B.M.S. COLLEGE OF ENGINEERING BENGALURU Autonomous Institute, Affiliated to VTU



Lab Record Artificial Intelligence (22CS5PCAIN)

Submitted in partial fulfillment for the 6th Semester Laboratory

Bachelor of Technology

in

Computer Science and Engineering

Submitted by:

Name: Shivani Sathyanarayanan USN: 1BM21CS203

Department of Computer Science and Engineering B.M.S. College of Engineering Bull Temple Road, Basavanagudi, Bangalore 560 019 Mar-June 2021

B.M.S. COLLEGE OF ENGINEERING DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



CERTIFICATE

This is to certify that the Artificial Intelligence (20CS5PCAIP) laboratory has been carried out by Shivani Sathyanarayanan (1BM21CS203) during the 5th Semester September-January 2021.

Signature of the Faculty Incharge:

Name: Dr. Pallavi G B

Designation: Assistant Professor

Department of Computer Science and Engineering

B.M.S. College of Engineering, Bangalore

Table of Contents

Sl. No.	Title	Page No.
1.	Tic Tac Toe	4-11
2.	8 Puzzle Breadth First Search Algorithm	12-16
3.	8 Puzzle Iterative Deepening Search Algorithm	17-20
4.	8 Puzzle A* Search Algorithm	21-27
5.	Vacuum Cleaner	28-35
6.	Knowledge Base Entailment	36-42
7.	Knowledge Base Resolution	43-49
8.	Unification	50-54
9.	FOL to CNF	55-61
10.	Forward reasoning	62-68

1. Tic Tac Toe

```
LAB - I TICTAC TOF
import random
            for -in range (10)]
 def insurtetus ( letter, pos):
  global board
  board [pos] = lette
def space is Fred pos):
  return board [pos] == " "
def print-Board (board):
    # print ('11')
    print ( ' + board [1] + 1 t board [2] + 21 + board [3])
             " + board [4] + 1' + board [55] + 1' + board [6])
    print (' ---- ')
   #prin+('11')
    print (' + board [7]+ 1'+ board [8] + 1'+ board [9])
del is Winner ( 60, 6):
      bo[87] == 4 and bo[8] == 4 and bo[9] == 4)or
      bo[4] = = 4 and bo[5] = : 6 and bo[6] = 4) or
     ( 60 [1] = : 6 and 60 [2] = : 4 and 50 [3] = 6) or
     ( bo[ 1] = 6 and bo[4] = 6 and bo[7] = 6) or
```

```
50[8] = : 4) or
 (60[2] == le and 60[5] == le
                                     60 [9] == 6) or
 ( bo [3] == 6
               and bo[6] = = Wand
                                      60[9] = · (1) 5T
 ( 60[1] == 4 and 60[5] == 4
                                      bo [7] == (4) three the
  (bo[3] == 6 and bo[5] == 6
deffloyer Have ():
   prood board
    run = Tru
     while run:
       (man) tri= man
       i) $ 1 <= move <= 9:
           if space Istra (move):
               insurtetter ('X', man)
         print ( borry, this space is occupied! )
           print ( Phase type a number within the range! ?)
     except & Valuerror:
        print ( Please type a Number! ?)
dy compton():
   global board
   possible Hours: Ex for x, why in enumerate (bogsed) is
                    little == " and x!=0]
  for let in ['o' 'x']:
    for i in possible Moves.
       board Copy - Board [:]
       brand copy [i] tet
       if is winner (boardapy, w):
```

- return i (Laid Add ma 1/0 pros	a J Seep
DAMES C	44599
corners agen = [1 for i in possible Heren if i in]	1,3,7,9]
i) cornersoper:	wilded that fi
return selectRandom (corners Open)	and it has been
if 5 in possible Moves	St. John S.
neturn 5 (Tenned 1)?	
t a few second and a second and	cale
edges Open = [i for i in possible Hower of icin [274	1500
lif edge open - 40 m 10 10 more of aligned	None.
return select Random (edges Open) (brand I beam	d some
	7.1.
return None and build and still some a ??"	100
Please tolers a pullban to alore parties (and) is	Naghy) if
dif select Rardom (li):	
in : len(li)	Chilemed of
	and I taken
retur ([r]	
No. 1 de la companya del companya del la companya del companya de la companya de la companya de la companya del companya de la companya del companya del companya del la companya del com	I well with
def is Board full 1 1 1 man may at my on !	Sugar Sugar
return board court () (= 1 comes or)	
Those total a serious to feel several (and	The broad
dy main ():	- 1 Holya
global board	Dalon
print (Welcom to Tic Tac Toe!)	
	1,13
print Board (board)	Very
KIB / B	
while not is Boardfull (board):	-050.f3
: (O , board) transition is	/ Page
playerMark()	
print Board (board)	
else:	12 11 12
Total I I I	store
	67

print ('barry, O's won this t	in !')
break	And the second second
to The + is it is it to provide the	Miller I - and raway
if not is Winner (board, 'x'):	AND MANYOU !
more = compMore() (myours	rail probability and ic
i more is the Non:	Propartition of the
print ('Tie yame!')	हे विस्मित
class	
insurt Letter ('O' move)	edge Open - I i for I in pour
print (Computer placed an	('O' in position', more 'i)
print Board (board) Conglet	to I reakon & to also amber
elsi:	
print ('x /'s won this his	m! good Job!) notes make
break	
Market Company	Cill makes & early Joh
if is boardfull (board):	(is) as I (is)
print (Tie yam!)	all spreading maker r
Total Control of the	CrJU needer
, hil fru:	ASSESS THE Esting TO
if answer lower () == 'y' or answer.	y again? (YIN)) bear of the
il answer lower () == 'y' or answer.	town () = a yes and and
0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,	
poorg: [Par in rough [10]	
print ()	C) over the
print ('') main ()	front longing
main ()	front longing
main ()	road todale
main ()	front longing
print (' · · ·) main () else: break	pol of thousand the Comp
print (· · · · · · ·) main () else: break anaira	prood towning
print (· · · · · · ·) main () else: break anain print (· · · · · · · ·) else: break	prood towning of broad broad any
print (' · · ·) main () else: break anain oin ()	and the said the said of the s
print (' · · ·) main () dy: break anain oin ()	prood towning Throad towning Throad throad throad Throad throad Throad throad Throad throad Throad throad Throad throad Throad throad
print ('') main() dy: break anain oin()	propert towning Throad towning Throad throad throad Throad throad throad Throad throad throad Throad throad Throad throad Throad throad Throad throad Throad throad Throad throad T

	MODAY S DAL
Output:	Menor A. N. othershoother V. v. venste
X 1 10	and complement of the 10 kings married
1-1	the official of A'd was long
and the second of the	Britaning Adams to the Territory A. " Go June
A.V.	ition to place an 'X' (1.9):5
	1 18 miles and with 31 Thought - Helpai me total
	atalon lague begut C Cates status of met
The second secon	ols also I sugar turnelane tugni calata
	n D'in position 9 : 1 - 1 - 1 - 1 - 1 - 1
X1 10	Progress to the long brown I a see
1 1 1	"A" turning the all the
	A sectional in the place of the organization of the
	ition to place on 'x' (1-9):6001
	In Cart 18 rack appropriate Tring
	o'= [A] asts load
	1 1 = 6 1/20 °
	"O" and position 4: 447 1) they
X 1 1 0 . banks	I wild said A moderns I I doing !
olxlx	
	" (I de drawitgmas dugaje ratore vije i wood
	sition to place an 1x (1-9):8
	3 January private) tolog and i
	of Carena a har to a marine
1/x10 man sun	19 minum roj test) tolog
Come til about	an 'O' in position 2:
XIOIO	1-1 308
1 × 1 × 400 1	was in water to the same of
TALA	and sed a manual) tarra
1 Kill O Exault A	1100 E 100 00 1/1-01 37
Please relief a p	osition to place as x'(1-9):37
X 1010	"Amalada a "Imaldan ata") dalah
OIXIX	Adapted and an inchested a selection of
The your.	

```
import random
# Initialize the game board board = [' ' for in range(10)]
def insertLetter(letter, pos): global board
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] # print(' | |')
def isWinner(bo, le): return (
(bo[7] = le \text{ and } (bo[4] == le \text{ and } (bo[1] == le \text{ and } (bo[1] == le \text{ and } (bo[2] == le \text{ and } (bo[3] == le \text{ and }
= le and (bo[1] = le and (bo[3] = le and
def playerMove(): global board run = True while run:
+'|'+ board[8]+'|'+ board[9])
)
bo[8] == le and bo[9] == le) or bo[5] == le and bo[6] == le) or bo[2] == le and bo[3] == le)
or bo[4] == le and bo[7] == le) or bo[5] == le and bo[8] == le) or bo[6] == le and bo[9] == le
le) or bo[5] == le and bo[9] == le) or bo[5] == le and bo[7] == le)
move = input('Please select a position to place an \X\'(1-9): ') try:
move = int(move) if 1 \le move \le 9:
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 ValueError:
print('Please type a number!')
```

```
def compMove():
global board
possibleMoves = [x \text{ for } x, \text{ letter in enumerate(board) if letter} == '' \text{ and } x != 0]
for let in ['O', 'X']:
for i in possibleMoves:
boardCopy = board[:] boardCopy[i] = let
if isWinner(boardCopy, let):
return i
cornersOpen = [i for i in possibleMoves if i in [1, 3, 7, 9]] if cornersOpen:
return selectRandom(cornersOpen)
if 5 in possibleMoves: return 5
edgesOpen = [i \text{ for } i \text{ in possibleMoves if } i \text{ in } [2, 4, 6, 8]] \text{ if edgesOpen:}
return selectRandom(edgesOpen) return None # Indicates a tie
def selectRandom(li):
ln = len(li)
r = random.randrange(ln) return li[r]
def isBoardFull(board): return board.count(' ') <= 1
def main():
global board
print('Welcome to Tic Tac Toe!') printBoard(board)
while not isBoardFull(board): if not isWinner(board, 'O'):
playerMove()
printBoard(board) else:
print('Sorry, O\'s won this time!') break
if not isWinner(board, 'X'): move = compMove()
if move is None:
print('Tie Game!') else:
insertLetter('O', move)
print('Computer placed an \'O\' in position', move, ':') printBoard(board)
else:
print('X\'s won this time! Good Job!') break
```

```
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 _ in range(10)] print('-----') main()
else: break
# Run the game
main()
```

computer placed an 'O' in position 3 : x | | 0 Please select a position to place an 'X' (1-9): 5 x | | 0 1 X 1 Computer placed an 'O' in position 9 : 1 × 1 Please select a position to place an 'X' (1-9): 6 X | | 0 | X | X Computer placed an 'O' in position 4 : x | | 0 0 | X | X 1 10 Please select a position to place an 'X' (1-9): 8 x 1 1 0 0 | X | X | X | O Computer placed an '0' in position 2 : $x \mid o \mid o$ 0 | X | X | X | O Please select a position to place an 'X' (1-9): 7 X | 0 | 0 0 | x | x x | x | o

