x)

def adaptive_simulated_annealing(start, temp, cooling_rate, lower_limit, upper_limit):

```
current = start
                  current energy
= energy(current)
  while temp > 1:
    # Adaptive step size based on temperature (larger steps when hot)
step size = random.uniform(-1, 1) * temp
                                              new = current +
step_size
    # Ensure new solution is within bounds
if new < lower limit or new > upper limit:
       continue
    new energy = energy(new)
    # If the new spot is better, move there
if new energy < current energy:
current = new
                     current energy =
new energy
                 else:
       # Acceptance probability (explore worse spots)
probability = math.exp((current energy - new energy) / temp)
if random.uniform(0, 1) < probability:
         current = new
         current energy = new energy
    # Adaptive cooling based on progress
                                              if
abs(new_energy - current_energy) < 0.01:
temp *= 0.98 # Slow cooling near solution
else:
       temp *= cooling rate
  return current
```

Best solution found: -0.7323104061658242

Implement A* search algorithm for N queens

Algorithm:

Cote	Ente / / Propr
Traplementing i A+ secret Algorithm	Algorithm for & open way Hill Charling
I triticilize by placing the givens rensonally with our green per column along with highly providing quee of themselves. 2 Deques the node with the lowest of value from providing serve of node that you will have possible southern whom by thought or and licht south and with column with column with the providing the great the successions trate privarily where and ell to get inch privarily great to the successions trate privarily where and ell to get inch privarily great to the point start slake. 5 Coinner transfer & exploring social many with the point start slake.	Tribelization Start will a randown configuration where a gree 13 glaced in and isluming Therefoly Improve. (Comple the bearish: for the current year. The wrishe value is zero, the politico is found. 3 Greeneste reightouring states: For each green grantale the neighbouring likes by moving it to every possible row within its rotumn of calculate the heart with for each way state. A Choose the best reighbour with lorothermash. The neighbour sloke improves the current state of firminate. State of firminate.

Code:

import heapq

Helper function to calculate the heuristic (number of conflicts) def heuristic(board):

```
conflicts = 0 for i in range(len(board)):
                                                 for j in range(i
+ 1, len(board)):
                        if board[i] == board[j] or abs(board[i] -
board[j]) == j - i:
          conflicts += 1
return conflicts
# A* Search for 8-queens
def a star 8 queens():
n = 8 open set = []
  # Initial state: empty board
heapq.heappush(open_set, (0, [])) # (f, board)
  while open set:
                       f, board =
heapq.heappop(open set)
     # Goal check
                       if len(board) == n and
heuristic(board) == 0:
      return board
                        #
Generate successors
row = len(board)
                      for
col in range(n):
       new board = board + [col]
                                          if
heuristic(new board) == 0: # No conflicts so far
g = row + 1
                      h = heuristic(new board)
heapq.heappush(open set, (g + h, new board))
                                                 return
None # No solution found
# Run A* search solution = a_star_8_queens() print("Solution
board (column positions for each row):", solution)
```

Solution board (column positions for each row): [0, 4, 7, 5, 2, 6, 1, 3]

Implement Hill Climbing search algorithm to solve N-Queens problem

Algorithm:

agorium;	
	Date//_Page
Hill climb seach Olganithm	ach Rel and
ga in	1 1 1 1 2 2
Place one green in each column	Continued
Court the number of conflicts between	en two green
It herristic is D, solution is town	d with a labor
Storting with current solution and its co	attits.
generate neighbors by ming each rows in its colors	queen to different
select neighbor with least conflicts,	on a lowest heigh
If no reighter aproves the unest state	to the program
Jermindel.	and a Tolerand
	Land denta
	the college
Q	a Charle in
	,
Distribution of the contrale the	NA
Q	e sta
the Aller and the sample that	9 -2
tana a i	1.8
[5,2,0,7,3,1,6,4]	
I coming some all the control land	7 19
to dother tot this size all and	tale and
is and lest take a fell named dealine	Q .
and the state of t	41
1112 0 1112 1111 1111	7-19
the second secon	1.5707.7
the man agent to a many as the	hasa
taring named	sale
in given shall this in market to	
all proving that told growing the	N
t Other	19

