VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.



LAB REPORT on

Artificial Intelligence

Submitted by

SUSHANTH (1BM21CS227)

in partial fulfillment 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

Nov-2023 to Feb-2024

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" carried out by SUSHANTH (1BM21CS227), who is bonafide student of B.M.S. College of Engineering. It is in partial fulfillment for the award of Bachelor of Engineering in Computer Science and Engineering of the Visvesvaraya Technological University, Belgaum during the academic semester June-2023 to Sep-2023. The Lab report has been approved as it satisfies the academic requirements in respect of a Artificial Intelligence (22CS5PCAIN) work prescribed for the said degree.

Sandhya A Kulkarni Dr. Jyothi S Nayak

Associate Professor Professor and Head

Department of CSE Department of CSE

BMSCE, Bengaluru BMSCE, Bengaluru

Index Sheet

Lab Program No.	Program Details	Page No.
1	Implement Tic –Tac –Toe Game.	1 - 6
2	Solve 8 puzzle problems.	7 - 10
3	Implement Iterative deepening search algorithm.	11 - 14
4	Implement A* search algorithm.	15 - 19
5	Implement vaccum cleaner agent.	20 - 22
6	Create a knowledge base using prepositional logic and show that the given query entails the knowledge base or not.	23 - 24
7	Create a knowledge base using prepositional logic and prove the given query using resolution	25 - 29
8	Implement unification in first order logic	30 - 35
9	Convert a given first order logic statement into Conjunctive Normal Form (CNF).	36 - 37
10	Create a knowledge base consisting of first order logic statements and prove the given query using forward reasoning.	38 - 42

Course Outcome

CO1	Apply knowledge of agent architecture, searching and reasoning techniques for different applications.	
CO2	Analyse Searching and Inferencing Techniques.	
CO3	Design a reasoning system for a given requirement.	
CO4	Conduct practical experiments for demonstrating agents, searching and inferencing.	



1. Implement Tic -Tac -Toe Game.

```
import math
import copy
X = "X"
O = "O"
EMPTY = None
def initial_state():
  return [[EMPTY, EMPTY, EMPTY],
      [EMPTY, EMPTY, EMPTY],
      [EMPTY, EMPTY, EMPTY]]
def player(board):
  countO = 0
  countX = 0
  for y in [0, 1, 2]:
    for x in board[y]:
      if x == "O":
         countO = countO + 1
      elif x == "X":
         countX = countX + 1
  if countO >= countX:
    return X
  elif countX > countO:
    return O
```

def actions(board):

```
freeboxes = set()
  for i in [0, 1, 2]:
    for j in [0, 1, 2]:
      if board[i][j] == EMPTY:
        freeboxes.add((i, j))
  return freeboxes
def result(board, action):
  i = action[0]
 j = action[1]
  if type(action) == list:
    action = (i, j)
  if action in actions(board):
    if player(board) == X:
      board[i][j] = X
    elif player(board) == O:
      board[i][j] = O
  return board
def winner(board):
  board[1][2] == X \text{ or } board[2][0] == board[2][1] == board[2][2] == X):
    return X
  board[1][2] == O \text{ or } board[2][0] == board[2][1] == board[2][2] == O):
    return O
  for i in [0, 1, 2]:
    s2 = []
    for j in [0, 1, 2]:
```

```
s2.append(board[j][i])
     if (s2[0] == s2[1] == s2[2]):
       return s2[0]
  strikeD = []
  for i in [0, 1, 2]:
     strikeD.append(board[i][i])
  if (strikeD[0] == strikeD[1] == strikeD[2]):
     return strikeD[0]
  if (board[0][2] == board[1][1] == board[2][0]):
     return board[0][2]
  return None
def terminal(board):
  Full = True
  for i in [0, 1, 2]:
     for j in board[i]:
       if j is None:
          Full = False
  if Full:
     return True
  if (winner(board) is not None):
     return True
  return False
def utility(board):
  if (winner(board) == X):
     return 1
  elif winner(board) == 0:
```

```
return -1
  else:
    return 0
def minimax helper(board):
  isMaxTurn = True if player(board) == X else False
  if terminal(board):
    return utility(board)
  scores = []
  for move in actions(board):
    result(board, move)
    scores.append(minimax helper(board))
    board[move[0]][move[1]] = EMPTY
  return max(scores) if isMaxTurn else min(scores)
def minimax(board):
  isMaxTurn = True if player(board) == X else False
  bestMove = None
  if isMaxTurn:
    bestScore = -math.inf
    for move in actions(board):
       result(board, move)
       score = minimax helper(board)
       board[move[0]][move[1]] = EMPTY
       if (score > bestScore):
         bestScore = score
         bestMove = move
```

```
return bestMove
  else:
     bestScore = +math.inf
     for move in actions(board):
       result(board, move)
       score = minimax helper(board)
       board[move[0]][move[1]] = EMPTY
       if (score < bestScore):</pre>
         bestScore = score
         bestMove = move
     return bestMove
def print board(board):
  for row in board:
     print(row)
# Example usage:
game_board = initial_state()
print("Initial Board:")
print_board(game_board)
while not terminal(game board):
  if player(game_board) == X:
     user input = input("\nEnter your move (row, column): ")
    row, col = map(int, user_input.split(','))
    result(game board, (row, col))
  else:
     print("\nAI is making a move...")
```

```
move = minimax(copy.deepcopy(game_board))
result(game_board, move)

print("\nCurrent Board:")
print_board(game_board)

# Determine the winner
if winner(game_board) is not None:
    print(f"\nThe winner is: {winner(game_board)}")
else:
    print("\nIt's a tie!")
```

```
Initial Board:
[None, None, None]
[None, None, None]
[None, None, None]
Enter your move (row, column): 1,2
Current Board:
[None, None, None]
[None, None, 'X']
[None, None, None]
AI is making a move...
Current Board:
[None, None, None]
[None, 'O', 'X']
[None, None, None]
Enter your move (row, column): 0,0
Current Board:
['X', None, None]
[None, 'O', 'X']
[None, None, None]
AI is making a move...
Current Board:
['X', '0', None]
[None, '0', 'X']
[None, None, None]
Enter your move (row, column): 2,1
```

```
Current Board:
['X', '0', None]
[None, '0', 'X']
[None, 'X', None]

AI is making a move...

Current Board:
['X', '0', None]
[None, '0', 'X']
['0', 'X', None]

Enter your move (row, column): 1,0

Current Board:
['X', '0', None]
['X', '0', 'X']
['0', 'X', None]

AI is making a move...

Current Board:
['X', '0', 'X']
['0', 'X', None]

The winner is: 0
```