2. 8 Puzzle Breadth First Search Algorithm

100.0	8 PUZZLE PROBLEM
LAB 3 -	8 POZZEE PROBLEMI
del Markey and Sold	A world a content was
act of core target	OA A What son
quin · []	- I mora work a whole with
quine append (src)	1 8 A 3 nothing of head world
[] gas	Anatoral in buildy of mounts
OKP - 1	Section Property Co.
while lon (queen)>0	a stronger where and sellered
source = open. poplo?	
exp. append (source)	a many it at 1931 mould
forester to make a started	1 1793 persona and trees
print (source)	LOST IN SOCK 13 July 1, 1501
	bruill and the A burned
if 'source == target	STATE HAU-)
print (" sucuss")) 10/0000
return	S. 1 trus durated to the 1 2
poss_moves_to_do	- C3 -
poss - mores - to-do	= possible_moves (source, exp)
Partie America in contrast, culture	
for neve in poss-	moves_to_do:
il move not in	exp and more not in agreen:
gum apper	d (more)
Cont. Bro. C. Harrista D. I.	
del possible mover (states, u	nisited states).
b = state. irolin (-1)	
2=[]	N - 10 - 10 - 10 - 10 - 10 - 10 - 10 - 1
il b not in [0,1,2]:	
d. append ('u')	
il b not in [6,7,8]:	

```
if b not in [0, 3, 6]:
          de. append ('1')
       il b not in [2,5,8]
         d. append ('v')
       pos-moves-it-can=[]
       for i in al!
        pos. noves_it.con. append (gen (state, i, b))
       return [ more_it_can for more_it_can in pos. mover_it_can
              more-it-can not in vigited states
  dy gen ( State, m, b):
    temp: state copy ()
     il & n = : 'd' :
          temp[b+3], temp[b] = temp[b], temp[b+3]
           temp [6.3], temp [6] = temp [6], temp [6.3]
           temp [b-1], temp [b] = temp[b], temp [b-1]
          temp[b+1], temp[b]: temp[b], temp[b+1]
       return temp
Arc = [2, -1, 3, 1, 8, 4, 7, 6, 5]
target - [1,2,3,8,-1,4,7,6,5]
bli ( prc, target)
```

OUTPUT !	1 to off a constant of fi
	C. L. Dharge it
[2-1,3,1,8,4,7,6,5	
[2831,-1,476,5	3
[-123184765	
[2,3,-1,1,8,4,7,6,5	7
[2831647-1	Proj.
[2,8,3,-1,1,4,7,6]	7
	Sal brigge nes of amount in
[1,23-18476	
	5 Jun ref no 11 man I reider
[28,3,16,4-17	S. J. of the not it some
[2,8,3,6,4,7,5	
[-1,8,32,1476	
[2,8,3,7,1,4-1,	65 Oppose date mil
[2,8,-1,1,4,3,7,6	.5]
[a, 8, 3, 1, 4, 5, 7, 6	
[1, 2, 3, 7, 8, 4, -1,	Carp Test of Cont Tomat
[1,2,3,8,-1,4,7	[-2,3,
Band Jane Palgon	O = Palyo Dead yet
	lavi 123
Detal and Odfamil	
A Proof Ash as	191: 101
[[] [] Justing Edding	Eddgad, Cod Jane
all part life expert fifther a	enture treat
and sould shall be a	[2 J.C. H. 8 . I. C. I. J.] - 25.
atolia	[2 d c 1 1 3 8 6 1 7 here
The season of the season of the	Chapter, and Vist
The second of the same)
of the second state of the	
of report California	
	store 67

```
def bfs(src,target):
    queue = []
    queue.append(src)
exp = []
    while len(queue) > 0:
        source = queue.pop(0)
        exp.append(source)
        print(source)
        if source==target:
            print("success")
            return
poss_moves_to_do = []
poss moves to do = possible moves(source,exp)
for move in poss_moves_to_do:
if move not in exp and move not in queue: queue.append(move)
def possible moves(state, visited states): #index of empty spot
b = state.index(-1)
    #directions array
d = []
#Add all the possible directions
    if b not in [0,1,2]:
        d.append('u')
    if b not in [6,7,8]:
        d.append('d')
    if b not in [0,3,6]:
        d.append('1')
    if b not in [2,5,8]:
        d.append('r')
# If direction is possible then add state to move
pos moves it can = []
# for all possible directions find the state if that move is played
### Jump to gen function to generate all possible moves in the given
directions
for i in d: pos_moves_it_can.append(gen(state,i,b))
    return [move it can for
in visited_states]
def gen(state, m, b):
```

```
temp = state.copy()
    if m=='d':
        temp[b+3], temp[b] =
    if m=='u':
        temp[b-3], temp[b] =
    if m=='1':
        temp[b-1], temp[b] =
    if m=='r':
        temp[b+1], temp[b] =
    # return new state with
return temp
src = [2,-1,3,1,8,4,7,6,5] target=[1,2,3,8,-1,4,7,6,5] bfs(src, target)
move_it_can in pos_moves_it_can if move_it_can not
temp[b],temp[b+3]
temp[b],temp[b-3]
temp[b],temp[b-1]
temp[b],temp[b+1]
tested move to later check if "src == target"
```

```
[2, -1, 3, 1, 8, 4, 7, 6, 5]
[2, 8, 3, 1, -1, 4, 7, 6, 5]
[-1, 2, 3, 1, 8, 4, 7, 6, 5]
[2, 3, -1, 1, 8, 4, 7, 6, 5]
[2, 8, 3, 1, 6, 4, 7, -1, 5]
[2, 8, 3, -1, 1, 4, 7, 6, 5]
[2, 8, 3, 1, 4, -1, 7, 6, 5]
[1, 2, 3, -1, 8, 4, 7, 6, 5]
[2, 3, 4, 1, 8, -1, 7, 6, 5]
[2, 8, 3, 1, 6, 4, -1, 7, 5]
[2, 8, 3, 1, 6, 4, 7, 5, -1]
[-1, 8, 3, 2, 1, 4, 7, 6, 5]
[2, 8, 3, 7, 1, 4, -1, 6, 5]
[2, 8, 3, 1, 4, 5, 7, 6, -1]
[1, 2, 3, 7, 8, 4, -1, 6, 5]
[1, 2, 3, 8, -1, 4, 7, 6, 5]
success
```

3. 8 Puzzle Iterative Deepening Search Algorithm

LAB-4 8 PUZZL	E PROBLEM BOMES
day PuzzuNode: 11 stole	of hour Sugar Stranger modification
	· Now, action : None): June water
sey. State = state	
0	of short bear build about for
self. achion achion	(state clan) trop is fi
4	and finise
def get-path (self):	they have been seen
path - []	relate Laber
current = self	ndr
	as the of deleted tool for ref
	, current action))
	(45) Andre General Algebra (F
return path [1:-1]	will arestor
	with & miller
lef is goal (state):	als had a second
	(, , ,)
neturn state == goal state	u.o.c.s.t) their ledist
ð	1 - Final Algeli
of get reighboons (state)	
nighbors []) also love	I was word tiget white
empty-index: state index (0)	Wysb .
now, col - dismodt employ-inde	x, 3) (there I him
	and the state of t
for now in L(0,1), (1,0)	, (o, -1) (-1, o):
new-row, new-col > you +	
11 0 < - nw_row < 3 an	
neighbor_stale = list(State)
anially lades a due of	South to 2 1 Alex and
nuchbor-state [emoty	indus], reighbor-state [neighbor- te [neighbor-index], reighbor-state
index] - (neighbor - Ala)	to Enughborindes T neighbor what
	of store

	4 2
[complyindex]	9335012 F 3A
neighbors. append (tuple (neigh	Abordate) the first the fit
return neighbors	a taking data that the little
1-1-2 5 1 R H T I	dola dola Bes
def depth - limited worth (rode	
if is-good (rode state):	nedeconolie plea
return True	
elif depth-limit == 0:	· (just stay deg det
	Ed- Non
else:	Ha - Januar
	104 of . A. 100 December 1
for Nichous state in get i	righbors (node state) in white
child - Puzzknedi (nu	ghbor state, node), 790
if depth_ Divited_search (child, goal state, depth-limit -1):
secure 17m	[1] Hey nested
return & false	
L R R A R R R R R R R R R R R R R R R R	Carles Loop 21 Jet
1 name == " main "! "	Y. D. Z. S. J. E. S. J. Dela Loop
initial_state = (1, 2, 3, 0,	4,5,6,7,8)
depth-limit = 1	0
	(initial_state) (state)
result = depth_limited_scarce	h (initial node, (1,2,3,6,4,5,0,7,8
	upph whit) I sale my so is
print-(result)	water was opened bought water
	٢,
OUT PUT: .: Ea -) (1 e	12 (4.1) (6) J. o. van y
	AN T GOOD TO THE TOTAL TO THE
	KAD E > WAY SAN > O II
Pour Care	
	dalded only and the
The state of the P	cor sun subst sellen
malgin I sole would no Print	of to done done to gather
Bets indepen fisher milion	Tark redding) - [Vilia
	store

```
class PuzzleNode:
def init (self, state, parent=None, action=None):
        self.state = state
        self.parent = parent
        self.action = action
def get path(self): # Not required path = []
        current = self
        while current:
path.append((current.state, current.action))
            current = current.parent
        return path[::-1]
def is goal(state):
goal_state = (1,2,3,6,4,5,0,7,8) return state == goal_state
def get neighbors(state): neighbors = []
empty_index = state.index(0) row, col = divmod(empty_index, 3)
for move in [(0, 1), (1, 0), (0, -1), (-1, 0)]: new_row, new_col = row +
move[0], col + move[1] if 0 <= new row < 3 and 0 <= new col < 3:
neighbor_state = list(state)
neighbor index = new_row * 3 + new_col neighbor_state[empty_index],
neighbor_state[neighbor_index] = (
neighbor state[neighbor index],
neighbor_state[empty_index], )
neighbors.append(tuple(neighbor state)) return neighbors
def depth_limited_search(node, goal_state, depth_limit): if
is goal(node.state):
        return True
    elif depth limit == 0:
        return False
    else:
for neighbor state in get_neighbors(node.state):
child = PuzzleNode(neighbor state, node)
if depth_limited_search(child, goal_state, depth_limit - 1):
return True
       return False
```

```
if __name__ == "__main__":
initial_state = (1, 2, 3, 0, 4, 5, 6, 7, 8)
depth_limit = 1 # Set the depth limit as needed
initial_node = PuzzleNode(initial_state)
result = depth_limited_search(initial_node, (1,2,3,6,4,5,0,7,8),

depth_limit)
    print(result)
```

```
dfs.py
True
```