Code:

import random

```
# Helper function to calculate the heuristic (number of conflicts)
def heuristic(board):
  conflicts = 0 for i in range(len(board)):
                                                 for j in range(i
+ 1, len(board)):
                        if board[i] == board[j] or abs(board[i] -
board[i] = i - i:
                           conflicts += 1 return conflicts
# Hill climbing for 8-queens def
hill climbing 8 queens():
= 8
  # Generate a random initial state board =
[random.randint(0, n - 1) for _ in range(n)]
  while True:
     current_h = heuristic(board)
if current h == 0:
       return board # Solution found
     # Find the best neighbor by moving each queen to every other column in its row
                           best h = current h
                                                   for row in range(n):
best board = board[:]
                                                                               for
                          if col == board[row]:
col in range(n):
                                                            continue
new board = board[:]
                                new board[row] = col
                                                                 new_h =
heuristic(new board)
          # If the new board has fewer conflicts, update the best board
if new h < best h:
                               best h = new h
best board = new board
     # If no improvement, we're stuck in a local minimum; restart
if best h \ge current h:
       board = [random.randint(0, n - 1) for in range(n)]
else:
       board = best board
```

```
# Run hill climbing search solution = hill_climbing_8_queens()
print("Solution board (column positions for each row):", solution)
```

```
Solution board (column positions for each row): [0, 6, 3, 5, 7, 1, 4, 2]
```

Create a knowledge base using propositional logic and show that the given query entails the knowledge base or not.

Algorithm:

Entailment using Herals	
Enfailment Usin)	Dote/
College of the Colleg	this means that if a pleasen has a children, they are considered sistings f(x) P(x) Pared (2,x). Pered (2,y)
Knowledge hars in of Sus	this means that it their children are sillings
	they are considered silling has a children,
0 10 11. 10 10101	P(n) P(y) Pared (2 n) 0 + 1 1
	(1,1) territ (2,4)
	· Alice is married to david
5 All parch have children	the does not directly affect the while
(0 = 4 = 4 0 = 1/2 = 1/2	relationship and may be related to the
1 Plue 13 morned to Parid.	(A) 100 + 100 + 11 (-12) Planette (12)
	Hypothesis
Hypothesis sittings of Bob	1. Ale 11 moth, of Bob
· charle is a sibings of Bub	2. But is father of charles 1/4 is
Ans Are 1 the mother of Box	5. (16) someon is a poised their children
Alice is parent of Box	l'awastings and, set is
M(Alice) -> P(Alice)	
M(Alice) = 1 (VIII)	So bos is the pount of charles of Alice is
	parent of Bub. The share porecided pink of sales
Bobis the John of charles	ging history touriban.
1605 13 10000	- logical Pervisional a sources (E)
F(505) -> P(1305)	M(MIU)-> P(MIU)
- Char	- (Bas) - Phas
At is a period	P(1 y)-s as lett or percels
But loing the fether of chariff the	b (charles, hos)
Ather opened But help of charles usepart	- A THE REST YAVE .
	- 11 & A.B. f(n) M(n) P(n) f(2) 3
Aru Sean, the mother of Got 1) also e pared M(n) -> P(n)	\$ A, 0, f(n), M(n), P(n) { (2, y) } ,
Ara Som, the visiting of 635 1) also e for	
M(n) -> P(n)	Alu - Buy as Lith Alue & but are
	10
All the gents have children	Pugint on sillen
Alice as a method of so, en a fait	Bob Fochalu S [So) & charle
All the gends have children Alice as a method of Sister of John Both hom children P(N) -> child (N+7)	
Maj - chair (147).	

```
Code: import
random
# Helper function to calculate the heuristic (number of conflicts)
def heuristic(board):
  conflicts = 0 for i in range(len(board)):
                                                 for j in range(i
+ 1, len(board)):
                        if board[i] == board[j] or abs(board[i] -
board[j]) == j - i:
          conflicts += 1
return conflicts
# Hill climbing for 8-queens def
hill climbing 8 queens():
= 8
  # Generate a random initial state
                                      board =
[random.randint(0, n - 1) for _ in range(n)]
  while True:
     current h = heuristic(board)
if current h == 0:
       return board # Solution found
     # Find the best neighbor by moving each queen to every other column in its row
best board = board[:]
                           best h = current h
                                                   for row in range(n):
                                                                                for
                          if col == board[row]:
col in range(n):
                                                            continue
new board = board[:]
                                new board[row] = col
          new_h = heuristic(new_board)
```