2. Solve 8 puzzle problems.

```
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(0)
  #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('l')
  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 move_it_can in pos_moves_it_can if move_it_can not in
visited states]
def gen(state, m, b):
  temp = state.copy()
  if m=='d':
    temp[b+3],temp[b] = temp[b],temp[b+3]
  if m=='u':
```

```
temp[b-3],temp[b] = temp[b],temp[b-3]
  if m=='l':
     temp[b-1],temp[b] = temp[b],temp[b-1]
  if m=='r':
     temp[b+1],temp[b] = temp[b],temp[b+1]
  # return new state with tested move to later check if "src == target"
  return temp
print("Example 1")
src= [2,0,3,1,8,4,7,6,5]
target=[1,2,3,8,0,4,7,6,5]
print("Source: " , src)
print("Goal State: " , target)
bfs(src, target)
print("\nExample 2")
src = [1,2,3,0,4,5,6,7,8]
target = [1,2,3,4,5,0,6,7,8]
print("Source: " , src)
print("Goal State: " , target)
bfs(src, target)
```

```
Example 1
Source: [2, 0, 3, 1, 8, 4, 7, 6, 5]
Goal State: [1, 2, 3, 8, 0, 4, 7, 6, 5]
[2, 0, 3, 1, 8, 4, 7, 6, 5]
[2, 8, 3, 1, 0, 4, 7, 6, 5]
[0, 2, 3, 1, 8, 4, 7, 6, 5]
[2, 3, 0, 1, 8, 4, 7, 6, 5]
[2, 8, 3, 1, 6, 4, 7, 0, 5]
[2, 8, 3, 0, 1, 4, 7, 6, 5]
[2, 8, 3, 1, 4, 0, 7, 6, 5]
[1, 2, 3, 0, 8, 4, 7, 6, 5]
[2, 3, 4, 1, 8, 0, 7, 6, 5]
[2, 8, 3, 1, 6, 4, 0, 7, 5]
[2, 8, 3, 1, 6, 4, 7, 5, 0]
[0, 8, 3, 2, 1, 4, 7, 6, 5]
[2, 8, 3, 7, 1, 4, 0, 6, 5]
[2, 8, 0, 1, 4, 3, 7, 6, 5]
[2, 8, 3, 1, 4, 5, 7, 6, 0]
[1, 2, 3, 7, 8, 4, 0, 6, 5]
[1, 2, 3, 8, 0, 4, 7, 6, 5]
Success
Example 2
Source:
         [1, 2, 3, 0, 4, 5, 6, 7, 8]
Goal State: [1, 2, 3, 4, 5, 0, 6, 7, 8]
[1, 2, 3, 0, 4, 5, 6, 7, 8]
[0, 2, 3, 1, 4, 5, 6, 7, 8]
[1, 2, 3, 6, 4, 5, 0, 7, 8]
[1, 2, 3, 4, 0, 5, 6, 7, 8]
[2, 0, 3, 1, 4, 5, 6, 7, 8]
[1, 2, 3, 6, 4, 5, 7, 0, 8]
[1, 0, 3, 4, 2, 5, 6, 7, 8]
[1, 2, 3, 4, 7, 5, 6, 0, 8]
[1, 2, 3, 4, 5, 0, 6, 7, 8]
Success
```

3. Implement Iterative deepening search algorithm.

```
def iterative_deepening_search(src, target):
  depth limit = 0
  while True:
     result = depth limited search(src, target, depth limit, [])
     if result is not None:
       print("Success")
       return
     depth limit += 1
     if depth limit > 30: # Set a reasonable depth limit to avoid an infinite loop
       print("Solution not found within depth limit.")
       return
def depth limited search(src, target, depth limit, visited states):
  if src == target:
    print state(src)
     return src
  if depth \lim_{t\to 0}:
     return None
  visited_states.append(src)
  poss_moves_to_do = possible_moves(src, visited states)
  for move in poss moves to do:
     if move not in visited states:
       print state(move)
       result = depth limited search(move, target, depth limit - 1, visited states)
       if result is not None:
```

return result

```
return None
def possible_moves(state, visited_states):
  b = state.index(0)
  d = []
  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('l')
  if b not in [2, 5, 8]:
     d.append('r')
  pos_moves_it_can = []
  for i in d:
     pos moves it can.append(gen(state, i, b))
  return [move_it_can for move_it_can in pos_moves_it_can if move_it_can not in
visited states]
def gen(state, m, b):
  temp = state.copy()
  if m == 'd':
     temp[b+3], temp[b] = temp[b], temp[b+3]
  elif m == 'u':
```

```
temp[b - 3], temp[b] = temp[b], temp[b - 3]
elif m == 'l':
    temp[b - 1], temp[b] = temp[b], temp[b - 1]
elif m == 'r':
    temp[b + 1], temp[b] = temp[b], temp[b + 1]

return temp

def print_state(state):
    print(f"{state[0]} {state[1]} {state[2]}\n{state[3]} {state[4]} {state[5]}\n{state[6]} {state[7]} {state[8]}\n")

print("Example 1")
src = [1,2,3,0,4,5,6,7,8]
target = [1,2,3,4,5,0,6,7,8]
print("Source: ", src)
print("Goal State: ", target)
iterative_deepening_search(src, target)
```

```
Example 1
Source: [1, 2, 3, 0, 4, 5, 6, 7, 8]
Goal State: [1, 2, 3, 4, 5, 0, 6, 7, 8]
0 2 3
1 4 5
6 7 8
1 2 3
6 4 5
0 7 8
1 2 3
4 0 5
6 7 8
0 2 3
1 4 5
6 7 8
2 0 3
1 4 5
6 7 8
1 2 3
6 4 5
0 7 8
1 2 3
6 4 5
7 0 8
1 2 3
4 0 5
6 7 8
```

```
1 0 3
4 2 5
6 7 8

1 2 3
4 7 5
6 0 8

1 2 3
4 5 0
6 7 8

1 2 3
4 5 0
6 7 8

Success
```

4. Implement A* search algorithm.

```
def print_grid(src):
  state = src.copy()
  state[state.index(-1)] = ' '
  print(
     f'''''
{state[0]} {state[1]} {state[2]}
{state[3]} {state[4]} {state[5]}
{state[6]} {state[7]} {state[8]}
  )
def h(state, target):
  #Manhattan distance
  dist = 0
  for i in state:
     d1, d2 = state.index(i), target.index(i)
     x1, y1 = d1 \% 3, d1 // 3
     x2, y2 = d2 \% 3, d2 // 3
     dist += abs(x1-x2) + abs(y1-y2)
  return dist
def astar(src, target):
  states = [src]
  g = 0
  visited_states = set()
  while len(states):
     moves = []
     for state in states:
```

```
visited_states.add(tuple(state))
       print_grid(state)
       if state == target:
          print("Success")
          return
       moves += [move for move in possible moves(state, visited states) if move not in
moves]
     costs = [g + h(move, target) for move in moves]
     states = [moves[i] for i in range(len(moves)) if costs[i] == min(costs)]
     g += 1
  print("Fail")
def possible moves(state, visited states):
  b = state.index(-1)
  d = []
  if 9 > b - 3 >= 0:
     d += 'u'
  if 9 > b + 3 >= 0:
     d += 'd'
  if b not in [2,5,8]:
     d += 'r'
  if b not in [0,3,6]:
     d += '1'
  pos moves = []
  for move in d:
     pos moves.append(gen(state,move,b))
  return [move for move in pos moves if tuple(move) not in visited states]
def gen(state, direction, b):
  temp = state.copy()
  if direction == 'u':
```

```
temp[b-3], temp[b] = temp[b], temp[b-3]
  if direction == 'd':
     temp[b+3], temp[b] = temp[b], temp[b+3]
  if direction == 'r':
     temp[b+1], temp[b] = temp[b], temp[b+1]
  if direction == 'l':
     temp[b-1], temp[b] = temp[b], temp[b-1]
  return temp
#Test 1
print("Example 1")
src = [1,2,3,-1,4,5,6,7,8]
target = [1,2,3,4,5,-1,6,7,8]
print("Source: " , src)
print("Goal State: " , target)
astar(src, target)
# Test 2
print("Example 2")
src = [1,2,3,-1,4,5,6,7,8]
target=[1,2,3,6,4,5,-1,7,8]
print("Source: ", src)
print("Goal State: " , target)
astar(src, target)
# Test 3
print("Example 3")
src = [1,2,3,7,4,5,6,-1,8]
```

```
target=[1,2,3,6,4,5,-1,7,8]
print("Source: ", src)
print("Goal State: ", target)
astar(src, target)
```

```
Example 1
Source: [1, 2, 3, -1, 4, 5, 6, 7, 8]
Goal State: [1, 2, 3, 4, 5, -1, 6, 7, 8]
1 2 3
 4 5
6 7 8
1 2 3
4 5
6 7 8
1 2 3
4 5
6 7 8
Success
Example 2
Source: [1, 2, 3, -1, 4, 5, 6, 7, 8]
Goal State: [1, 2, 3, 6, 4, 5, -1, 7, 8]
1 2 3
  4 5
6 7 8
1 2 3
6 4 5
  7 8
Success
```

```
123
Example 3
                                                                           6 5
Source: [1, 2, 3, 7, 4, 5, 6, -1, 8]
Goal State: [1, 2, 3, 6, 4, 5, -1, 7, 8]
                                                                          478
123
                                                                          1 2 3
7 4 5
6 8
                                                                          4 7 8
1 2 3
                                                                          1 2 3
6 7 5
7 4 5
  6 8
1 2 3
                                                                          1 2 3
 4 5
                                                                          6 7 5
4 8
7 6 8
 2 3
                                                                          123
1 4 5
                                                                           7 5
768
                                                                          6 4 8
123
                                                                           2 3
4 5
768
                                                                          175
                                                                          6 4 8
                                                                          1 2 3
7 5
6 4 8
123
4 6 5
7 8
```

```
713
4 6 5
 2 8
713
465
2 8
713
4 5
2 6 8
713
465
2 8
713
4.5
268
713
2 4 5
 6 8
Fail
```