4. 8 Puzzle A* Search Algorithm

Ten I in months 420 acqu	, Nidas
V* Vidaithu : 100) the first top the	(44)
importational Ex 2 07 Fr e 13 Too date their energy	17
the same block (that, not substituted and a	
class Node:	
dif winit - (self, dato, level, frol) like the liver	
self, data = data	
sily live (coloneland) With there is blid	v v
I such beginned super both the pay you had	
The state of the s	att Andre Ver
def generate child (self):	
x, y = sey. find (self. data, ')	ATT TO
Val-list . [[x,y-1], [x, y+1], [x-1,y], [x	
children = [][8 = F7, F3 N 0] [2, E]] = De	A today
for i in val-list:	1.00
child - self shifte (self data, x, y, if o	1,1643
if child is not None:	Δ
child-node: Node (child, self, level +1)	0)
children append (child node)	Talsey
regetus children (":440") ?	2
the literaturalists about northless of more	
def shuffle (self, puz, x1, y1, x2, y2):	alieta La Ta
if n2>=0 and n2< len(self.data) and y2>=	> = O and y 2 < lin (
and but the man the second	1 dairy
temp = temp-pux [x2][y2]	
temp-pux [a2][y2] = temp-puz [21][y1]	Pagent
temp-puz [x1][y1] = temp	
setten temp-puz	a re-restricted
else:	Post !
return None	[2 3]
	(1 7 3)
	(0 8 store)

	96 A
-	dy copy (seef , root) was also more yes I coming fet
-	temp [] To a dex hotel well that
3	for i in root: (old lange water) the conte
3	1 2 F: [] Chell) knyge naga 12:0
3	for jin 1: Catal Strang
7	t. append (j)
7	temp append (t)
3	return temp to I regarded and
3	
3	dy pod (sey, puz, x); ("o/") tring
	for i in range (0, Un (self. data)):
3	for j in range (o, Un (self, data)):
3	a ouz (i) (i) := x :
3	return i j
3	10 = (clab long Nobel out I d. Hel J:
3	class Puzzle:
3	dy -init- (sey, size) is downg on it in
3	self n : size (odst lage i) [flex son) i
3	sel oper :[] (1) brigge reger flor
3	self closed = [] & where I brigger break flexingered in
3	Mr. Makagat Hes Mebilia pulm
8	dy (sul, stort, god): (1)
	return self. h (start. data, goal) + start level
	The state of At I francis on the field 17] to state more
	de a (self share , goal)
	Tenas = 0
•	for i in range (o, self.n): (2) 2) = ent
9	for in range (o) self in) state dead is herevery and
•	if start [i][j]! = goal[i][j] and start [i][j]!
9	
3	temp +=1
	record comp
	shre to
	67

dy process (self, start data, goal data): start - Node (start-data, 0,0) start : self - { (start good data) self. open append (start) print- (" 1010") while Irue: aux = self. open [0] print ("10") for i in our data: un) for j in i wet you and print (j, end ") = [][] self. h (cur. data to good_data) == 0: for i in cur generate child () ... Held i (val self (i, goal data) self. open append (i) self closed append (cur) Lely, open (0) self open sort (key . lambda x x frat neverse false Start state - [[1,2,3], [: '4', 6'] ['T',5" puz - Puzzle(3) 2. process (start state, goal state)

	$\lambda = 3.19 M_{\odot}$
OUTFUT !	TOU OF SUPPLIED THAT THE SUPPLIED SUPPL
	Contribute Daniery) long thereton y
123	The adultion
- 46	Fresh and the state of state of the
7 2 8	I were a state of the
	Lestert in function of many of the
123	Swell of best to downing to
4 _ 6	English or what a room of the
758	I wish to med a court of
	I went of work of work 193
1 2 3	Liver is well to well of 3
456	
7 - 8	work alietas
	(toleban is leban ral
1 2 2 3 vale	
4 5 6	Crepan noutrons 1
78-	20 12
	# Nord
Dias Consonit 6	
Time Complexity 6	of BFS: worst case, it can be expressed as
Time Complexity 6	of BF5: Worst case, it can be expressed as $O(b^d)$ where b is branching factor
Time Complexity &	of BF5: Worst case, it can be expressed as $O(b^d)$ where b is branching factor
	O(bd) where b is branching factor and d is depth of solution.
Time Complexity of	O(bd) where b is branching factor and of 13 depth of solution. At : Depends on houristic, if hereight is
	O(bd) where b is branching factor and of 13 depth of solution. At : Depends on houristic, if hereight is
	O(bd) where b is branching factor and d is depth of solution.
	O(bd) where b is branching factor and of 13 depth of solution. At : Depends on houristic, if hereight is
	O(bd) where b is branching factor and d is depth of solution. At : Depends on houristic, if houristic is perfect, then it is O(d) and worst case O(bd)
ine Complexity of	O(bd) where b is branching factor and of 13 depth of solution. At : Depends on huristic, if huristic is perfect, then it is O(d) and worst case O(bd)
ine Complexity of	O(bd) where b is branching factor and of 13 depth of solution. At : Depends on huristic, if huristic is perfect, then it is O(d) and worst case O(bd)
liber of Cheen	O(bd) where b is branching factor and of 19 depth of solution. At : Depends on huristic, if huristic is perfect, then it is O(d) and worst case O(bd)
liber of Cheen	O(bd) where b is branching factor and of 13 depth of solution. At : Depends on humistic, if humistic is perfect, then it is O(d) and worst case O(bd)
liber of Cheen	O(bd) where b is branching factor and of 13 depth of solution. At : Depends on huristic, if huristic is perfect, then it is O(d) and worst case O(bd)

```
import heapq
class Node:
def _ init (self, data, level, fval):
self.data = data self.level = level self.fval = fval
def generate child(self):
x, y = self.find(self.data, '_')
val list = [[x, y-1], [x, y+1], [x-1, y], [x+1, y]] children = []
for i in val list:
child = self.shuffle(self.data, x, y, i[0], i[1]) if child is not None:
child node = Node(child, self.level+1, 0)
children.append(child node) return children
def shuffle(self, puz, x1, y1, x2, y2):
if x2 \ge 0 and x2 \le len(self.data) and y2 \ge 0 and y2 \le len(self.data):
temp puz = self.copy(puz)
temp = temp\_puz[x2][y2] temp\_puz[x2][y2] = temp\_puz[x1][y1] temp\_puz[x1][y1] = temp
return temp puz
else:
return None
def copy(self, root): temp = []
for i in root: t = []
for j in i: t.append(j)
temp.append(t) return temp
def find(self, puz, x):
for i in range(0, len(self.data)):
for j in range(0, len(self.data)): if puz[i][j] = x:
return i, j
class Puzzle:
def init (self, size):
self.n = size self.open = [] self.closed = []
def f(self, start, goal):
return self.h(start.data, goal) + start.level
```

```
def h(self, start, goal): temp = 0
for i in range(0, self.n): for j in range(0, self.n):
if start[i][j] != goal[i][j] and start[i][j] != ' ': temp += 1
return temp
def process(self, start data, goal data): start = Node(start data, 0, 0) start.fval = self.f(start,
goal data) self.open.append(start)
print("\n\n")
while True:
cur = self.open[0]
print("\n")
for i in cur.data:
for j in i:
print(j, end=" ")
print("")
if self.h(cur.data, goal_data) == 0:
break
for i in cur.generate child():
i.fval = self.f(i, goal data)
self.open.append(i)
self.closed.append(cur)
del self.open[0]
self.open.sort(key=lambda x: x.fval, reverse=False)
# Define puzzle states
start_state = [['1', '2', '3'], ['_', '4', '6'], ['7', '5', '8']] goal_state = [['1', '2', '3'], ['4', '5', '6'], ['7', '8',
'_']]
# Create Puzzle object and run the process
puz = Puzzle(3)
puz.process(start state, goal state)
```

```
1 2 3

4 6

7 5 8

1 2 3

4 5 8

1 2 3

4 5 8

1 2 3

4 5 8

1 2 3

4 5 8
```

5. Vacuum Cleaner

LAB 2 - VACCUM	
break - The state of the state	Lugher (t
of vaccum-world ().	Contract to
geal state & (A': 'O', B': 'O'}	
(05C+0	
Total Telleral 'x' as only at medicar	a way well.
location-input = input (" Enter Location of Vacc	
status_input input ("Enter status of " + local	ion input) 1
status_input_complement = Input ("Enlis status	of ather groom")
print (" Iritial Location condition" + 2+ x (o	pal state)
excellented (world) / very	01/12
if location_input == 'A':	191
print (" Vacuum is placed in location A")	011
if status input - x the way as nothing	Flore while a
print (" location A is dirty.")	outs.
goal-state [A'] = 'O'	x txf
cost += 1	0 1 1
print (" COST for CLEARING A "+	str (cost-))
print ("Location A has bur clea	ned. ") o 1 1 00
Assessed it that I have been be place and with a	ALL XIXIO
if status-input-complement == '1':	
print (" location () is dirty.")	
print (Moving right to the coc	The state of the s
(a)+ += 1	y ly ta
	Clair State TH
print (" cost for moving Rich	larged and the
good state Listy o'	William Managaria
cost += 1	1.2000
print ("cost for suck" + st	T (MSF)
print ("Location B has been	
Telse P-1) x as well of restring	
print ("No action" + str (wsf))	The state of the s
print (" Location B is already	
	O Store
	67