Solution board (column positions for each row): [6, 2, 7, 1, 4, 0, 5, 3]

Implement unification in first order logic

Algorithm:

Statement " It all mammals Streeth and It statement and Max is a dog then be	Date_/_I_
Statiment "If all mammale seem and	Poge
ar maint	(3) Conclusion
all	from the above
(1) For person to hon small small air	1 All mammels breethe air
1 (2) NII MO	2 All dogs are manuals
(Nu memmed , then n breath or)	3 Max 12 a 209
	Breethe Air (Max)
Sij All dogs our manuels	[Max Szeethes air]
4) (100(n)-) Hommolo (x))	E sel
1 1 203 , 100	
111 \$ Mar 13 = dog	
Max 13 c dos Dog (Max)	Max breethes Air.
The second section of the second	Max briefnis III
(9) Reasoning of Delivition	X.VX
(2) Reasoning 4 Deductions . Man 15 Manman	PAN
from ali) Vx (Doela) -> Mommels (2)) 4 (
(Max) we get Manmal (Max)	
· May breethy air	
from (8) 42 (Mammel(2) -> Brothe Air (4))4	
(May 15 momend, ell moments South All May	
	W t
and is not a damy	
[3] -2. 3 / 12 / 13	
days at the	

Code:

```
def unify(x, y, subst=None):
    """
    Unification Algorithm: Unifies two terms, X and Y.
    """ if subst is
None: subst
= {}
```

```
if x == y: # Step 1(a): If X and Y are identical
                                                        return subst
elif isinstance(x, str) and x.islower(): # Step 1(b): If X is a variable
return unify variable(x, y, subst)
  elif isinstance(y, str) and y.islower(): # Step 1(c): If Y is a variable
                                                                             return
unify variable(y, x, subst) elif isinstance(x, tuple) and isinstance(y, tuple): # Step 2:
Check predicates and arguments
                                       if x[0] != y[0] or len(x) != len(y): # Predicate symbol
or argument count mismatch
                                                        for x i, y i in zip(x[1:], y[1:]): # Step
                                      return None
5: Recurse through arguments
                                       subst = unify(x i, y i, subst)
                                                                              if subst is None:
return None
                                 else:
                  return subst
     return None # Step 1(d): Failure case
def unify variable(var, x, subst):
  *****
  Unify variable with another term.
  *****
         if var in
subst:
     return unify(subst[var], x, subst)
                                          elif
occurs check(var, x, subst): # Check if var occurs in x
return None
               else:
     subst[var] = x
return subst
def occurs check(var, x, subst):
  *****
  Check if a variable occurs in a term.
  111111
        if
var == x:
return True
elif
```

```
isinstance(x,
tuple):
    return any(occurs_check(var, xi, subst) for xi in x)
elif isinstance(x, str) and x in subst:
    return occurs_check(var, subst[x], subst)
return False

# Test cases for unification
x1 = ("P", "a", "x") y1 =
("P", "a", "b")

x2 = ("Q", "x", ("R", "x")) y2
= ("Q", "a", ("R", "a"))

print("Unifying", x1, "and", y1, "=>", unify(x1, y1))
print("Unifying", x2, "and", y2, "=>", unify(x2, y2))
```