5. Implement vacuum cleaner agent.

```
def clean(floor, row, col):
  i, j, m, n = row, col, len(floor), len(floor[0])
  goRight = goDown = True
  cleaned = [not any(f) for f in floor]
  while not all(cleaned):
     while any(floor[i]):
       print_floor(floor, i, j)
       if floor[i][j]:
          floor[i][j] = 0
          print_floor(floor, i, j)
       if not any(floor[i]):
          cleaned[i] = True
          break
       if j == n - 1:
          j -= 1
          goRight = False
       elif j == 0:
          j += 1
          goRight = True
       else:
          j += 1 if goRight else -1
     if all(cleaned):
       break
     if i == m - 1:
       i = 1
       goDown = False
     elif i == 0:
       i += 1
```

```
goDown = True
     else:
        i += 1 if goDown else -1
     if cleaned[i]:
       print_floor(floor, i, j)
def print_floor(floor, row, col): # row, col represent the current vacuum cleaner position
  for r in range(len(floor)):
     for c in range(len(floor[r])):
        if r == row and c == col:
          print(f'' > \{floor[r][c]\} < ", end = ")
        else:
          print(f'' \{floor[r][c]\} '', end = '')
     print(end = '\n')
  print(end = '\n')
#Test 1
floor = [[1, 0, 0, 0],
     [0, 1, 0, 1],
     [1, 0, 1, 1]]
print("Room Condition: ")
for row in floor:
  print(row)
print("\n")
clean(floor, 1, 2)
```

```
Room Condition:
                                                         1
                                                              0
                                                                         0
                                                                    0
[1, 0, 0, 0]
                                                         0
                                                              0
                                                                    0
                                                                         0
[0, 1, 0, 1]
                                                        >1<
                                                              0
                                                                    1
                                                                         1
[1, 0, 1, 1]
                                                         1
                                                              0
                                                                    0
                                                                         0
                                                         0
                                                              0
                                                                    0
                                                                         0
                                                        >0<
                                                              0
  1
        0
             0
                    0
  0
        1
            >0<
                    1
                                                              0
                                                                    0
                                                                         0
  1
        0
                                                         0
                                                                    0
                                                              0
                                                                         0
                                                         0
                                                              >0<
                                                                    1
                                                                         1
  1
        0
             0
                   0
        1
              0
                  >1<
  0
                                                         1
                                                              0
                                                                    0
                                                                         0
        0
  1
              1
                   1
                                                         0
                                                              0
                                                                    0
                                                                         0
                                                         0
                                                              0
                                                                   >14
                                                                         1
  1
        0
              0
                   0
                                                                         0
                                                         1
                                                              0
                                                                    0
  0
        1
              0
                  >0<
  1
        0
              1
                                                         0
                                                              0
                                                                    0
                                                                         0
                                                         0
                                                              0
                                                                   >0<
                                                                         1
  1
        0
             0
                    0
                                                                    0
                                                                         0
                                                         1
                                                              0
        1
                    0
  0
            >0<
                                                         0
                                                              0
                                                                    0
                                                                         0
        0
                    1
  1
              1
                                                         0
                                                              0
                                                                    0
                                                                        >1<
  1
        0
                    0
              0
                                                         1
                                                              0
                                                                    0
                                                                         0
  0
       >1<
              0
                    0
                                                         0
                                                              0
                                                                    0
                                                                         0
        0
  1
              1
                    1
                                                         0
                                                              0
                                                                    0
                                                                        >0<
  1
       0
              0
                    0
                                                         1
                                                              0
                                                                    0
                                                                         0
                                                                        >0<
                    0
                                                         0
                                                              0
                                                                    0
  0
       >0<
              0
                                                         0
                                                              0
                                                                    0
                                                                         0
  1
        0
              1
                    1
                                                         1
                                                              0
                                                                    0
                                                                        >0<
  1
        0
              0
                    0
                                                         0
                                                              0
                                                                    0
                                                                         0
  0
        0
              0
                    0
                                                         0
                                                               0
                                                                    0
                                                                         0
  1
       >0<
              1
                    1
```

```
1
       0
            >0<
                    0
       0
             0
 0
                    0
 0
       0
             0
                    0
 1
      >0<
             0
                    0
 0
       0
             0
                    0
 0
       0
             0
                    0
>1<
       0
             0
                    0
0
       0
             0
                    0
0
       0
             0
                    0
>0<
       0
             0
                    0
0
       0
             0
                    0
 0
       0
             0
                    0
```

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

```
def evaluate expression(p, q, r):
  expression result = (p \text{ or } q) and (not r \text{ or } p)
  return expression result
def generate_truth_table():
  print(" p | q | r | Expression (KB) | Query (p^r)")
  print("---|---|---")
  for p in [True, False]:
     for q in [True, False]:
       for r in [True, False]:
          expression result = evaluate expression(p, q, r)
          query result = p and r
          print(f" {p} | {q} | {r} | {expression_result}
                                                        | {query result}")
def query entails knowledge():
  for p in [True, False]:
     for q in [True, False]:
       for r in [True, False]:
          expression result = evaluate expression(p, q, r)
          query result = p and r
          if expression result and not query result:
            return False
  return True
```

```
def main():
    generate_truth_table()

if query_entails_knowledge():
    print("\nQuery entails the knowledge.")

else:
    print("\nQuery does not entail the knowledge.")

if __name__ == "__main__":
    main()
```

```
KB: (p or q) and (not r or p)
            Expression (KB)
                             Query (p^r)
              True True
                                          True
              False
                     True
                                           False
       False
True | False
               False | True
                                            False
 False
       True
               True | False
 False
        True | False | True
 False
        False | True |
                      False
                                             False
 False | False | False | False
Query does not entail the knowledge.
```