	* 4
if & status_input = : 'o':	111 1100
print ("location A is	already clear ")"
	liment a: "i'm I doll . loop
print ("Location	B is Dirty. ")
print (" Moving RIC	THI to the Cocation B ")
cost += 1	sed Alm Revel ") Josep
print ("COST for	moving RICHT + Str (wit)
	Provide on James
	nda n A miles I") dorg
print (Cost for	suck "+ str (wif)
	has bun Cleaned ")
clse	Catala long I foreg
	str(cost))
print (wst)	
print ("Location B	is already than ") and sor whom
if status-input-co	mpliment ==11. 709708
proint ('Location	A is birty.")
print (" Moving	LEFF to the Location A.")
tost + 1	18 to 10000 and at
print ("cosi f	or moving Lift" + 8tr (cost)
goal-State [A.	Jeron to the total to the of the
print (" Cosa f	or suck of thr (was)
print ("Location	A has been cleaned.")
· ·	Cast for Creation A.
else:	totaling A bas bus closed.
point (wast)	Location to to develo
	s already clean 1)
	CONDIA BORREN SOL EXEL
il Atabus is out car	penint == "1" & xous = 1201
mint (" I asabi-	A is distain
(" Li	EFT to location A. ") reare 1900
A Laboratoria de la companya de la c	ECT = lass bas A ")

cost +: 1 all a sure la	3,3
print ("cost for moring LEFT" + str (cost))	
goal-state['A'] 'O'handyna lega mileta fo	
Cost 1:1 (pant all a make as) tong	
print ("coss for sur x" + str (cost))	
print (" Location A has been cleaned.")	
to elice and the hoteland among of 1800.) long	
print ("No action" + Ltr (Logt)	
print ("Location A is already clean.")	
print (7 dec) with + " x suc and 1205") long	
+ (" GOAL STATE: ") and mad as made of ") though	
	×27
1- ("Performance Measurement: " + Lor (wet))	
Cases I strong	
Tayon A (Ke Kitha) D Larry	
location of Vacuum A The Land I have	
status of A)	
status of other rooms who is with	
al location condition { A': 0' B': 103	
um is placed in Location A	
Hon A is borry and red & referred " James	
for CLEANING A 1	
ion A has bun cleaned.	Ja
tion B is dirty (Mean) 4,000	
ng right to the Location B	3
for moving RIGHT2	
for Suck 3 11 and analysis dings what is	
kan A has the chart A state of	
tion B has been cleaned	
STATE (: " A MEDICAL TO THE MAN AND MANY	

5141	
E. W. O. BHO 3	8 - E 3/43
Performance Measurement: 3	
Enter location of Vacuum B	Cheles realized
tates status of BO	[7] men
total status of other room!	Carl Strange way
Initial location condition & A ': 'O'	`B':'0'3
Vacuum is placed in Location B	T 1 44
0 10 1 10 4	
Location Bis already clean.	interest (many) at stite
Location A is dirty	Colors were ward
Moving LEFF to the bushion A	
Cost for moving LEFT 1	OTEA STATES
	Cusues Hong
LOCATION A has been cleaned.	212 -1 1 2 2 3 2 3
	12 depend - simula ji
(NOAL STATE: 2013	("muses") tong
Performance Measurement: 2	
Bup [6+3] tamp [67 - 16	
La serie de la 1911	
(green strong liver at silling	
All married to the control of the co	
1 1 1 1 1 about Sterlar	n State (M) authors Sed T
Samo of day man his way	at that many I
Thomps of the many bee were	Kampa Billy Alexander
	and the same of th
	as belieft Manner at Land
Tellian Hole (artoll billie	(1.)
The second second	17 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
Martin [1 2 3 1 5 1 - 1 , 4 , 7 , 4 , 5]	
The torget and	Telefoliand
	(w) haragen b
	Ila + 2 I as for d 1

```
def vacuum_world():
        # initializing goal state
# 0 indicates Clean and 1 indicates Dirty
goal_state = {'A': '0', 'B': '0'} cost = 0
location input = input("Enter Location of Vacuum") #user input of location
vacuum is placed
status_input = input("Enter status of " + location_input) #user_input if
location is dirty or clean
status_input_complement = input("Enter status of other room")
print("Initial Location Condition" + str(goal_state))
if location input == 'A':
# Location A is Dirty.
print("Vacuum is placed in Location A") if status input == '1':
print("Location A is Dirty.")
# suck the dirt and mark it as clean goal state['A'] = '0'
cost += 1 #cost for suck print("Cost for CLEANING A " + str(cost))
print("Location A has been Cleaned.")
if status_input_complement == '1': # if B is Dirty
print("Location B is Dirty.")
print("Moving right to the Location B. ")
cost += 1 #cost for moving right print("COST for moving RIGHT" +
str(cost))
# suck the dirt and mark it as clean
goal state['B'] = '0'
cost += 1 #cost for suck print("COST for SUCK " + str(cost))
print("Location B has been Cleaned. ")
else:
print("No action" + str(cost))
# suck and mark clean print("Location B is already clean.")
if status_input == '0':
print("Location A is already clean ")
if status input complement == '1':# if B is Dirty
print("Location B is Dirty.")
print("Moving RIGHT to the Location B. ")
cost += 1 #cost for moving right print("COST for moving RIGHT " +
str(cost))
goal state['B'] = '0'
cost += 1 #cost for suck print("Cost for SUCK" + str(cost))
print("Location B has been Cleaned. ")
```

```
else:
print("No action " + str(cost)) print(cost)
# suck and mark clean print("Location B is already clean.")
else:
print("Vacuum is placed in location B") # Location B is Dirty.
if status_input == '1':
print("Location B is Dirty.")
# suck the dirt and mark it as clean goal_state['B'] = '0'
cost += 1 # cost for suck print("COST for CLEANING " print("Location B has
been
if status input complement # if A is Dirty
== '1':
print("Location A is Dirty.") print("Moving LEFT to the Location A. ")
cost += 1 # cost for moving right print("COST for moving LEFT" +
str(cost)) # suck the dirt and mark it as clean goal_state['A'] = '0'
cost += 1 # cost for suck
print("COST for SUCK " + str(cost)) print("Location A has been Cleaned.")
else:
    print(cost)
    # suck and mark clean
print("Location B is already clean.")
if status_input_complement == '1': # if A is Dirty print("Location A is
Dirty.")
print("Moving LEFT to the Location A. ")
cost += 1 # cost for moving right
print("COST for moving LEFT " + str(cost)) # suck the dirt and mark it as
clean goal_state['A'] = '0'
cost += 1 # cost for suck
print("Cost for SUCK " + str(cost))
print("Location A has been Cleaned. ") else:
print("No action " + str(cost))
+ str(cost))
Cleaned.")
               # suck and mark clean
print("Location A is already clean.")
   # done cleaning
```

```
print("GOAL STATE: ")
print(goal_state)
print("Performance Measurement: " + str(cost))
vacuum_world()
```

```
Enter Location of VacuumA
Enter status of A1
Enter status of other room1
Initial Location Condition{'A': '0', 'B': '0'}
Vacuum is placed in Location A
Location A is Dirty.
Cost for CLEANING A 1
Location A has been Cleaned.
Location B is Dirty.
Moving right to the Location B.
COST for moving RIGHT2
COST for SUCK 3
Location B has been Cleaned.
GOAL STATE:
{'A': '0', 'B': '0'}
Performance Measurement: 3
```

```
Enter Location of VacuumB
Enter status of B0
Enter status of other room1
Initial Location Condition{'A': '0', 'B': '0'}
Vacuum is placed in location B
0
Location B is already clean.
Location A is Dirty.
Moving LEFT to the Location A.
COST for moving LEFT 1
Cost for SUCK 2
Location A has been Cleaned.
GOAL STATE:
{'A': '0', 'B': '0'}
Performance Measurement: 2
```

6. Knowledge Base Entailment

AI LAB - 6	
The Biggins and Allendary	
elo def evaluate expression (q,p,r):	
expression-result = (I not of or not por r) and (not and q)	g and p)
return expression-result	
def generate truta-table ():	
print (" q p1 ~ 1 Expression (kB) 1 any (r)")	
print ("")	
The Total Art Thought to Thought	
for g in [ime, false]:	
for p in lime, talsed!	
for r in [True, False]:	
expression - result = evaluate - expression (q,p,	x)
gury result = r	
TOTAL COLOR	- // -
print (1" Eq 3 1 E p 3 1 E cxpression result?	١
2 grong-result 3")	
def gury-entoils-knowledge ()!	
for q in [True, Foolse]:	
for p in [True, false]:	
for rin [iru false]:	
expression-result = evaluate expression (q.p.	(2)
gury-result = r	
V	
if expression result and not query-result:	
neturn false	
return 1 me	
versage as falls	
	store

		* *
dy main 1):	LERD / GARL	menoged Lety 15" Voice
generate truth t	able	to to to 1 1 Decora
0		
i) gury estails	- Knowledge ().	" [will well to p mi
print "In kn	sulida las e	stoils quey ")
else:	and says	Dated by Claim and
orials (") a kno	l. D have o	(" usus listes the real
trace in suc	may and	Loes not entail query")
ilnane== "	to base	1 Company
		2/1-4-1/
main ()		31 (p.1.1) toing
011-00-		Ducke Prints
OUTPOT !		
	1	of questioners stinger presidents.
9 P 8 6	expression (KB)	· · · · · · · · · · · · · · · · · · ·
	^	Listed merid may and
Mu True True	False	Polos well Tom of
hu Tru false	False	Most noderfalse
ion tals True		- there pulgu
Irm false fatse	Falu	false
Falsy True True	o totalises the	we managed Jour
talse Iru False	False	ental multialse
ialse false Town	false	Pom
false false false	Falsi	at the
		Land States of the Landson of the La
Knowledge Base entai	6 guy	: Chaine b
0	, 0	(J.M. Alud . Catters
)		2
ly evaluate expression	(9,0,8):	I amb entail baseles
experien smith !	(o or a) and 1	not or p)) a " lang
meting execution and	WIT .	5 7 2 2 2 2 2
return expression-re	Laday his asa	to send geterhounded") sing
		S and a second
of generale truth tob	CC ()	shre
49		store 67

print (" g | pl + 1 Expression (KB) 1 anery (+)") print (" --- 1 -- 1 -- 1 -- -for q in [True, false]: for p in LTow, false]: for r in [Tru, False]: expression result - evaluate expression qip, ~ gury-risult - p and ~ print (1" { 9 3 1 { 9 3 1 { 7 3 1 { expression result 3 Equiry-rout3") FUTTUR duf guery-entails-knowledgel): for of Intiru, fairi! for pin [iru, false]: for or in [Tru, false]: expression rout: evaluate expression (9, p, r) result - p and r return true dy main (): 3 delwars generate - truth - table () if gury-entails-knowledge(): print ("In knowledge basi entails quiry") else: "Inknowledge Gass does not entail guy ");