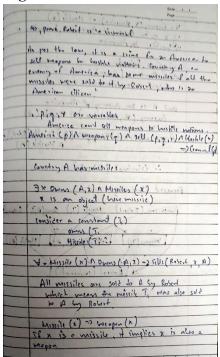
Output:

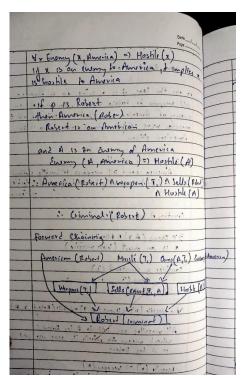
```
Unifying ('P', 'a', 'x') and ('P', 'a', 'b') => {'x': 'b'}
Unifying ('Q', 'x', ('R', 'x')) and ('Q', 'a', ('R', 'a')) => {'x': 'a'}
```

Program 9

Create a knowledge base consisting of first order logic statements and prove the given query using forward reasoning.

Algorithm:





Code:

Define the knowledge base (KB) as a set of facts

KB = set()

Premises based on the provided FOL problem

KB.add('American(Robert)')

KB.add('Enemy(America, A)')

KB.add('Missile(T1)')

KB.add('Owns(A, T1)')

```
# Define inference rules def
modus ponens(fact1, fact2, conclusion):
  """ Apply modus ponens inference rule: if fact1 and fact2 are true, then conclude conclusion """
if fact1 in KB and fact2 in KB:
                                    KB.add(conclusion)
                                                              print(f"Inferred: {conclusion}")
def forward chaining():
  """ Perform forward chaining to infer new facts until no more inferences can be made """
  # 1. Apply: Missile(x) \rightarrow Weapon(x)
if 'Missile(T1)' in KB:
KB.add('Weapon(T1)')
print(f"Inferred: Weapon(T1)")
  # 2. Apply: Sells(Robert, T1, A) from Owns(A, T1) and Weapon(T1)
if 'Owns(A, T1)' in KB and 'Weapon(T1)' in KB:
KB.add('Sells(Robert, T1, A)')
                                   print(f"Inferred: Sells(Robert, T1,
A)")
  # 3. Apply: Hostile(A) from Enemy(A, America)
if 'Enemy(America, A)' in KB:
KB.add('Hostile(A)')
                          print(f"Inferred:
Hostile(A)")
  # 4. Now, check if the goal is reached (i.e., if 'Criminal(Robert)' can be inferred)
'American(Robert)' in KB and 'Weapon(T1)' in KB and 'Sells(Robert, T1, A)' in KB and
'Hostile(A)' in KB:
     KB.add('Criminal(Robert)')
print("Inferred: Criminal(Robert)")
  # Check if we've reached our goal
  if 'Criminal(Robert)' in KB:
print("Robert is a criminal!")
                                else:
```

print("No more inferences can be made.")

Run forward chaining to attempt to derive the conclusion forward_chaining()

Output:

```
Inferred: Weapon(T1)
Inferred: Sells(Robert, T1, A)
Inferred: Hostile(A)
Inferred: Criminal(Robert)
Robert is a crimina<u>l</u>!
```

Program 10

Implement Min-Max Algorithm for Tic Tac Toe

Algorithm:

	Dote		
	(1x les treed) evenerary		
	White Trillight Grand & rought		
	The Tac Toe Using Min Max down come		
	Deline 3×3 grid where tack all and 1.		
	r. plegit 1		
	O: player 2 sented but 15		
	-10: Marmize wingsdor had -10: Minimize wings 0: Draw of Wingsgood does not		
1	del minimax (Soard, death 1) maximize,)		
	Score = evallete or (board)		
	Vest o sworm ford		
	with store = File briston = = -10		
	refurn stored that arother		
	of 1) board full (board)		
1)	see return o ast		
/	10 10 10		
	AT's more : If is Meximiar:		
	best= - infinity		
	for each empty cell:		
	To Lond Till = 1x1 14		
	Lest max (Sest , Williams (Social Leght)		
	V V - 0 X X - 0 0 7 + 0x (bi)		
	book [cell]= empty		
-	return best		
-	mes els: T4 year		
	Oppound: Sest ? + infinity 10 0		
-	for each empty cell		
	1 10 10 0 17 10		
11	0 (1 1)		

Date ,				
makemore (board (cell, 'x'))				
Sion = inimmer (Sourd, Light +1, Jely)				
undo-more (Social cell)				
bosision = min (bosision - done)				
notion bestscore sin Ell and				
T world (A.)				
def find Bestmon (Socret):				
bustmore = None				
for each emply sell in Spar?				
				make more (board, (ell, 'x')
				more relie = innumer (Social O. folso)
Mote value = innumar (Social, 0, telse)				
best more = cell				
host value more value				
return best more worth				
(Bleed Whited is);				
User AT User				
0 0 0 0 1				
X X X				
1 1 single diel 0				
: 1) of the last of 1				
TA 's THU, ITA				
0				
·XXXDCXXDCXX				
9141 .01.11				
UNT AT 1110 UNT				
0 101 10101× 0101×				
A X O D X X O				
DIX DIX				
77 13 a drawl				

Code:

import math

Constants for the players

AI = 'X'

HUMAN = 'O'

EMPTY = '_'

```
# Function to print the board
def print board(board):
for row in board:
     print(" ".join(row))
print()
# Function to check if a player has won
def check winner(board, player):
Check rows, columns, and diagonals
for row in board:
                      if all(cell ==
player for cell in row):
       return True
                     for col in range(3):
                                              if all(row[col] == player for row in board):
              if all(board[i][i] == player for i in range(3)) or all(board[i][2 - i] == player for i
return True
in range(3):
     return True
return False
# Function to check if the game is a draw def
is_draw(board):
  return all(cell != EMPTY for row in board for cell in row)
# Minimax algorithm def minimax(board,
depth, is_maximizing):
                        if
check winner(board, AI):
     return 10 - depth if
check winner(board, HUMAN):
     return depth - 10
if is draw(board):
     return 0
  if is maximizing:
```

```
best score = -math.inf
for i in range(3):
                        for j in
                   if board[i][j] ==
range(3):
EMPTY:
            board[i][j] = AI
                                        score =
minimax(board, depth + 1, False)
board[i][j] = EMPTY
                                 best score =
max(best score, score)
                            return best score
                                               else:
     best score = math.inf
                               for i in range(3):
for j in range(3):
                          if board[i][j] == EMPTY:
board[i][j] = HUMAN
                                  score =
minimax(board, depth + 1, True)
board[i][j] = EMPTY
                                 best score =
min(best score, score)
                           return best score
# Function to find the best move for AI
def find best move(board):
best score = -math.inf move = (-1, -1)
1) for i in range(3):
                          for j in
                 if board[i][j] ==
range(3):
EMPTY:
         board[i][j] = AI
                                   score
= minimax(board, 0, False)
board[i][j] = EMPTY
                               if score >
best score:
                        best score = score
            move = (i, j)
return move
# Example usage if
name == " main ":
# Initialize a sample board
board = [
```

```
['X', 'O', 'X'],
    ['O', 'X', 'O'],
    ['_', '_', '_']
  print("Current Board:")
print board(board)
  best move = find best move(board)
print(f"The best move for AI is: {best_move}")
Output:
```

```
11
Player O's turn.
Enter the row (0, 1, 2): 1
Enter the column (0, 1, 2): 1
X | 1
| 0 |
11
Player X's turn.
Enter the row (0, 1, 2): 0
Enter the column (0, 1, 2): 2
X| |X
 1.1
Player 0's turn.
Enter the row (0, 1, 2): 0
Enter the column (0, 1, 2): 1
X|0|X
101
  101
Player 0's turn.
Enter the row (0, 1, 2): 2
Enter the column (0, 1, 2): 1
X|0|X
Player O wins!
```