7. Create a knowledge base 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 = 1for step in steps: print(f' {i}.\t| {step}\t| {steps[step]}\t') i += 1def 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') 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 i != 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 } 1 \text{ if } t != c]
             t2 = [t \text{ for } t \text{ in terms 2 if } t != negate(c)]
             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 {temp[i]} and {temp[i]}.'
       j = (j + 1) \% n
     i += 1
  return steps
rules = 'Rv\sim P Rv\sim Q \sim RvP \sim RvQ' \#(P^{\wedge}Q) \le R : (Rv\sim P)v(Rv\sim Q)^{\wedge}(\sim RvP)^{\wedge}(\sim RvQ)
goal = 'R'
print('Rules: ',rules)
print("Goal: ",goal)
main(rules, goal)
rules = PvQ \sim PvR \sim QvR' \#P=vQ, P=>Q : \sim PvQ, Q=>R, \sim QvR
goal = 'R'
print('Rules: ',rules)
print("Goal: ",goal)
main(rules, goal)
rules = 'PvQ PvR \simPvR RvS Rv\simQ \simSv\simQ' # (P=>Q)=>Q, (P=>P)=>R, (R=>S)=>\sim(S=>Q)
goal = 'R'
print('Rules: ',rules)
```

```
print("Goal: ",goal)
main(rules, goal)
```

```
Example 1
Rules: Rv~P Rv~Q ~RvP ~RvQ
Goal: R
        |Clause |Derivation
Step
         Rv~P
                  Given.
1.
         Rv~Q
                  Given.
3.
         ~RvP
                  Given.
4.
         ~RvQ
                  Given.
                  Negated conclusion.
5.
         ~R
                 Resolved Rv~P and ~RvP to Rv~R, which is in turn null.
A contradiction is found when ~R is assumed as true. Hence, R is true.
Example 2
Rules: PvQ ~PvR ~QvR
Goal: R
        |Clause |Derivation
Step
1.
         PvQ
                 Given.
                  Given.
         ~PvR
                  Given.
3.
         ~QvR
         ~R
                  Negated conclusion.
4.
                 Resolved from PvQ and ~PvR.
5.
         QvR
         PvR
                 Resolved from PvQ and ~QvR.
6.
         ~P
                  Resolved from ~PvR and ~R.
 7.
                  Resolved from ~QvR and ~R.
8.
         ~Q
                  Resolved from ~R and QvR.
9.
         Q
         Р
                 Resolved from ~R and PvR.
10.
11.
         R
                 Resolved from QvR and ~Q.
                 Resolved R and ~R to Rv~R, which is in turn null.
12.
A contradiction is found when ~R is assumed as true. Hence, R is true.
```

```
Example 3
Rules: PvQ PvR ~PvR RvS Rv~Q ~Sv~Q
Goal: R
Step
       |Clause |Derivation
 1.
          PvQ
                  Given.
          PvR
                  Given.
 2.
                  Given.
         ~PvR
 3.
         RvS
                  Given.
 4.
                  Given.
 5.
          Rv~Q
 6.
          ~Sv~Q
                  Given.
         ~R
                  Negated conclusion.
 7.
 8.
         QvR
                  Resolved from PvQ and ~PvR.
                  Resolved from PvQ and ~Sv~Q.
 9.
         Pv~S
        l P
 10.
                  Resolved from PvR and ~R.
 11.
          ~P
                  Resolved from ~PvR and ~R.
 12.
         Rv~S
                  Resolved from ~PvR and Pv~S.
 13.
         R
                  Resolved from ~PvR and P.
                  Resolved from RvS and ~R.
 14.
          S
                  Resolved from Rv~Q and ~R.
 15.
          ~Q
 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. Implement unification in first order logic

```
import re
def getAttributes(expression):
  expression = expression.split("(")[1:]
  expression = "(".join(expression)
  expression = expression[:-1]
  expression = re.split("(?<!\(.),(?!.\))", expression)
  return expression
def getInitialPredicate(expression):
  return expression.split("(")[0]
def isConstant(char):
  return char.isupper() and len(char) == 1
def isVariable(char):
  return char.islower() and len(char) == 1
def replaceAttributes(exp, old, new):
  attributes = getAttributes(exp)
  for index, val in enumerate(attributes):
     if val == old:
       attributes[index] = new
  predicate = getInitialPredicate(exp)
  return predicate + "(" + ",".join(attributes) + ")"
def apply(exp, substitutions):
  for substitution in substitutions:
```

```
new, old = substitution
     exp = replaceAttributes(exp, old, new)
  return exp
def checkOccurs(var, exp):
  if exp.find(var) == -1:
     return False
  return True
def getFirstPart(expression):
  attributes = getAttributes(expression)
  return attributes[0]
def getRemainingPart(expression):
  predicate = getInitialPredicate(expression)
  attributes = getAttributes(expression)
  newExpression = predicate + "(" + ",".join(attributes[1:]) + ")"
  return newExpression
def unify(exp1, exp2):
  if exp1 == exp2:
     return []
  if isConstant(exp1) and isConstant(exp2):
     if exp1 != exp2:
       return False
  if isConstant(exp1):
```

```
return [(exp1, exp2)]
if isConstant(exp2):
  return [(exp2, exp1)]
if is Variable(exp1):
  if checkOccurs(exp1, exp2):
     return False
  else:
     return [(exp2, exp1)]
if is Variable(exp2):
  if checkOccurs(exp2, exp1):
     return False
  else:
     return [(exp1, exp2)]
if getInitialPredicate(exp1) != getInitialPredicate(exp2):
  print("Predicates do not match. Cannot be unified")
  return False
attributeCount1 = len(getAttributes(exp1))
attributeCount2 = len(getAttributes(exp2))
if attributeCount1 != attributeCount2:
  return False
head1 = getFirstPart(exp1)
head2 = getFirstPart(exp2)
initialSubstitution = unify(head1, head2)
if not initial Substitution:
```

```
return False
  if attributeCount1 == 1:
     return initialSubstitution
  tail1 = getRemainingPart(exp1)
  tail2 = getRemainingPart(exp2)
  if initialSubstitution != []:
     tail1 = apply(tail1, initialSubstitution)
     tail2 = apply(tail2, initialSubstitution)
  remainingSubstitution = unify(tail1, tail2)
  if not remaining Substitution:
     return False
  initialSubstitution.extend(remainingSubstitution)
  return initialSubstitution
print("\nExample 1")
exp1 = "knows(f(x),y)"
exp2 = "knows(J,John)"
print("Expression 1: ",exp1)
print("Expression 2: ",exp2)
substitutions = unify(exp1, exp2)
print("Substitutions:")
print(substitutions)
print("\nExample 2")
exp1 = "knows(John,x)"
```

```
exp2 = "knows(y,mother(y))"
print("Expression 1: ",exp1)
print("Expression 2: ",exp2)

substitutions = unify(exp1, exp2)
print("Substitutions:")
print(substitutions)

print("\nExample 3")
exp1 = "Student(x)"
exp2 = "Teacher(Rose)"
print("Expression 1: ",exp1)
print("Expression 2: ",exp2)

substitutions = unify(exp1, exp2)
print("Substitutions:")
print(substitutions)
```

OUTPUT:

```
Example 1
Expression 1: knows(f(x),y)
Expression 2: knows(J,John)
Substitutions:
[('J', 'f(x)'), ('John', 'y')]

Example 2
Expression 1: knows(John,x)
Expression 2: knows(y,mother(y))
Substitutions:
[('John', 'y'), ('mother(y)', 'x')]

Example 3
Expression 1: Student(x)
Expression 2: Teacher(Rose)
Predicates do not match. Cannot be unified
Substitutions:
False
```

9. Convert a 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]+\backslash([A-Za-z,]+\backslash)'
  return re.findall(expr, string)
def Skolemization(statement):
  SKOLEM CONSTANTS = [f(chr(c))') for c in range(ord('A'), ord('Z')+1)]
  matches = re.findall('[\exists].', statement)
  for match in matches[::-1]:
     statement = statement.replace(match, ")
     for predicate in getPredicates(statement):
        attributes = getAttributes(predicate)
        if ".join(attributes).islower():
          statement = statement.replace(match[1],SKOLEM CONSTANTS.pop(0))
  return statement
import re
def fol to cnf(fol):
  statement = fol.replace("=>", "-")
  expr = ' \setminus [([^{\wedge}]] + ) \setminus ]'
  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

return Skolemization(statement)

print(fol_to_cnf("bird(x)=>~fly(x)"))

print(fol_to_cnf("dx[bird(x)=>~fly(x)]"))

print(Skolemization(fol_to_cnf("animal(y)<=>loves(x,y)")))

print(Skolemization(fol_to_cnf("∀x[∀y[animal(y)=>loves(x,y)]]=>[∃z[loves(z,x)]]")))

print(fol_to_cnf("[american(x)&weapon(y)&sells(x,y,z)&hostile(z)]=>criminal(x)"))
```

OUTPUT:

```
Example 1

FOL: bird(x)=>~fly(x)

CNF: ~bird(x)|~fly(x)

Example 2

FOL: ∃x[bird(x)=>~fly(x)]

CNF: [~bird(A)|~fly(A)]

Example 3

FOL: animal(y)<=>loves(x,y)

CNF: ~animal(y)<|loves(x,y)

Example 4

FOL: ∀x[∀y[animal(y)=>loves(x,y)]]=>[∃z[loves(z,x)]]

CNF: ∀x~[∀y[~animal(y)|loves(x,y)]]|[[loves(A,x)]]

Example 5

FOL: [american(x)&weapon(y)&sells(x,y,z)&hostile(z)]=>criminal(x)

CNF: ~[american(x)&weapon(y)&sells(x,y,z)&hostile(z)]|criminal(x)
```

10. Create a knowledge base 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 = ' ([^{\wedge})] + )'
  matches = re.findall(expr, string)
  return matches
def getPredicates(string):
  expr = '([a-z\sim]+)\backslash([^{\wedge}\&|]+\backslash)'
  return re.findall(expr, string)
class Fact:
  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
     1 = 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])
     for key in constants:
        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()
  def tell(self, e):
     if '=>' in e:
        self.implications.add(Implication(e))
     else:
        self.facts.add(Fact(e))
     for i in self.implications:
        res = i.evaluate(self.facts)
        if res:
          self.facts.add(res)
  def query(self, e):
     facts = set([f.expression for f in self.facts])
     i = 1
     print(f'Querying {e}:')
     for f in facts:
        if Fact(f).predicate == Fact(e).predicate:
          print(f \setminus t\{i\}, \{f\}')
          i += 1
```

```
def display(self):
     print("All facts: ")
     for i, f in enumerate(set([f.expression for f in self.facts])):
       print(f'\setminus \{i+1\}, \{f\}')
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()
kb = KB()
kb_tiell('king(x)\&greedy(x)=>evil(x)')
kb_.tell('king(John)')
kb_.tell('greedy(John)')
kb_.tell('king(Richard)')
kb .query('evil(x)')
```

OUTPUT:

Observation Book

	Python		papergrid
Shing f	amed King		
Piles a seq	4		
4×1 = "	The price is 14 .	lograms"	
	M. format (prus)		
20.243.000	by plus with 2	desimals	
4×1=	"The paid to X	:. 21 y dossam"	
moltiple	Nation :		
Point C	1x1 - formest (Prose	, themas, teent))
FA! 9W	antity = 9		
130	mne - 56-9		
Pri	1 = 49		
my	ardy = " I won!	Ky Pierra of the	number 3
U	for (= 2+)	dalton"	
Fin	of (myerder-forms	t (quantity, shown	ry prite 3.)
Jedis n	umber -		
quantity =	3		
ilimne - 5	49		
Pau - 50			
Dryanda 7	" I wont Log	piccus or item	rumber 414
d -for	2:-214 doss	or".	
Post (no	gader-former (g	unlity , Hemo , P	ru))
It else.			
Equation	0 = = 0		
Not negati	u a1=b		
Less Hom			
	or equal to a K=b		
Questo to	des em		
	have or found to a	to	

```
papergrid
                                       Date: 1 7
(x 0=3)
     b=200
     if boa:
     Print (" b is greate than a")
Ex! 0= 33
   b=33
    if 6>a:
    Port ( 16 is greater there or)
   ell a==h:
      Print ("a and 6 are equal")
Ex. 0-330
  6- 970
   Print ("A") if ook else print ("=") if a==b eix
                              Post (*8")
  White Loop
    1=1
    while (<6
    Pan (i)
     i #=1
     break statement
     1=1
     while I KE:
      Print (i)
       If i==3:
        break
        1+=1
```

1 10	papergrid
	Date: / /
	For Juops
	-traits - Corres "bonone" changes
	ter ox in faults:
	floot (x)
	Deoping 15 mough 3 felog
	for a in "bonnono"
	print (x)
	range () Function:
	dor s in range (6):
	Print (3) 0 1 2 9 4 5 0
	To: " in ronge (2,6).
	Part (a) 2 3 4 5
-	CAR
	the second secon

	papergrid
Tic-Tac-Toe	Onte: 7 J
while a pregram to attimular the Ton - I	ve gore
det part board (board):	
for you in board:	
Print ("1", join (row))	
(p ×:" -") tong	
det cher-evenus (beard-proyet)	1 10 1
for 1 in 12mg 3	
if au (board[i][]) == proyer for or on (board[i][] == proyer for	in month)
er ou (board [I] [] == proyer for	i in range (1)
orhan Thu	9 112
if au (bood [i][i] == progre for	i en tometall
au (boord[][2-1] = - progre for)	in waster
refum Thee	- noge-2
repun Fore	
det is begin full (board):	
Thun ou (board[J[J] = " " to	F 10 Va man (55)
for j in range(22)	I in runge (1)
det get-movers;	
who mo	
fry -	
prove = Int Cirput C" Entre your	man (100 7: ")]
it 1 <= mov <= q:	10000 100 10 11
refun move	
Clyc:	
for (" Invald , red")	
except voius From	
port Tovaid)	

```
def fir for toe (1-
   board - [[" " for _ in rong(s)] for - in range(i)]
    Current_player = "x"
   white The
   Print-board (board)
     move = ger-movers
     70w, col = (move-1)//3, (mov-1)/3
     if board [raw][col] == 0""
        bound [row] [cus] = Comert player
     . If there winner (bread word proper):
      Part - board (board)
        Part ( "It is a tre (")
        ely =
        common plant = "0" Il amon plant = = "x"
             CUE "X"
       cins
          find (" Frivaid move, we compared")
  If _nom_ == "_mole_" =
       tic teres for Co
 Ocupar.
enter your move (1-9)
                XI
                colly your mov flower & ouise
entr your may
XII
                enter your mou
coles your mou
```

	papergri
8-Auzzu Prehum	Date: / /
tent a program to Ferre & periods probe	ion.
from Collections impare chapits	
clan puzzla S:	
det - init - (Seek , Sice = 3);	
Self white = Size	
del was a stall was	
del diprograma (sett. Weete):	
for i in range (c. lin (steels) proof ("their Li-Li + Sell-si	
fmff)	(43)
elet ger-bloom-index (sell, steats):	7
MAIN State rocks (-1)	
dy got-rughboun (sal, min):	
nighten; b, r, c = [], Sell-gol.	- home indu.
divinual (sal gra-hiant ind	be (Shuff) sell
800	3
for m in [(0,1), (1,0), (0,-1), ((1,0)];
]m . (0]m + (0]T = 10 m	J+3[J
14 01= DT < 8411 - SIU '01	d okanck
ru = stax []	
ns[6]. As [nr * self that.	. 7 .
Cont. 2013 - 214 - 214	THE = 14[1
Rest four oppose (ns)	m(6)
istum neighborn	
dit in great state (Sell , state , 1	Manager III and a second
section State = tangel Hack	sugget - Steete) :
ded by Gent interm- view, torget	· strette):
9 , v = clyue ([Linimu=1	July [717]
Section	11/

```
papergrid
                                       Date: 1 1
   while q:
       0 , p = 9 , poplet ()
     If sollies your stak (cs. torget-ship):
        Sthur P
    IF tupes (11) not in v
        v. odd (tures (a))
        go refind ((n, p - [n]) for an in salfages
               ruly (bour (65))
       ittum mon
    iniHUL-1104 = [1,2,3,4,-1,5,6,718]
   gow-stub= [1,0,3,4,5,0,7,3,-1]
   PURZEL + SOLUTIONS PURZER (1. PLESSEL) (1. PLES
                 WHIN , state, gow-states
  if Solution :
     Part ( "Selection foured ")
      for sten, that in chamoras (suumon):
          part (1" step & step + 17 - 11)
         14234 Staping - Melle (Mode)
     Clar
          PHAT ("No JOHNHAM FOUND")
Colput
         Sten :
pleps:
                  SHIS.
                          Ster7
                                  Strg.
                                          she 11
 123
         113
                  1/2/3
                          123
                                 1 23
                                          1 27
 45-1
         453
                  -153
                          563
 62 8
                                 54-1
                                          h 5 6
          6-17
                  467
                          474 478
                                         9 -15
Steps
         Show
                Ship 6
                         SHIB
                                 Stepic
                                         3 Mp12
123
         123 123
                                 123
                         123
                                         123
4 53
         8 2 4
                 5 41 5
                         568
                                 546
                                         456
67-1 -169
                 467
                         478
                                 475
                                         7 3-1
```

```
papergrid
          Vareum chone Agent
                                             Date: / /
det vacum-com Of
  gew. Hat = ('A" . 'o', 'B' 'o')
  Isration input a const (" Entry Accounts of Value of
  Status-input = input ( " Enfer Status of "+ Servition input )
  Stedus-input-computed = logis (" Eater States a other man")
 print " Intell documen condition" + str (quee state ))
 if lumber intol = " " :
      Print ("Vacanim is plosed in Loration A")
     it Stote innet = = " !!
         Print ( Location A is Dirty 1)
          goen_ Had ['A'] = 101
          cost +=1
          Print ("too for choning n"+ sh(cos))
         PHOT ("LOWERT ST has been chance")
     If Status incur_congument == "1";
        Mini ( "Lorano Ris ding ")
         Pitty ("moving right to the soretion go)
         EST 4=1
        PART ("CON For MONNY Hight "+ SH (con))
       900-Stok ['B'] = '0'
        Lin) +=1
         PART ("COIT for Suck "+ Str (COI))
         PHINI "LOIGHON B has been chouse")
         PHRY ("pu action"+ Six (con))
        the ( Lucusin Bis almost claus)
    if Status-inpug = "O":
        PART ("LOIGHER A IS GIVING Cher")
```

Date / / if status input complaint == "1"; Print (*(oraken Bl. disty ") Print (moving tight to the breather By) Cos1+=1 Print (" (as for many Right "+ sm (tall)) goalshop [18] = 'o' Ce2) 4=1 PHAT ("TEXT FOR SUCK "+ SW (B) Danie (aux) Frist (" Location As already clear ") 21501 Pant (Value 12 ploud in suitable gu) if Status topus == "/"; Pant (Lorotton Bis dirty ") 90W-3/cm [10] = 101 CO1++=1 PHAT ("100 for cloning" + stillow)) Part ("Recation & how been chosen") if 3/2/as tope - comprimes = = 1/10 PHATE ("LE COHON A 12/0 My") Ather "moving left to the Jorden " ") Cost +=1 Prior 1 " car for moving left "+ success good_State [PAI] = 61 (01) +=/ Part (Voor for such "+ str (rath)) PHAI ("Lucation A horber Wound) eur: Print (cost) PHOL ("Lecour Bis Olrody coon") il Statu singut-compliment == 1:

	papergrid
	Date: / /
	print ("Lurragion in its printy")
	part ("moving Left to the Jordan #19)
	(3)) + = 1
	proff"cor for moving 1 cll "+ 370 (cos))
	guar-son ['A') - 'b'
	Con +=1
	fant ("row for sink "+3h (cos))
	enn ("Lucurum or han ham classed")
	DWC:
	PHOLL NO GLAND "+ STE (COST))
	PART (" Execution A is already claim")
	price ("God State:")
	Print (gow- state)
/	print (" performance inconversant "+ ser(clist))
	outeut-
	Enter Lecution of Vaccium A
V	False Stephen of A 1
10	Enky slapp of other from o
1.11	Fixed Lorentran condition ("A": "" B's "O" }
	Valuum is ploud in solution a
	Louisin A is dirty.
	cost for closury A I
	Location of har been chorned
	No action 1
	Lication B. Is observely those
	Gove Street
-	LIAN 1011 181:1017
	Perhimony many months

papergrid papergrid	popergrid
	II ment 12 aug Poure
& presentation with theorem deservey book	action mark
(to, (e-ger))	when been
eld thereby chapter y time (in, toger);	
and the same of th	del paper (cent , want there)
with - differential short (in , I may)	Estable - date - da
diffication (a)	4-17
If must not need	# 6 m/ in Touris
Print Contra 2	diapper (v)
"Hha!"	# 6 mm (5 (5, 3, 37)
glopth = Head + = 1	a office ('d')
the clipth-dimit > 80	# 6 not in [1 8 1 2]
part (" rest found");	displied ("I")
31 A 100	it to the land and the same and
	d approx (T')
det depte some de south (son, tonger , dieth	
Drail, Visit de Malus)	dit gen (stor, m.b)
of second larger	Time = Vietr (conft)
Print, Mak (Sri)	11- m== 'd'
Harr SV	from [11=1] start [1] = tracti].
of dieth-kind == 0)	- lemp [hell]
schim mine	CIA # = 2 "4":
valued_states, agrand (see).	-frant(4-5), Lamp(1) = Lamp(1)
post moves - horse pulling moves (124)	-temp [+]
Violed - Vertes):	and meetaly
VIII	for [4+1], torp[1] = +100[
for moves in past-moves feeds:	
	for (40)
If more not in wasted allotte.	Schur Ami
pilet - Nate Imore)	
remove durity smoothly smouth	det port - stub (stor)
reger . Hongs , depts_simus-1,	Imil 1' Litar-Toly Sign Ed
Visited_state)	Statisty) Litarity (Trave La y Jacon
	(Shintes) \ Trans) \ " "
	Quit.

	popergrid	papergrid
		WYUK 5
Sec = [2,24,3,	0.2.1 1.12.17	Best Fred Hearth
Auropes = Essa in	h 1, v, 6, 3, 27	Chartes 60 Gue somet graph as his but it
		Count the mail generally high hand he
hinko-dumog - l	mak (Are theight)	hearth funker a full nationally districtly
		about bout by terral approach
output 1.2	1 113	The same of the sa
(D 4.3		Friends Grown 1955 was provide grown for provides
		market thanks are record to the second
6 1 1 13	3 173	completes but compate, brane of referrit Aug.
	127 103	is Kennih Ambor party danged
263 133	DAS HIS	
1 H 5 6 H5	0.77	Dil Aphrolit
(3) (7)	7.03	Shale Siv & Note way Alls
107 17	122 121	· Deline the stack representation
h 2 f h 3		. Delive to initial and great studies
691 603		- suppression and up down myst lists
93. 300	Turvey.	Jagelment Homeks Function : comment mathan
/	3.0.00	distant of each his such year person
		Sometic county for Printing described by
		harrie tooler
		Manketten disense
0/13 10	1 /4-5	Car le Lourd by company mystered from with
1 1 3 A 5 3 A	25	your company sometimes from with
6 3 8 6 3	.5	monthstan divini
		al _ l
		a ratus
		N EAN

Des 1	popergrid
Surrey & pour polar may (Gov. First best)	A* algoritor, trought
Street & Days tayou and	H* digitality
prosent grow or E Con D . Ch. N.O. [Trit. C]]	As deposite sents as
MANUAL DISTRICT	
	tree ned.
attending = gray	To takende their patter for step at a least
	F. Carbonals Co. a.
det brante varia (Grant):	Stinited
1	-1(c) = g (c) + k(c)
A company (for (post)) *	and Course and the course
T. 1 (600 (800) L+12 1)	her commended to chapt park him I be y
o postalenos del	puid.
Cerel	
*(Apr (7)_	P For are, dy in [feel] (6.7) (60) (-1.0)]
dit grandrati (months)	Posts (- Obs -) = (+ Ob) (- Ob)
The state of the s	of all of family but the
The same of the same Last I.	that Trans [1] Let you in commy strate.
it constanting and	to be a little output at the lease of a little output
offer () All	new of the Council Council, now the [I]
Man S. Land	of the (map (myse minus)) were much
6 / N = 600	to be a fil terminagent Come with a mornel
of the county (122) -> best . 15 box 0 6 = 163 and 0 6 = 3 \$1	(A. J. C. (A.
Ithm Comments	struc -t
des por (sich , gies) -> 1911 -	State = [[1:40], [n.50], [3,018]]
	more = 950 (Han . Mar)
Valid - sites	
py = 6, pointly quest)	the man 2 = 112
Pr Pur ((Harmin man Grant System))	From (" no easy to small grow that")
white not specimely 15:	Eiy-
- y makes, Committee - payer ()	from (" feedful or" + Str. (must) - "more")
If (worther = Feat.	OLPIGE.
Hour makes	
of the (may (they, arismother))	Rundation 1 mays
Count with - ger count hate (amount)	
1. J= Commun [o], Congrantil	
STATE OF THE PARTY	

	poperund	papergid
-	At surfa	gran misk
-	Swing & perile parkett way no superfeet	
	2007	it has been been connected the send
		.contrast.
	per - But all Fest All (see all	
	dis some Creez:	yours and free (ran Chen your county)
-	GG: 1000G-17062	Salamber S ger condenses (Lames Hale)
-	Mark Sup T Typed	1 = constructe fil, not rate ld
+	the manufact	
	dit the philosof redit	for dely instant land took total
	Mar Carlo	10 May 100 - 1 - 1 - 10 - 1 - 12 - 12
-	As a roy (on God)	Turant (re-1, recet)
-	to see you (in (particil))	Now - State = [prod [-] for your in case
-	A gentled ; - some total.	now world all more than I man all ment of
-	CH+14	Trou Hour Jan sur [1]
	Firmen and	Chest of Sec. 13. Sec. 13.
		of topologo Chronica state and revented
	did gabourd a sales (pagent lists)	Private (Citacanta varia (m. 1844) + 10
	for it in respectations)	from as a temperature)
	A. J. in rouge (without) (I).	If how Ind - wynd,
=	A constantilled = co	ford (env strek)
	series (1-2)	7 200-1
	- Constant	
-	A word (, j) - s bead	Street = (1 1 4 10) [41 10 2] [1 2 3 2 1]
-	who rester on rester	prove Anthor (Man, paid)
-	Phir Contract Contract	private parties (result, page)
-		II MONTES 4
_	der have (seek, year) - son :	Print(" No coup"))
	Vail-1 = 2001	NA CANA
	pg the Printing quarter	13 poul puches in a "in Course " The
	ps. No ((feather valid (text), e, Steam))	
		Outries
	who rathy analy it:	Purhet in I men
	-, man, conviction = 19,900()	
	If copies that = gray.	
	W. Andrea Trefe : = - dem s	

	Wifet 6 papergrid	popergrid
*	Could be said the said topological digital bost to the said place going terms that beautidge bost to	it garp-orthogonal-types from (" forther extension the terrorisitys") life from (" forther since out pather the terrorisitys")
	the commence of (2, 1, x) the commence of (2, 1, x) col p) and p and p or of (2, x) x the commence of	Parts.
	elu genera tra tibo (). pero (*a la la la terraria (*a) progres () pero (* - 1(-1	Sury coints the bounding
	for you true folials for you true track Is. for you to true true.	
	suppressed = to make expense (1.9.7) quey rout = t entr (4 - (4x / (4x / (4x / (4x mm) routs) Lyung routs) =)	
	and ging colors. Principal 1: For gint ton print : For you trong parts	
	The product that a Common consistion (4.6.4) Transportation = 7	
	of Representation and Gung Scouts Action 1669 Action 1669	
	del resurce to the toke ()	

MEEK - 19/10/10	papararis
Creek a branchyddor heleg Proportion of the	Particulation - Cd Construction County
Ord year to give 1977, they would be	A Charles Charles I amend I am
the board of the state of the s	water clarge or circle plates to place (three)
Page 12	an Control of the last
and mar (som-good)	did now (com ; god)
man Paper and C. C.	And to sale sury
The state of the s	fore - from (seed)
A STATE OF THE PROPERTY OF THE PARTY OF THE	Stort aled ()
Face (In Day 1 Colore 1) Department 12	go sour forgi
Proof le tie 11	Story [name] = "Store"
Part (-, 1 = (a)	Smile grid (god) - "wood as done
Not 1	1-8
du tie is 190	Esta is for female
100 14 1 15 7 12 / 21 10 5 1 1 2 a most 2 40 16 7	h = ten (frag)
Treef 4 DAY SEA CONTRACTOR CONTRACTOR	4 = (111) 1.0
(7.1)	
	claum =L3
alch super (hom)	where when j' to 1 1
without I a face y' it them (v) = "w" can from (v)	from (= Spir-time [timp [J])
	From 2 = April Latina Potence [2])
del resear (coor)	for the from 12
if his (sheet) >2.	If region Cot. (n. hom?)
A = 900 Auro (1200)	GINT & for a sen front of
simm t, ittle In (110),	fin (the tinford is t
years of	toyne (
	111 - At -12
del tree from land	Jul - 41 - 12
Cx+ - (([roxs]).	the goods a more land
prime = ne-finden (egg, rent)	class = IC (go b)
	Classic Co. Carrier
MACHE - Sumi	cost synth?
Davis Duras	d release to 112
Option (- Fox)	it when we (for it is a digital it is
[me, ted	Ago MC2
dit zashoslickim (grav jahrana))	

popergrid	papergrid
try open (1 (gently (gentl))) and there (1) to	Provide (parties &)
provided the second of the sec	State Claim Demantion State Claim Demantion
Name State	2 EVAD GIVE 2 NEVP GIVE
Chair += [1 (pole))	1 MR REPOSED CONSTRUCT ON REVENUE REV
time spend (1 (him () V (house))) Store [] = 1 " no and (house) and from []] h (house) } while it	asked in him now a conference is found when he was a stand on the second of the second
turn our In A coloration is found when English (County) is count as four Many, Egous is	
true = referent 21/01	37
to claim to the mangered down a scene (there)	
Ges more (char) set to forp: House sport (char)	
Mire (start) = f " family for Lkoy (3)	
5-4 (3-7 × n	
ET 661	
test = "KV-F KV-B ~ KVF ~ L+6"	
How (run, feet)	
AND AND WAS GOOD	
main (No 1914)	

West 5 popergra	papergrid
Fernand Andrakos in hist saler logis	Michael Candesce to Barro
- Transfer of the second	
to harbor is to pour or delepting whollow there by	The fixed I - Fixed by spirit, to Hand beating
to higher is to your or direct the male advant by	acres place
Opening the best of their velocities opening the Committee of their velocities	
All the second	singer = Sign vehic (1) June (1)
The second second	Cagal - Cogne valled (2) - (10)
PLANE 2) (x40, x41(6), x41)= x/0.0.1(6)(2)	
	Chirelyn - Ch
ayon live:	
I CAL BY	thery agreed
O. H. or or Paris vaiche / contest	Ly or on a zip orgit pigal
La of the and the are substitut	if all islams and as insome is and all
	Substation [av] = 62
PECE & Al Ottom in 02 return	Clif Oli Ellewith C) and out O2: 15 Square (1)
Fish	Champutan [m] ====
Lotte When (BE/M)	Clif with collisioned (1 and as is amore)
Lachus for D2 or D1	Schuttphon [mJ=91
Lafel II Az (Um = n)	Clif all P. Cl.
Locan serm (11/4)	park trapement sower ho works "
@ it pulled by lone)	Seem one
& 1+ dill-eigenst	strong substitute
(1) Elle subst to Will	
& host	dit comy substitutes (rope, substitutes)
@ Pater Solis	for tay, value in publishmen - through
	Eagl = capt - replace (ely , y scor)
code	Johan Capa
Jet undy (spir, cap)	
all god copenies into function and arguments	Horas Carron
front, again = copy sport (40 1)	the man company
Source ages 2 come you (C. 1)	
1902 1 491 - 174 AVE 1 1 1 1 1	Copy = mas (rate fin) capuse)
	Cappe japen ("colo from Comme)

popergrid	Witck 6 baberaug
substitute - uning (soul, estal	First receiving suntinester. New Section may
Cope Same = Gray whater (completely man) Cope Same = Gray whater (completely man) Cope Same = Gray whater (cope whater) Cope Same = Gray whater (cope whater) From ("Combre coperated Same mane") From ("Combre coperated Same mane")	Code Code
Extra the fire appears (After a) Folia the served correspond (4, material) 4 figures 4 material	det gerfredister (strong) = report = [Co-2-1] + ([A-Zo-2, J+ 1) - report = Ye - Andrew (strong) =
United Capacition 15 (5-be a mather (5-be)) Cash of Capacitins (5-be) multiple (5-be)) Ext. Cash Kasas (3-be) Gran Francis (16-be) 4 / John	SMELTIN CONSIDERS = [3"((hor(c))) bor (in Yoray (sid('a')), urd (2')+1)] midde to = re-hindow (13) Steen min!) For more to medder [: -4] Steenment = Steenment repres (more) for present 10 systematican (steenment) attribute = giptimbates (more) 11 '-join (dinikura), alemans.
	Student = statemen upto (mont []) Sket Constants for (2) Your Students Invert To

papergrid to papergrid	papergrid
Systematic Control of Progress of the Court Regard Systematic Control of Progress of Systematic Control of Systematic	ges stat gette in Cost) School September
Manage.	des yelenment (sur)
At (s)mula(e): **The the (a) = x and x (think!) and x (think!)	Sell : passell
	des gerfamme (2000):
dif grants (100) (1) (1)	m (all-greens)
Trans residet	dep substitute (2004 constants) 4 = (constants, largel) f = 4 " (cost professor) (5 %, year ([constants)
seer = ((-+1)/((4)))	pap (a) if alterate (a) an p for the less pure
now re lines (cop., son)	clan Segunters
dan Feet:	If not spinstages (expression, the) ?
It has appeared (spears 3%)	VIII Vanu. pres ("Experient reset he Story Little Experient Temperient
Sub-represe serguera	1 = express spect (w') see the = [see () - for fin Alid spect " \$
Problem : Parson = Soft Trad Expression (Parsonne)	Sel Ha = Fore (s(q))
Sell-pediati Predian Sell-person - Arrons	des commen (Ell (face))
Kell Health = Cong (Seet, gat contants))	for four face:
And Spile Empress (See, consequent)	for and in section
Prome generalist (common let som Cel).	For I, V A stranger Com
reduce Francis Park (")	grevander (1) 1

poperard	poper
	he India
	S Free Departure 2 a Sept Co. Sept.
	200 11 11 11 12 1 17 1
En state applies two particles (day No.	(49)
torono fiel a la possibilitate	
tipasir f(s), (v) passenate (suc.	An dwag (2m)
do by it contact."	- 17 m 4 m 1
di Carireno [Engl.	Sall in manua (30 (1) ton
arbitra Sanaturi separ (be.	Ze Ca Mar. II
Continue (bull)	See (21(41-7-(1))
Con # 1 Taxon Maraner	
year, Face Car ist for form the sal	Ex+28()
an (Paylenne) for & man has) and	V_ Jun (Ling (a) (group (a) = 101 (9)
Pont /	B. Jo. ("fine (mod))
dur kg t	15 Sep (quidy (2000)
glar_seet_Comb	KLOW ("Ting (Edical)")
1617 (Gar # 1619)	Elitary ("moles")
See my chartest	th-ody og ()
dot . fell (100 , e.):	Avil (70hr) Thus
1 (=) (0.1)	Art (30ho) Thus
the improduce and (improducted)	
Age 1	
Sep Face and (Fee (4))	
the plan step regressions	
gar house (white)	
st 90.	
Dis fort and mi	
del may (our el)	
your sell treesons be I is not upon	
THE DESCRIPTION NO. 1 TO SERVICE	
Part (1 Employ LOV)	
Lat. 12 partil col.	