i)	ane -	" main			
D	cini			1 x y 7 to	
			in we work	the 19 print old a	
dupou	1 !			b and	
				Colour accur I man	18
9	P	~	ks	Dury (1) Page when the	- 4
Tru	Tru	-	Tru	Thus other I whose is so	ila.
irm	True	false		198 (False 1812 / 1/42) 1 1/) 400	
True	Fralse	True		the false (OBA' -) this	
True	Traise	Faire	fru	False	
False	True	Tru	True	True 11 sport of gold o	9.1
false	True	False	True	1/8 3 Fally 1/ 8 17 1 1 Gary	4
false	false	True	Folse	Police	
folse	False	false	Falsa	False	
Knowle	dge Bas	e does n	at entail qu		-
Knowle	dze Bas	e does ,	at entail qu		-
Knowle	dy Bas	does n		Canno I want	-
Knowle	dy Bas	does n		Carrier (no Element ! I terper to a comment of the	-
Knowle	der Bas	does n		Canno I want	-
Knowle	dy Bas	does n		Carrier (no Element ! I terper to a comment of the	
Knowle	dy Bas	does n		Cenny (chara) if trying the common of the co	
Knowle	dy Bas			Central I Emile 3 m J and the Central Secretary and Secretary 1 to 1 t	
Knowle	der Bas			Cerus (church) if toping the country of the church) is a country of the church of t	
Knowle	de Bas			Central I Emile 3 m J and the Central Secretary and Secretary 1 to 1 t	
Knowle	der Bas			Central ! Emile ? " I topy ! if topy	of the second
Knowle	der Bas			Cerus (church) it is provided to the country of t	
Knowle	der Bas			the first file of the file of the series of	
Knowle	der Bas	100		the of the ford the file of the first than the file of	
Knowle		100		the of the ford the file of the first than the file of	

```
1) Evaluate the given expression (\neg q \lor \neg p \lor r) \land (\neg q \land p) \land q. Check whether knowledge base
entails query or not.
def evaluate expression(q, p, r):
# Evaluate the given expression (\sim q \ v \sim p \ v \ r) ^ (\sim q \ ^p) ^ q expression_result = ((not q or not
p or r) and (not q and p) and q) return expression result
def generate truth table():
# Print the header of the truth table
print(" q | p | r | Expression (KB) | Query (r)") print("---|---|--------")
# Evaluate and print each row of the truth table for q in [True, False]:
for p in [True, False]: for r in [True, False]:
expression result = evaluate expression(q, p, r) query result = r
print(f'' \{q\} | \{p\} | \{r\} | \{expression result\} | \{query result\}'')
def query entails knowledge():
# Check if query entails the knowledge for q in [True, False]:
for p in [True, False]: for r in [True, False]:
expression result = evaluate expression(q, p, r) query result = r
# If the expression is true and the query is false, query does not entail the knowledge
if expression result and not query result: return False
# If the loop completes without returning, query entails the knowledge
return True
def main():
# Generate and print the truth table generate truth table()
# Check if query entails the knowledge and print the result if query entails knowledge():
print("\nKnowledge base entails query") else:
print("\nKnowledge base does not entail query")
if name = " main ": main()
```

```
input
             Expression (KB)
                                 Query
True
                       False
        True
                True
                                               True
True
        True
               False
                        False
                                                False
True
        False
                 True
                        False
                                                True
True
        False
                 False | False
                                                 False
False
                       False
         True
                 True
                                                True
False
                False
                         False
                                                 False
         True
False
                  True
                         False
         False
                                                 True
False
         False
                 False
                                                  False
                        False
Knowledge base entails query
```

 Evaluate the given expression (p v q) ^ (~r ^ p). Check whether knowledge base entails query or not.

```
def evaluate expression(q, p, r):
# Evaluate the given expression (p v q) ^{\circ} (\sim r ^{\circ} p) expression result = ((p or q) and (not r or
p)) return expression result
def generate truth table():
# Print the header of the truth table
print(" q | p | r | Expression (KB) | Query (r)") print("---|---|--------------")
# Evaluate and print each row of the truth table for q in [True, False]:
for p in [True, False]: for r in [True, False]:
expression result = evaluate expression(q, p, r) query result = p and r
print(f'' \{q\} | \{p\} | \{r\} | \{expression result\} | \{query result\}'')
def query entails knowledge():
# Check if query entails the knowledge for q in [True, False]:
for p in [True, False]: for r in [True, False]:
expression result = evaluate expression(q, p, r) query result = p and r
# If the expression is true and the query is false, query does not entail the knowledge
if expression result and not query result: return False
# If the loop completes without returning, query entails the knowledge
```

return True

```
def main():
# Generate and print the truth table generate_truth_table()
# Check if query entails the knowledge and print the result if query_entails_knowledge():
print("\nKnowledge base entails query") else:
print("\nKnowledge base does not entail query")
if __name__ == "__main__":
main()
```

```
input
             Expression (KB)
                                Query
 True
               True
                       True
                                             True
        True
               False
        True
                        True
                                              False
 True
                       False
 True
        False
                 True
                                               False
 True
        False
                False | True
                                               False
 False
         True
                 True True
                                              True
 False
         True
                False
                         True
                                               False
                                                False
 False
         False
                  True
                         False
                  False
                                                 False
 False
         False
                        False
Knowledge base does not entail query
```

7. Knowledge Base Resolution

			-/1/0-Ve	1016	13
	WEEK-7			C/a	cor C
Create a kb using	Pl and 1	nove the	muy usi	of head	
import re	0		v 0	9 . (untall
def main (rules, goal	Ca				
rules = rules speit (201	1. 95	9	
steps = resolve (rue	us, gool)	Just?	total.	well	Loyf
print (In step/+1	clause It I Deri	valorit')	igtal.	3445	Haliley
print ('- ' 130)	W.A.	pulso?	post?	Tayof.	and?
ial	- Wal	lor)	Veller	5/11/0	_uvr/
for step in steps :	Aur Trees	wil.	1000	Just?	ight t
byut (1. Ei3.	1618 step 316	19 8 taps 9 51	tp3/t')	port!	galact
1421	972	relat	wiff	solut)	14,000
	ruta i	Falsa	Solve	1,201	10/13
return 1' ~ { term	U	j : 100 to 11	else term	[0]) kon Al
dy reverse (clause)		તું : moteo (4)	clse term	[1]	thin sh
dif reverse (clause)		પું : ભાગ ા (ન	clse term	נים אין	r kun Ad
def reverse (clause): if ten (clause) > 2 t - split - terms	· (clause)		clse term	to ye	r town All
def reverse (clause) if ten (clause) > 2 t - split-terms return f' { tl	· (clause)		clse term	10 y	r tous 131
def reverse (clause): if ten (clause) > 2 t - split - terms	· (clause)		clse term	10 y	r kun a M
return (" ~ { term" def reverse (clause) " if her (clause) > 2 t - split - terms return f ' { tl return "	(danse) 1)3 V { t (o)		clse term	10 y	r kons a V
def reverse (clause) if her (clause) > 2 t - split - terms return f' { tl return "	(danse) 1)3 V { t (o)		else term	10 y	t kon s N
def reverse (clause) if her (clause) > 2 t - split - terms return f' { tl return "	.: (clame) 1)3 V { t (o)"	3'	clse term	10 y	r kun a M
octure (clause) def reverse (clause) if her (clause) > 2 t - split - terms return f' { the return " def split - terms (ru exp = ' (~ * [p	.: (clame) 1)3 V { t (o)"	3'	clse term		t kun a M
octure (clause) def reverse (clause) if her (clause) > 2 t - split - terms return f' { the return " def split - terms (ru exp = ' (~ * [p term = re- findo	.: (clame) 1)3 V { t (o)"	3'	clse term		1 kgus 1 N
octure (clause) def reverse (clause) if her (clause) > 2 t - split - terms return f' { the return " def split - terms (ru exp = ' (~ * [p term = re- findo		3'	clse term		1 kgus AV
def reverse (clause) if les (clause) > 2 t · split - terms return " { t l return " def split - terms (ru exp = " (~*[p term : re findo end return terms def contradict (goal	(clame) 1) 3 V { t (o) de): ars])' all (corp, rule) , clame):	3'			t kun s N
def reverse (clause) if les (clause) > 2 t · split - terms return " { t l return " def split - terms (ru exp = " (~*[p term : re findo end return terms def contradict (goal	(clame) 1) 3 V { t (o) de): ars]) clame) clame) clame) f good 3 V {	3. regati (good	١١١٠)		I koo a N
def reverse (clause) if les (clause) > 2 t - split - terms return f' { the return " def split - terms (ru exp = ' (~*[p term : re findo gred return terms def contradict (goal contradiction : [f	clame) (clame) (clame) (clame) (ars]) (cup, rule (cup, rule (cup, rule (cup, rule) (cup, rule) (cup, rule)	regali (gor	ne))'		A adichion
def reverse (clause) if les (clause) > 2 t - split - terms return f' { the return " def split - terms (ru exp = ' (~*[p term : re findo gred return terms def contradict (goal contradiction : [f	(clame) 1) 3 V { t (o) de): ars]) clame) clame) clame) f good 3 V {	regali (gor	ne)];		

dy resolve (rues, goal): temp + = [regate (good)] steps = dict() for rule in tange: " I () steps [rati] = 'given' slips [nigate (goal)] = 'Negated conclusion' while i < les (temp) n= len (temp) j=[i+1)%-n terms = split terms [temp (i)] terns2 = split-terns[tamp(j)) for c is terms 1 if regali (c) in terms. tl- [+ for t in time 1 if t = c] t2. [t for t in terms if ti - regula] gen = t1+t2 9 9 len (gen) == 21 if gento): rejute (genti)): (land along) clauses + . () '{ gin [0] 3 v } gen [, 23') else: if contradiction (goal, f'Equi [0] 3 v { gur [1] 3'): temp. append ({ 2 gen (0) 3 v 8 gen (1) 331) steps [']] " Resolved { temp[i] } and E temp (j) 3 to E temp [-1] 3, which is the term muly

elif len (gen): 1: C'EC-1728 17 = + 22 2005) else: if contradiction (good, f' & terms (0)3 vE terms 2003'): temp. append (| Eterns 1 (0) v { terms 2 (0) ')

steps ['] - | "Resolved { temp (1) 3 and { temp (j) 3 to Etemp [-13] which is in turn null. In A contradiction is found when Egal (goal) 3 15 assumed as true. Hence, & goal 3 is true." return steps for clause in clauses: if clause not in temp and clause! - reverse Eclause] and never (clause) not- in temp? temp. append (clause) stips (clause ? + f Resolved from ? temp (:) 3 and Etemp (j33: j= (j+1)1.00 n return steps muly = RUNP RUND Output: Derivoh'sn 1. RNNP RVND 3. PRVP

4. NRVQ	0.
5. ~ R	given
	Regalid conclusion Resolved RUNP and NRUP to RUN
<u>Re</u>	
A second of the second of the	which is In two null
14. corculation is	found when MR is assumed as to
Hence Ris true.	Color of the Capetal region (Sept. 2 Smile
	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
	The first to fine ?
A 7 76 X	A No Married Antibracy of Thing
	The state of the s
	A T P and the Control of the Control
) Type ("() gider types - types
Page Caute (* g	O W TO STATE STATE
(X) of seconds of	Paul 2u.
	O (1) 2 as dill dedut
ARABI HATTA	tapeation ("Capea Shalk V.)
patient 1 Direct of	Configura, Layera Piptay At Ga., Gr. 11 253
	water Sie and Marian 1 a fi
	See Tradrocheritisms
	of the for his it smealth I fills
	Some Edited on the Folders
	in to builthaid to file
li justeli hiddena disa " :	to : I to I no duff robed"
(19) / 3	harita file
Addoporos T. brying is	Print (Comment county)
Lichel Bernstein	sure of Take I C drawing
of the Party of the	a lease of the hand of the
	or Hillston and so
the section of many	er i mal (Dabage (*);
and the land of Great	apply - rubet between troppe with the
	The sport of the day of the sales of the

Create a knowledgebase using prepositional logic and prove the given query using resolution.

```
import re
def main(rules, goal):
rules = rules.split(' ')
steps = resolve(rules, goal) print('\nStep\t|Clause\t|Derivation\t') print('-' * 30)
i=1
for step in steps:
print(f' \{i\} \t \{step\} \t \{steps[step]\} \t') i += 1
def negate(term):
return f \sim \{\text{term}\}' \text{ if } \text{term}[0] != '\sim' \text{ else } \text{term}[1]
def reverse(clause): if len(clause) > 2:
t = split terms(clause)
return f'{t[1]}v{t[0]}' return "
def split terms(rule):
exp = '(\sim *[PQRS])'
terms = re.findall(exp, rule) return terms
split terms('~PvR')
['~P', 'R']
def contradiction(goal, clause):
contradictions = [f\{goal\}v\{negate(goal)\}', f\{negate(goal)\}v\{goal\}']
return clause in contradictions or reverse(clause) in contradictions def resolve(rules, goal):
temp = rules.copy() temp += [negate(goal)] steps = dict()
for rule in temp: steps[rule] = 'Given.'
steps[negate(goal)] = 'Negated conclusion.' i=0
while i < len(temp):
n = len(temp) j = (i + 1) \% n clauses = [] while j != i:
terms1 = split terms(temp[i]) terms2 = split terms(temp[i]) for c in terms1:
if negate(c) in terms2:
t1 = [t \text{ for } t \text{ in terms } l \text{ if } t != c]
t2 = [t \text{ for } t \text{ in terms } 2 \text{ if } t != \text{negate(c)}] \text{ gen} = t1 + t2
if len(gen) == 2:
```

```
if gen[0] != negate(gen[1]):
clauses += [f'\{gen[0]\}v\{gen[1]\}']
else:
if contradiction(goal,f {gen[0]} v {gen[1]}'):
temp.append(f'\{gen[0]\}v\{gen[1]\}')
steps["] = f"Resolved \{temp[i]\} and \{temp[j]\} to \{temp[-1]\}, which is in turn null. \
\nA contradiction is found when {negate(goal)} is assumed as true. Hence, {goal} is true."
return steps elif len(gen) = 1:
clauses += [f'\{gen[0]\}'] else:
if contradiction(goal, f {terms1[0]} v {terms2[0]}'): temp.append(f {terms1[0]} v {terms2[0]}')
steps["] = f"Resolved {temp[i]} and {temp[j]} to {temp[-1]},
which is in turn null. \
\nA contradiction is found when {negate(goal)} is assumed as
true. Hence, {goal} is true." return steps
for clause in clauses:
if clause not in temp and clause != reverse(clause) and reverse(clause)
not in temp: temp.append(clause)
steps[clause] = f'Resolved from \{\text{temp}[i]\}\ and \{\text{temp}[j]\}.' j = (j + 1) \% n
i += 1 return steps
rules = 'Rv\sim P Rv\sim Q \sim RvP \sim RvQ' \#(P^{Q}) \le R : (Rv\sim P)v(Rv\sim Q)^{(\sim RvP)^{(\sim RvQ)}}
goal = 'R'
main(rules, goal)
rules = 'PvQ PvR ~PvR RvS Rv~Q ~Sv~Q' # (P=>Q)=>Q, (P=>P)=>R, (R=>S)=>~(S=>Q)
main(rules, 'R')
```

```
input
        Clause | Derivation
Step
1.
        RV-P
                  Given.
          Rv-Q
                  Given.
2.
          -RvP
                  Given.
 3.
 4.
          -RVQ
                  Given.
                  Negated conclusion.
5.
          -R
                  Resolved Rv-P and -RvP to Rv-R, which is in turn null.
 6.
A contradiction is found when -R is assumed as true. Hence, R is true.
Step
        Clause | Derivation
1.
          PVQ
                  Given.
2.
          PVR
                  Given.
3.
          -PvR
                  Given.
                  Given.
4.
          RvS
5.
          RV-0
                  Given.
 6.
          -Sv-Q
                  Given.
 7.
          ~R
                  Negated conclusion.
8.
                  Resolved from PvQ and ~PvR.
          OVR
9.
          Pv-S
                  Resolved from PvQ and ~Sv-Q.
10.
          P
                  Resolved from PvR and -R.
          -P
                  Resolved from -PvR and -R.
11.
                  Resolved from -PvR and Pv-S.
12.
          RV-S
13.
                  Resolved from -PvR and P.
          R
 14.
          S
                  Resolved from RvS and -R.
          ~Q
15.
                  Resolved from Rv-Q and -R.
16.
                  Resolved from ~R and QvR.
          Q
                  Resolved from -R and Rv-S.
17.
          -S
18.
                  Resolved -R and R to -RvR, which is in turn null.
A contradiction is found when -R is assumed as true. Hence, R is true.
```

8. Unification

Mil Andrew Control	12 × A	92
MEEK 8 10 MM	4 -	2
Rulling French and mere to Purp		-13
lef unity (expr1, expr2): funct, aigs1 = expr1. sper('(', 1)	af a serie	
func2, args 2 = expr2. split-('(', 1') · . with	LEAST
il & func 1 ! = func 2 :		
print ("Expressions cannot be	unfied . Diffuer	1- functions.
return None	,	,
The state of the s	1. 1	
age 1 = age 1. ntip (')'). split	- (',')	
args 1 = args 1. 20tip (')'). 8plit args 2 = args 2. 25tip (')'). split	-(',')	Se 1
substitution - 83		
for a1, a2 in zip(args1, args2 if a1-istowa() and a2-ist substitution[a1] = a2	own() and al	1-02:
ell alistome () and not a.	2.19 laure ():	
substitution [al] = a2		
elif not al. rlower() and a2.	islawer (7:	
Substitution[al] = a1	nsel.	
elizal!=a2	700	
print ("Expressions carro organists.")	t be unified. In	compatible
Jelan None		
setur substitution		
All the Control of th	(S. 1925)	
by apply-substitution (expr, substitution for key, value in substitution		
TO THE PARTY OF TH		
expr = expr. replace (key, val	val]	

expr! = input (" Enter ten first expression expr2 : input (" fite the second expression substitution = unity (expri, expre) substitution: print ("The substitutions are: ") key, value in substitution. items (7: print- (1' { key 3/ { value 3') (2pr 1-result = apply_substitution (expr), substitution) expr2. result apply-substitution (expr2, substitution) print (["Unified expression 1: 8 expr 1_ result 3') print (& Unified expression 2: { expr2-result 3') Outputs Enter the first expression: knows (f(x), y) Enter the Second expression: knows (J, John) The substitutions are (x)/J y/ John Unified expression 1: knows (J John) Unified empression 2: knows (J, John) Enter to first expression: Student (x) Enter the second expression, Teacher (Rose) Expressions cannot be unified. Different functions

	Stand rare neutro
Even the first expression; knows ((Collection)
tales He Heard IMPRILLION KNOWS	(4,
The publishments are:	a Little Control of the Control of t
y 15ohn	1 100 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
2 / Mother by	AUR 11-2
Unified expression 1: knows (John,	Mallin (Tolas)
Unified expression 2 knows (John.	: And water 1
Ester the first expression: like (1	1, 4, 2
Enter the second expression: like (Expressions cannot be unified. In	1110
trypussions cannot be unified. In	composite argumins.
front dareton by the board then	of the leading
Contable Competer Conductions	Gara .
71.6	18/1/20
17 States track of the	andre order francis
(Labor brought & religion	ches perhaps 1 sensit
	1 4 1 A
	thulpula !
	A morninger trop of the
Tadol I I have no	
	The state of the s
THE RESERVE OF THE PERSON OF T	L/ (E)
CARL DIS	
and the state of the	Unified Supposement Surject
Testile delegation	
(*) Tribos	to be the fine topogramme all
and the Alexander	I married branch at what
	hypariad diasot markeyy)
LOSSING TAKE SEE	

```
def unify(expr1, expr2):
Split expressions into function and arguments funcl, argsl =
exprl.split('(', 1)
func2, args2 = expr2.split('(', 1)
    if func1 != func2:
print ("Expressions cannot be unified. Different functions.") return
args1 = args1.rstrip(')').split(',') args2 =
args2.rstrip(')').split(',')
substitution = {}
for al, a2 in zip(args1, args2):
if al.islower() and a2.islower() and a1 != a2:
substitution[a1] = a2
elif al.islower() and not a2.islower():
substitution[a1] = a2
        substitution[a2] = a1
   elif al != a2:
print ("Expressions cannot be unified. Incompatible
    return substitution
def apply substitution(expr, substitution):
for key, value in substitution.items(): expr = expr.replace(key,
value)
return expr
expr1 = input("Enter the first expression: ") expr2 = input("Enter
substitution = unify(exprl, expr2)
```

Outputs:

```
Enter the first expression: knows(f(x),y)
    Enter the second expression: knows(J, John)
    The substitutions are:
    f(x) / J
    y / John
    Unified expression 1: knows(J, John)
    Unified expression 2: knows(J, John)

→ Enter the first expression: Student(x)

    Enter the second expression: Teacher(Rose)
    Expressions cannot be unified. Different functions.

☐ Enter the first expression: knows(John,x)

    Enter the second expression: knows(y, Mother(y))
    The substitutions are:
    y / John
    x / Mother(y
    Unified expression 1: knows(John,Mother(y)
    Unified expression 2: knows(John, Mother(John))
Enter the first expression: like(A,y)
    Enter the second expression: like(K,g(x))
    Expressions cannot be unified. Incompatible arguments.
```

9. FOL to CNF

WEEK-9	I'CHCS Daro
for to cop	I to I not . I - buntlett
the Churchels If &	1) Make Jaco zatrino
dy get Attributes (string):	Column of Action and
exex = , /([,)]+/),	hursel & foundlake
matches: re.findall(expr. string)	Unbil or + Woodlets
return [m for m in str (matches)	if m. isalpha ()]
Philippine A State on the	milest Transfell the
dy get-Predicates (thing):	Kritis - darberg ref
expr: '[a-z~]+\([A-Za-z,]+\)	
return re. findall (expr. string)	SECOND TO SECOND
Tanci Manage of the Meton I malgor manufall	
dy betweegen (sentence):	Long Tiles
string = " join (list (sentence) . copy ()	The state of the s
string = string. replace ('~~')	
flag : [in string	
string = string . repedice('~[', '')	
for predicate in get Predicates (string)	
string: string. replace (predicate, f	
s=list (string)	1,000,000
for i, c in conversate (string):	et est - se troo
il c == 11:	(July) Insuer left in
ر. ۱۹ [۱] - ۱۶.	water to I have date
	templets of " state
ال المالية الم	by the dansatette
string: ' join (s)	I damalely ton
string = string replace (" ~")	January - USE
return [[{ string }] "] flag else str	ing laundell
Page (O me of) O	Invested Statestels
Sholenization (sentinu):	1010111/ yes
" SKOLEM_CONSTRAINTS . [[Chr (c) 3'	for cin range (ord ('A'),

ord ('z')+	1)]
Material	- · ' . io: (list (sentence) . copy ())
matches	= re. findale ('[HF].', Statement)
for mat	th in matches [::-1]: (and) attached the fit
state	ment : statement. replace (match) (1) (1)
Mate	ments = re. findall ('ICICI_]]+ \] , statement)
lor	sin statements:
1	statement: Statement, replace (s, s[1:-1])
901 0	redicate in getPredicate (Statement):
Lufter to	attributes : get Attributes (predicate)
Capital Car	if " join (attributes) is lower () :
	Statement: Statement, replace (match [1], SKOUTH_CONSTRA
	(NTS.pop (O))
	els: (proposi (sonitre) los) das la estate
	al: [a for a in attributes if a relower ()]
	av- La for a in attributes of not a islamer ()][0]
	Statement - Statement . replace (aU) (SKOLTM_CONST
	RAINTS. pop(0)3 (Eaclo] if un (ac) the match!
	3)') (ziner) marine true in the thing bed
xt	tier statement ! Buildy lessey a state pride
	(2-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1
import re	Control of Survey of Sinol
	M((101):
Statement	[]ol. replace ("<=>"")
while '-	in statement:
	atimust.index('-')
	statement = '[' = statement [:i]+'=>' + statement [i+1=]
17	&['+ statement [i+1:]+ "' -> '+ statement [: i]+)
	statement - new statement
Untin	unt = statement replace ("=>", "-")
OTOKUP AMAY	= 1/ E(E_J;+)/J. A month of proper manger of
expr	inents = rc. findall (expr. statement)
4 Ame 40	

for i, & in enumerate (statements):	ann Data
if 'E' in a and 'J' not in s: O' ! [O')	1-1-1-1-1
statements [i] += i] . when I see that I see	
for s in statements:	e di di ref
statement - statement replace (&, fol-to-cr) (&) attell
while '-' in statement:	teh colle-
The color of the state wint when () + 1) a top of medical or	
br = statement . ('L') if '['in statem	int else 0
new statement : 'n" + statement [br: i] + i1	+ statement [
+1:1 (("CN) haming a ((6) N)	
statement = statement [: br] + new_statement	i) 62>0 els
new. Statement	6 Luglary
while '~ &' in statement:	
i = Statement. India ("~ V .) A ~ & E) will !	(A) hoof in
statement = list (statement)	(x) mail
state wit [i], statement [i+i], statement	
statement [i+2], '~'	
statement = ' '. join (statement)	
while 'at in statement:	
i = state must . Indus ('~]')	Lange III.
6: ligt (blatement)	
δ[i], δ[i+i], δ[i+2] · 'V', δ[i+2], '~'	
statement ". join (1)	
Statement : Statement replace ('~[V', [~ V')	
Statement: Statement. replace ('~[3', '[~]')	
(CEINJ) = 1 (MEINJ)	
statements = re-findall (expr. statement)	
for s in statements:	aby (1 th
statement: Statement . replace (s, fol-to-c	NO.
expr = '(~[VI]].)'	(
state nunts = re. findall (expr. statement)	
for s in statements:	
	store 67

statement = statement, repla	a (6, folto up (1))
1 [/+[[/]]/~, -1das	at some of that a so the
statements : re findale l'expr,	
for s is statements:	structured and red
statement = statement rep	lace (s, DeMorgan (s))
setur statement	through the state of
print (Akolimization (fol-to-c	of (" 4x food (x) => likes (John, x)"))
Datpato print (sholenization ((("[[(s, a) and] x []x ")] no - at log
print (fol to af ("[american	(x) & weapon (y) & sells (x, y, z) &
12 Od od 1 browstell sun = [
Output 1	Samuel Color
	+ Througholt + W- Oaky
~ food (A) 1 Okes (John, A)	") subst damatell " i
[((x)B(x))]	Low Block Delicities of association of
	a) and francisco
	Full add to the district
	(Travers 12) see 1
of Feeling Vi	Carilly Elega Fills
	Crimity and a season
(V-) 035'),	(Charge Samethold Samethold
(v-) va-'),	salger demonstrate territori
(v-) vg-').	Telgio territoti territoti
L Ref. Flet)	Telgio favorilett Javardelli CEFIVIII 1
L Ref. Flet)	Telgio tamatele tamatele CEETVION poet
h town to be of	Telgro touritels touritels CEELVIEL 1991 (1) the ball of Standards (2) the ball of Standards
h town to be of	relgio damentale damentale relgio damentale damentale CEELVIDAL aport (1) see built as diamentale chimalate damentale landate damentale
Go as left a langer	Telgio deventete deventete Telgio deventete deventete C.F. E. V. J. v.
Go as left a langer	relgio damadale damadale CEETVITT Proper (1) ste ball in Strandale I then alote also mol Limitale damadale V. LETVITT 1 1909
Go a lot x hostoge	Telgio deventete deventete Telgio deventete deventete C.F. E. V. J. v.

Convert given first order logic statement into Conjunctive Normal Form (CNF).

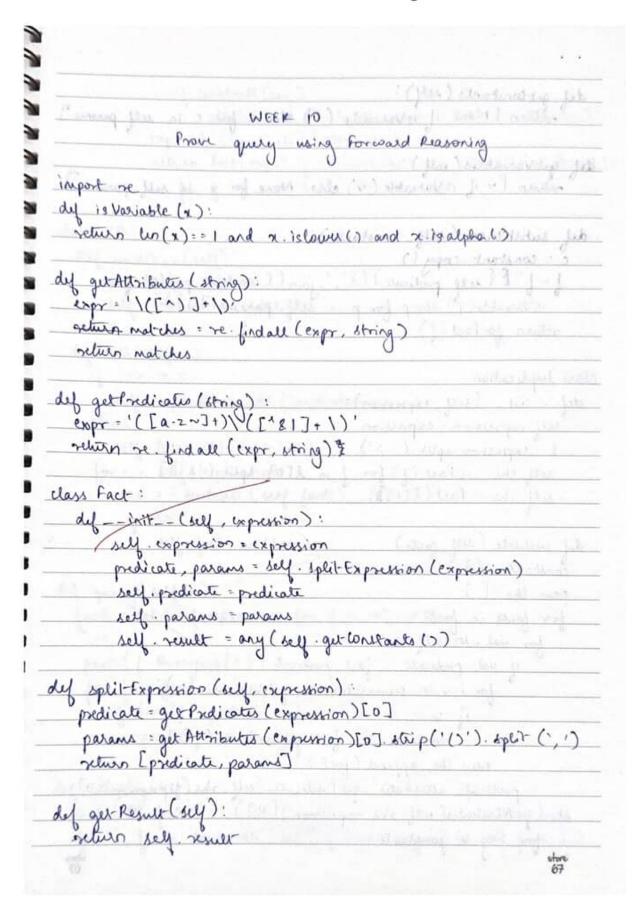
```
def getAttributes(string):
expr = '\([^)]+\)'
matches = re.findall(expr, string)
return [m for m in str(matches) if m.isalpha()]
def getPredicates(string):
expr = '[a-z\sim]+\([A-Za-z,]+\)' return re.findall(expr, string)
def DeMorgan (sentence):
string = ''.join(list(sentence).copy()) string =
string.replace('~~','')
flag = '[ in string
string = string.replace('~[','')
string = string.strip(']')
for predicate in getPredicates(string):
string = string.replace(predicate, f'~{predicate}') s = list(string)
for i, c in enumerate(string):
string = ''.join(s)
string = string.replace('~~','')
return f'[{string}]' if flag else string
def Skolemization(sentence):
SKOLEM CONSTANTS = [f'{chr(c)}' for c in range(ord('A'),
ord('Z')+1)]
statement = ''.join(list(sentence).copy()) matches =
re.findall('[V3].', statement) for match in matches[::-1]:
statement = statement.replace(match, '') statements =
re.findall('\[[\[[^]]+\]]', statement) for s in statements:
statement = statement.replace(s, s[1:-1]) for predicate in
getPredicates(statement):
attributes = getAttributes(predicate)
if ''.join(attributes).islower():
                statement =
statement.replace(match[1], SKOLEM CONSTANTS.pop(0)) else:
aL = [a for a in attributes if a.islower()]
aU = [a for a in attributes if not a.islower()][0] statement =
statement.replace(aU,
```

```
['{SKOLEM CONSTANTS.pop(0)}({aL[0] if len(aL) else match[1]})')
return statement
def fol to cnf(fol):
statement = fol.replace("<=>", " ") while ' ' in statement:
        i = statement.index(' ')
new statement = '[' + statement[:i] + '=>' + statement[i+1:] +
']&['+ statement[i+1:] + '=>' + statement[:i] + ']'
        statement = new statement
statement = statement.replace("=>", "-") expr = '\[([^]]+)\]'
statements = re.findall(expr, statement) for i, s in
enumerate(statements):
if '[' in s and ']' not in s: statements[i] += ']'
for s in statements:
statement = statement.replace(s, fol to cnf(s))
while '-' in statement:
i = statement.index('-')
br = statement.index('[') if '[' in statement else 0 new statement =
'~' + statement[br:i] + '|' + statement[i+1:] statement =
statement[:br] + new statement if br > 0 else
new statement
while '~∀' in statement:
i = statement.index('~∀')
statement = list(statement)
statement[i], statement[i+1], statement[i+2] = '3',
statement[i+2], '~'
while '-3' in statement:
i = statement.index('~∃')
s = list(statement)
s[i], s[i+1], s[i+2] = 'V', s[i+2], '-' statement = ''.join(s)
statement = statement.replace('~[∀','[~∀') statement =
statement.replace('\sim[\exists','[\sim\exists') expr = '(\sim[\forall[\exists].)'
statements = re.findall(expr, statement)
for s in statements:
statement = statement.replace(s, fol to cnf(s))
expr = '~\[[^]]+\]'
statements = re.findall(expr, statement)
```

```
for s in statements:
statement = statement.replace(s, DeMorgan(s))
return statement
```

Output:

10. Forward reasoning



def get constants (sey): return [None if is Variable (c) else c for c in self parons] def get variables (ulf): return [" if 18 Variable (v) else None for v in self params dy substitutes (self, constants): c = constant copy () b. 6" [E self. predicate 3 (E', '. join ([constants pap (0) if is Variable (p) elep for p in self. paranis 703) return for fact (f) class Implication: def -- init - (self, expression): self expression : expression l = expression, split ('=>') self the = [fact (f) for f in 200] split (8) self . This = fact (P [1] 3) dy evaluate (seef, facts): new-Chs = [] for facts in facts: for val. in self the if val. prediate == fact. prediate: constants [v] - fact getConstant () [i] new the append (fact) predicate, attributes : get Predicates (self. rhy. expression)[0] Ato (gutatributes (self. The expression) [0])" for key in constants:

if constants [key]:	
attributer : attributes	replace (key, constants [key]
color P colorana 21 anni 20	us j
for 1 in new Chall the	o_lhs) and all (Lf. get Resul
	Columbat Language Language
class KB:	of Chief advent 1 At a
	and with more little
	Con acres Manage State and
self. implications = set ()	1000 1 Tr. 100 11 100 11
	manuffer residence that the
dy tu (wy, c):	to the same to partie of
if '=> 'in e:	· Cladest dd
self implications add (Implication)	())
self. facts add (fact (es))	Jupta
for i is sell implications:	XXI danimet assumed
for i in self implications: res = i evaluate (self facts)	1 Maria Paralando S.
if res:	otas) da
sey, facts add (ms)	Count manifes
0	and the second of the second
def gury (sey, e).	- Buttername
facts : set (f. expression for fin .	self. facts])
1-1	out continues to
print (f' Averying { c 3:')	Later Language James
for fin facts:	Clear Harmer v
if fact (b) predicate = fact (c).	predicate
point (1'It (13. { (3')	
0.1 0.1 ()	
all display (my)	
frint ("All facts:") for i, I in enumerate (set ([] expre	The state of the s

print (1'1t Ei+13. 213')	Maglaces /
The fit of the said I have a mark the should	alotte.
Kb: KB()	
kb. tall ('missile (x) => weapon (x)')	(at node
The control of the co	
kb.till ('missik (MI)')	
kb.till ('energy (x, America) => hostile (x))	24
kb. Lell ('amuican (West)')	13' 1201
No. tele (energ (Nono, America))	Charles and the
kb. tu ('owns (Noro, MI)')	Add and the little
46. till (missile (r) sowns (Nono, x) = sells (h	lest, x, Nono))
kb till (ancien (x) sweapon(y) & sell (x, y, z) st	restil(z) => criminal(z)')
Kb. query ("criminal (x)")	77 UK 00 68
kb display!)	1 1 1 1 1
Control to the con-	reade Ho
	The second secon
Output.	- Contract E
Output:	e mal till
Output: Out	Mai Vand
anaying wining (x):	2/24
Querying criminal (x): fall	2/24
Durying wining (x): wiming (west) All facts:	2/24
Durying winised (x): 1. Winised (west) All facts: 1. hostile (Nono)	2/24
Durying criminal (x): 1. Criminal (west) All facts: 1. hostile (Nono) 2. seles (West, M1, Nono)	2/24
Durying wining (x): 1. wining (west) All facts: 1. hostile (Nono) 2. seles (West, M1, Nono) 3. anewige (west)	2/2/
Durying winisal (x): 1. Wiminal (West) All facts: 1. hostile (Nono) 2. seles (West, M1, Nono) 3. anericas (West) 4. owns (Nono, M1) 5. energy (Mono, America) 6. weapon (M1)	2/2/
Durying winisal (x): 1. Wiminal (West) All facts: 1. hostile (Nono) 2. seles (West, M1, Nono) 3. anericas (West) 4. owns (Nono, M1) 5. energ (Nono, America)	2/14
Durying winisal (x): 1. Wiminal (West) All facts: 1. hostile (Nono) 2. seles (West, M1, Nono) 3. anericas (West) 4. owns (Nono, M1) 5. energy (Mono, America) 6. weapon (M1)	
Durying winised (x): 1. Wiminal (West) All facts: 1. hostile (Nono) 2. heles (West, M1, None) 3. anericas (West) 4. owns (Nono, H1) 5. energy (Nono, America) 6. weapon (M1) 7. criminal (West)	
Durying winised (x): 1. Wiminal (West) All facts: 1. hostile (Nono) 2. heles (West, M1, None) 3. anericas (West) 4. owns (Nono, H1) 5. energy (Nono, America) 6. weapon (M1) 7. criminal (West)	
Durying winised (x): 1. Wiminal (West) All facts: 1. hostile (Nono) 2. heles (West, M1, None) 3. anericas (West) 4. owns (Nono, H1) 5. energy (Nono, America) 6. weapon (M1) 7. criminal (West)	
Durying winised (x): 1. Wiminal (West) All facts: 1. hostile (Nono) 2. heles (West, M1, None) 3. anericas (West) 4. owns (Nono, H1) 5. energy (Nono, America) 6. weapon (M1) 7. criminal (West)	
Durying winised (x): 1. Wiminal (West) All facts: 1. hostile (Nono) 2. heles (West, M1, None) 3. anericas (West) 4. owns (Nono, H1) 5. energy (Nono, America) 6. weapon (M1) 7. criminal (West)	

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

```
import re
def isVariable(x):
return len(x) == 1 and x.islower() and x.isalpha() def
getAttributes(string):
expr = ' \setminus ([^{})] + \setminus )'
matches = re.findall(expr, string) return matches
def getPredicates(string):
expr = '([a-z~]+)\([^&|]+\)' return re.findall(expr, string)
def init (self, expression):
self.expression = expression
predicate, params = self.splitExpression(expression) self.predicate
= predicate
self.params = params
self.result = any(self.getConstants())
def splitExpression(self, expression):
predicate = getPredicates(expression)[0]
params = getAttributes(expression)[0].strip('()').split(',') return
[predicate, params]
   def getResult(self):
        return self.result
   def getConstants(self):
return [None if isVariable(c) else c for c in self.params] def
getVariables(self):
return [v if isVariable(v) else None for v in self.params]
def substitute(self, constants):
c = constants.copy()
f = f"{self.predicate}((','.join([constants.pop(0) if
isVariable(p) else p for p in self.params])})" return Fact(f)
class Implication:
def init (self, expression):
self.expression = expression
l = expression.split('=>')
self.lhs = [Fact(f) for f in 1[0].split('&')] self.rhs = Fact(1[1])
    def evaluate(self, facts):
        constants = {}
        new lhs = []
        for fact in facts:
```

```
for val in self.lhs:
if val.predicate == fact.predicate:
for i, v in enumerate(val.getVariables()): if v:
constants[v] = fact.getConstants()[i] new lhs.append(fact)
predicate, attributes = getPredicates(self.rhs.expression)[0],
str(getAttributes(self.rhs.expression)[0])
            if constants[key]:
attributes = attributes.replace(key, constants[key]) expr =
f'{predicate}{attributes}'
return Fact(expr) if len(new lhs) and all([f.getResult() for f
in new lhs]) else None
class KB:
   def init (self):
       self.facts = set()
       self.implications = set()
self.implications.add(Implication(e)) else:
            self.facts.add(Fact(e))
       for i in self.implications:
            res = i.evaluate(self.facts)
       if res:
def query(self, e):
facts = set([f.expression for f in self.facts]) i=1
print(f'Querying {e}:')
for f in facts:
f Fact(f).predicate == Fact(e).predicate: print(f'\t{i}. {f}')
```

Output:

```
kb = KB()
kb.tell('missile(x)=>weapon(x)')
kb.tell('missile(M1)')
kb.tell('enemy(x,America)=>hostile(x)')
kb.tell('american(West)')
kb.tell('enemy(Nono,America)')
kb.tell('owns(Nono,M1)')
kb.tell('missile(x)&owns(Nono,x)=>sells(West,x,Nono)')
kb.tell('american(x)&weapon(y)&sells(x,y,z)&hostile(z)=>criminal(x)')
kb.query('criminal(x)')
kb.display()
Querying criminal(x):

    criminal(West)

All facts:
        1. hostile(Nono)
        sells(West,M1,Nono)
        american(West)
        4. owns(Nono,M1)
        5. enemy(Nono,America)
        6. weapon(M1)
        criminal(West)
        8. missile(M1)
```