Implement Alpha-Beta Pruning for 8 queens

Algorithm:

TANA - P. A. O.	
Apha Bola for 8: grans	Page
Also	
	min - score : win (min - som stephen side (so
det 11-Nolud (Soaid now (al))	You+1, dyha, seta, Ame
for I to some (o)	Good [row]=-1
for 1 in sould (sou); 1.	5 cold from 5 = 1 5 cta = mm (Sele imaxione) 1 f Sele < = expha Grak
also (i-20)	of listeries alpha
return dels	brick
return true () move 3 who fit	return byte cost
	glob social
Jet alpho - Seta (Society row, alpha, beta, Ismax):	de solve 8 quem () " maire
if you has ling board). still	1 Loan 2 = [-1] +0
return apple between close to says on the	Loane = fell ("int)
if is mori	return alphe beto (Societ no, alphe, but of
water gove = 0	1 4-90 61 11
for col in rease (learboard) is and	0 : Ne 2 Young
If invalid (Social row, 101):	Organ (hotel more and a
Larz [row] = (a) do film row t	
max serve += alpha - scha (book, row + alpha sto, film)	· O · lear were const
Secret (row ord)	10 de
alpha = nox (apha, max=score)	1, 00 ack
alpho = hox (april)	agrana mile feat " cigle
o Greak	0 . 0.10./33 0000 41
return was store	
Yeturn Wax ter	· · · · · · · · · · · · · · · · · · ·
musion = float ('sint)	0
minister = float (minister): for (a) in range (Im (board)): for (a) in stange (Im (board)):	(the street section of set
	The most sensed three me
40ar2 (10L)=10)	let level and

Code:

return False

return True

```
def alpha beta search(self, board, col, alpha, beta, maximizing player):
     """Alpha-Beta Pruning Search."""
                                          if col >= self.size: # If all queens are placed
return 0, [row[:] for row in board] # Return 0 as heuristic since it's a valid solution
    if maximizing player:
max eval = float('-inf')
best board = None
                           for row in
range(self.size):
                          if
self.is safe(board, row, col):
            board[row][col] = 1
            eval score, potential board = self.alpha beta search(board, col + 1, alpha, beta, False)
board[row][col] = 0
                                if eval score > max eval:
                                                                          max eval = eval score
best board = potential board
                                          alpha = max(alpha, eval score)
                                                                                      if beta <=
alpha: # Beta cutoff
                                   break
       return max eval, best board
else:
       min eval = float('inf')
       best board = None
                                  for
                                 if
row in range(self.size):
self.is safe(board, row, col):
            board[row][col] = 1
                                             eval score, potential board =
self.alpha beta search(board, col + 1, alpha, beta, True)
                                                                     board[row][col] = 0
if eval score < min eval:
                                        min eval = eval score
                                                                               best board =
potential_board
                                                                      if beta <= alpha: # Alpha
                            beta = min(beta, eval score)
cutoff
                     break
       return min eval, best board
  def solve(self):
```

```
"""Solve the 8-Queens problem."""
                                              board
= [[0] * self.size for _ in range(self.size)]
     _, solution = self.alpha_beta_search(board, 0, float('-inf'), float('inf'), True)
return solution
  def print_board(self, board):
"""Print the chessboard."""
                                for
row in board:
       print(" ".join("Q" if col else "." for col in row))
print()
if name == " main ":
game = EightQueens()
solution = game.solve()
                          if
solution:
     print("Solution found:")
     game.print board(solution)
else:
     print("No solution exists.")
```

Output:

