PRINCIPLES OF ARTIFICIAL INTELLIGENCE
LABORATORY PROGRAMS

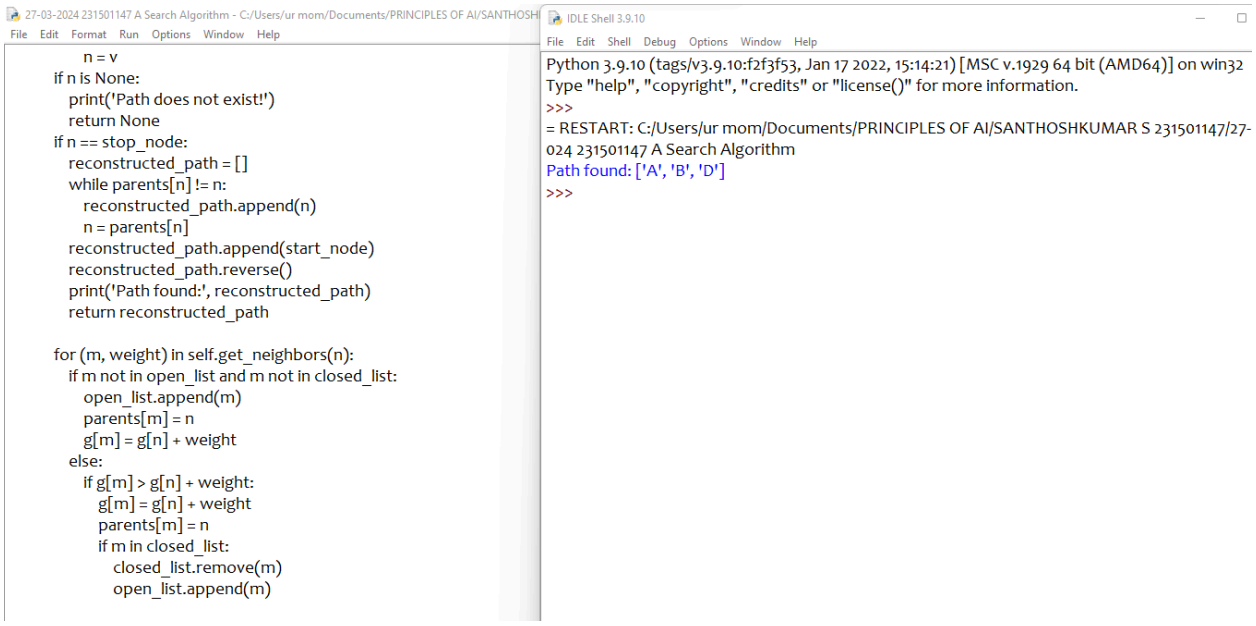
A* SEARCH ALGORITHM PYTHON PROGRAM

SOURCE CODE:

```
from collections import deque

class Graph:
    def __init__(self, adjacency_list):
        self.adjacency_list = adjacency_list

    def get_neighbors(self, v):
        return self.adjacency_list[v]

    def h(self, n):
        # Replace this with your specific heuristic function
        H = {
            'A': 1,
            'B': 1,
            'C': 1,
            'D': 1
        }
        return H[n]

    def a_star_algorithm(self, start_node, stop_node):
        open_list = deque([start_node])  # Use deque for efficient insertions/removals
        closed_list = set()
        g = {start_node: 0}  # Cost from start to each node
        parents = {start_node: start_node}

        while open_list:
            n = None
            for v in open_list:
                if n is None or g[v] + self.h(v) < g[n] + self.h(n):
                    n = v
            if n is None:
```

```
                print('Path does not exist!')
                return None
            if n == stop_node:
                reconstructed_path = []
                while parents[n] != n:
                    reconstructed_path.append(n)
                    n = parents[n]
                reconstructed_path.append(start_node)
                reconstructed_path.reverse()
                print('Path found:', reconstructed_path)
                return reconstructed_path

            for (m, weight) in self.get_neighbors(n):
                if m not in open_list and m not in closed_list:
                    open_list.append(m)
                    parents[m] = n
                    g[m] = g[n] + weight
                else:
                    if g[m] > g[n] + weight:
                        g[m] = g[n] + weight
                        parents[m] = n
                        if m in closed_list:
                            closed_list.remove(m)
                            open_list.append(m)
            open_list.remove(n)
            closed_list.add(n)

        print('Path does not exist!')
        return None
# Example usage (assuming the adjacency list is defined as before)
adjacency_list = {
    'A': [('B', 1), ('C', 3), ('D', 7)],
    'B': [('D', 5)],
    'C': [('D', 12)]
}
graph1 = Graph(adjacency_list)
graph1.a_star_algorithm('A', 'D')
```

OUTPUT:



```
27-03-2024 231501147 A Search Algorithm - C:/Users/ur mom/Documents/PRINCIPLES OF AI/SANTHOSH
File  Edit  Format  Run  Options  Window  Help
            n = v
        if n is None:
            print('Path does not exist!')
            return None
        if n == stop_node:
            reconstructed_path = []
            while parents[n] != n:
                reconstructed_path.append(n)
                n = parents[n]
            reconstructed_path.append(start_node)
            reconstructed_path.reverse()
            print('Path found:', reconstructed_path)
            return reconstructed_path

        for (m, weight) in self.get_neighbors(n):
            if m not in open_list and m not in closed_list:
                open_list.append(m)
                parents[m] = n
                g[m] = g[n] + weight
            else:
                if g[m] > g[n] + weight:
                    g[m] = g[n] + weight
                    parents[m] = n
                    if m in closed_list:
                        closed_list.remove(m)
                        open_list.append(m)
```

```
IDLE Shell 3.9.10
File  Edit  Shell  Debug  Options  Window  Help
Python 3.9.10 (tags/v3.9.10:f2f3f53, Jan 17 2022, 15:14:21) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/ur mom/Documents/PRINCIPLES OF AI/SANTHOSHKUMAR S 231501147/27-024 231501147 A Search Algorithm
Path found: ['A', 'B', 'D']
>>>
```

PRINCIPLES OF ARTIFICIAL INTELLIGENCE
LABORATORY PROGRAMS

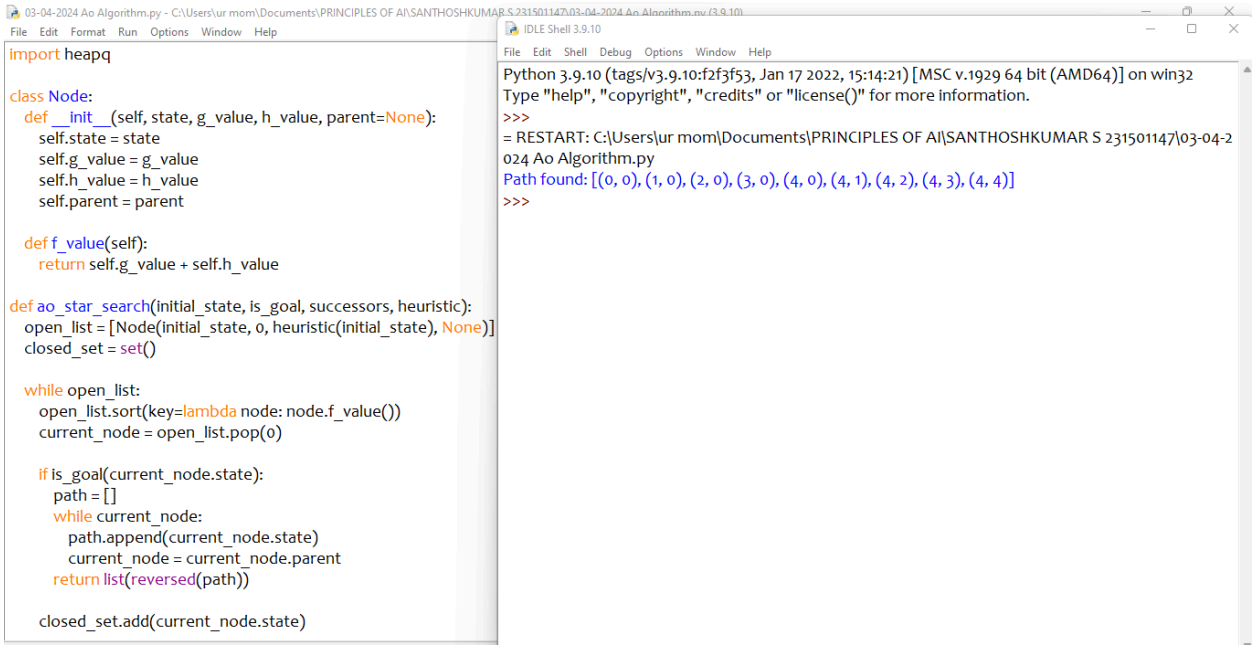
Ao* SEARCH ALGORITHM PYTHON PROGRAM

SOURCE CODE:

```
import heapq

class Node:
    def __init__(self, state, g_value, h_value, parent=None):
        self.state = state
        self.g_value = g_value
        self.h_value = h_value
        self.parent = parent

    def f_value(self):
        return self.g_value + self.h_value

def ao_star_search(initial_state, is_goal, successors, heuristic):
    open_list = [Node(initial_state, 0, heuristic(initial_state), None)]
    closed_set = set()

    while open_list:
        open_list.sort(key=lambda node: node.f_value())
        current_node = open_list.pop(0)

        if is_goal(current_node.state):
            path = []
            while current_node:
                path.append(current_node.state)
                current_node = current_node.parent
            return list(reversed(path))

        closed_set.add(current_node.state)

        for child_state in successors(current_node.state):
            if child_state in closed_set:
```

```
            continue
        g_value = current_node.g_value + 1
        h_value = heuristic(child_state)
        child_node = Node(child_state, g_value, h_value, current_node)
        in_open_list = any(node.state == child_state for node in open_list)
        if not in_open_list:
            open_list.append(child_node)
        else:
            existing_node = next(node for node in open_list if node.state == child_state)
            if existing_node.g_value > g_value:
                existing_node.g_value = g_value
                existing_node.parent = current_node

    return None

def is_goal(state):
    return state == (4, 4)

def successors(state):
    x, y = state
    return [(x + 1, y), (x, y + 1)]

def heuristic(state):
    x, y = state
    return abs(4 - x) + abs(4 - y)

initial_state = (0, 0)
path = ao_star_search(initial_state, is_goal, successors, heuristic)
if path:
    print("Path found:", path)
else:
    print("No path found")
```

OUTPUT:



```
import heapq

class Node:
    def __init__(self, state, g_value, h_value, parent=None):
        self.state = state
        self.g_value = g_value
        self.h_value = h_value
        self.parent = parent

    def f_value(self):
        return self.g_value + self.h_value

def ao_star_search(initial_state, is_goal, successors, heuristic):
    open_list = [Node(initial_state, 0, heuristic(initial_state), None)]
    closed_set = set()

    while open_list:
        open_list.sort(key=lambda node: node.f_value())
        current_node = open_list.pop(0)

        if is_goal(current_node.state):
            path = []
            while current_node:
                path.append(current_node.state)
                current_node = current_node.parent
            return list(reversed(path))

        closed_set.add(current_node.state)
```

```
Python 3.9.10 (tags/v3.9.10:f2f3f53, Jan 17 2022, 15:14:21) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\ur mom\Documents\PRINCIPLES OF AI\SANTHOSHKUMAR S 231501147\03-04-2024 Ao Algorithm.py
Path found: [(0, 0), (1, 0), (2, 0), (3, 0), (4, 0), (4, 1), (4, 2), (4, 3), (4, 4)]
>>>
```

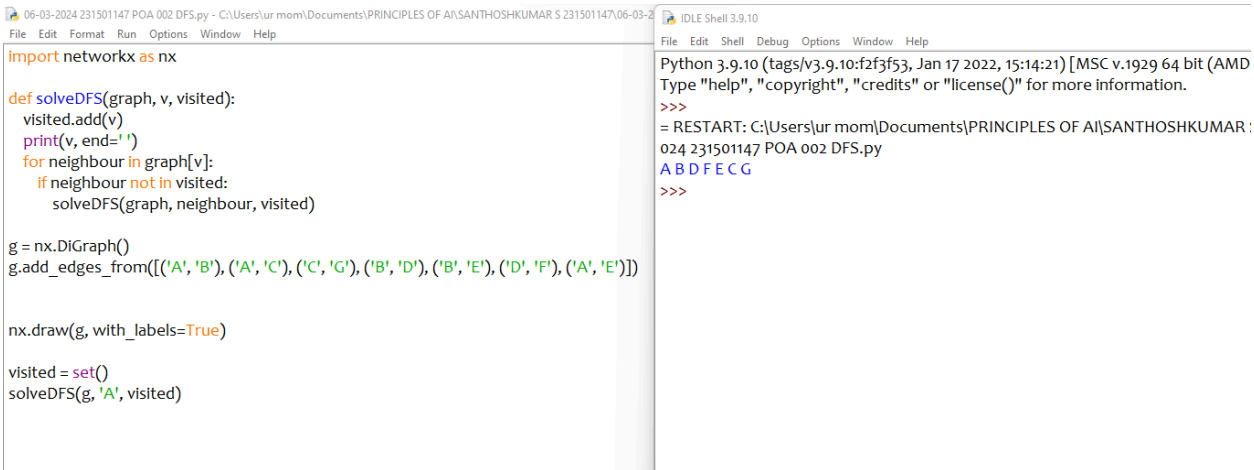
PRINCIPLES OF ARTIFICIAL INTELLIGENCE
LABORATORY PROGRAMS

DEPTH FIRST SEARCH:

SOURCE CODE:

```
import networkx as nx

def solveDFS(graph, v, visited):
    visited.add(v)
    print(v, end=' ')
    for neighbour in graph[v]:
        if neighbour not in visited:
            solveDFS(graph, neighbour, visited)

g = nx.DiGraph()
g.add_edges_from([('A', 'B'), ('A', 'C'), ('C', 'G'), ('B', 'D'), ('B', 'E'), ('D', 'F'), ('A', 'E')])


nx.draw(g, with_labels=True)

visited = set()
solveDFS(g, 'A', visited)
```

OUTPUT:



PRINCIPLES OF ARTIFICIAL INTELLIGENCE
LABORATORY PROGRAMS

INTRODUCTION TO PROLOG PROGRAM:

SOURCE CODE:

KB1:

```
woman(mia).
woman(jody).
woman(yolanda).
playsAirGuitar(jody).
party.
```

KB2:

```
happy(yolanda).
listens2music(mia).
Listens2music(yolanda):-happy(yolanda).
playsAirGuitar(mia):-listens2music(mia).
playsAirGuitar(Yolanda):-listens2music(yolanda).
```

KB3:

```
likes(dan,sally).
likes(sally,dan).
likes(john,brittney).
married(X,Y) :- likes(X,Y) , likes(Y,X).
friends(X,Y) :- likes(X,Y) ; likes(Y,X).
```

KB4:

```
food(burger).
food(sandwich).
food(pizza).
lunch(sandwich).
dinner(pizza).
meal(X):-food(X).
```

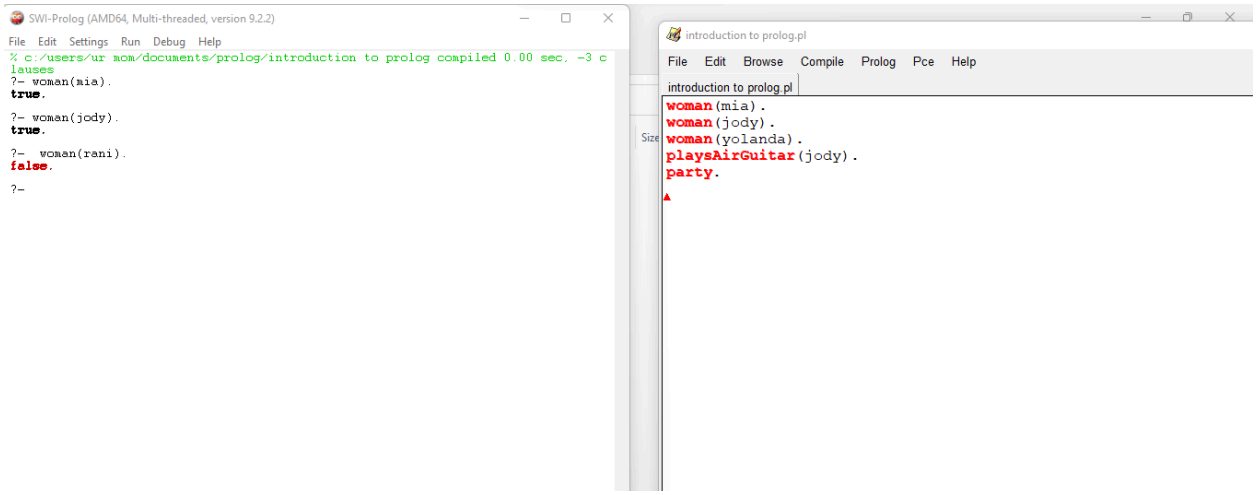
KB5:

owns(jack,car(bmw)).
owns(john,car(chevy)).
owns(olivia,car(civic)).
owns(jane,car(chevy)).
sedan(car(bmw)).
sedan(car(civic)).
truck(car(chevy)).

KB6: Find minimum maximum of two numbers

find_max(X,Y,X):-X>=Y,!.
find_max(X,Y,Y):-X<Y.
find_min(X,Y,X):-X=<Y,!.
find_min(X,Y,Y):-X>Y.

OUTPUT:



PRINCIPLES OF ARTIFICIAL INTELLIGENCE
LABORATORY PROGRAMS

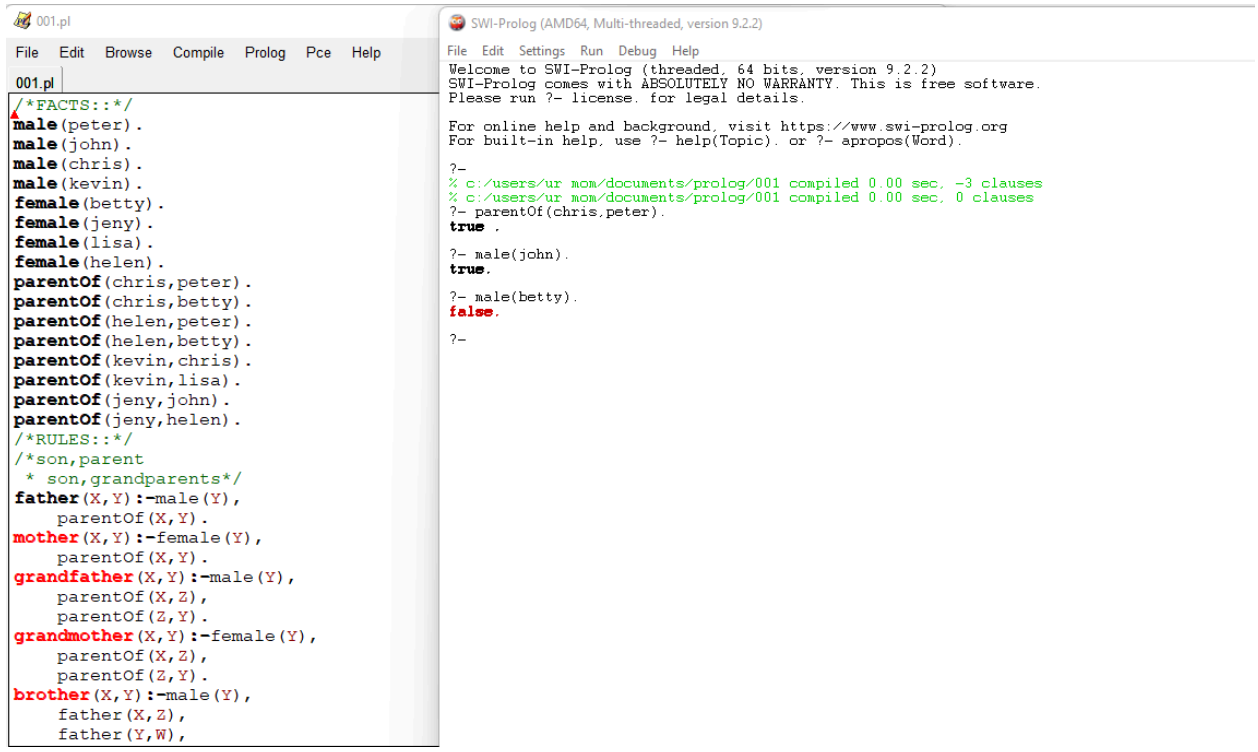
PROLOG PROGRAM:

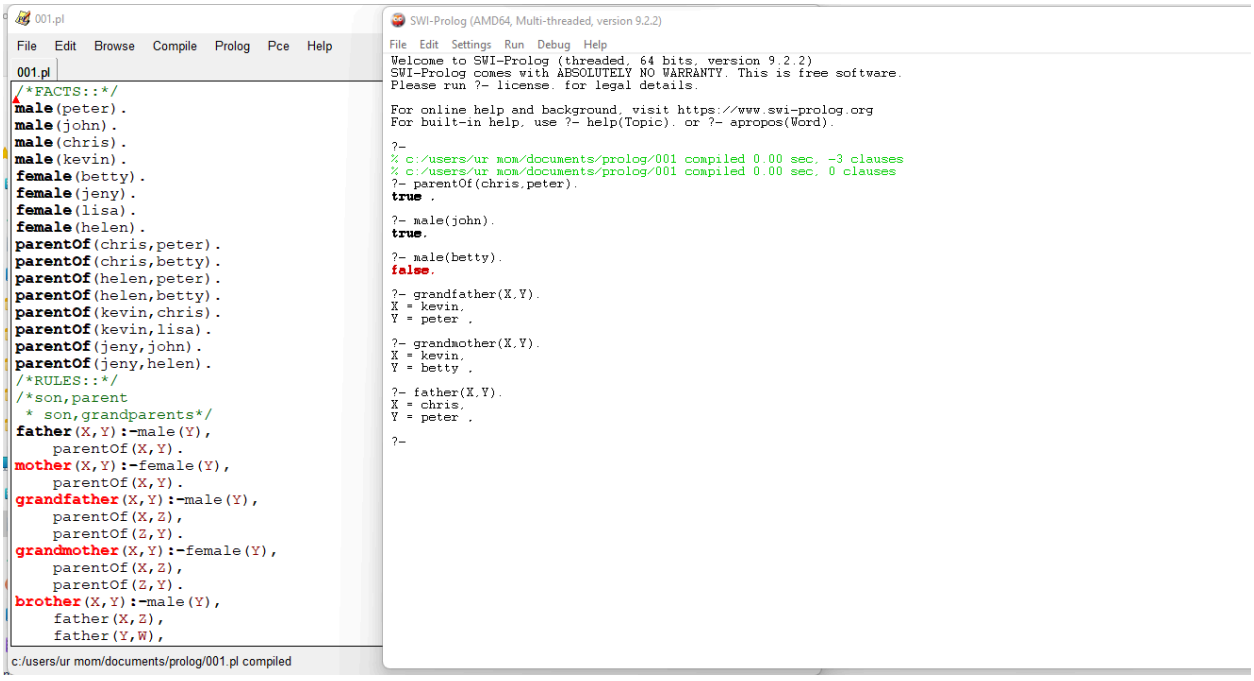
SOURCE CODE:

```
/*FACTS::*/
male(peter).
male(john).
male(chris).
male(kevin).
female(betty).
female(jeny).
female(lisa).
female(helen).
parentOf(chris,peter).
parentOf(chris,betty).
parentOf(helen,peter).
parentOf(helen,betty).
parentOf(kevin,chris).
parentOf(kevin,lisa).
parentOf(jeny,john).
parentOf(jeny,helen).
/*RULES::*/
/*son,parent
 * son,grandparents*/
father(X,Y):-male(Y),
   parentOf(X,Y).
mother(X,Y):-female(Y),
   parentOf(X,Y).
grandfather(X,Y):-male(Y),
   parentOf(X,Z),
   parentOf(Z,Y).
grandmother(X,Y):-female(Y),
   parentOf(X,Z),
   parentOf(Z,Y).
brother(X,Y):-male(Y),
```

```
    father(X,Z),
    father(Y,W),
    Z==W.
sister(X,Y):-female(Y),
    father(X,Z),
    father(Y,W),
    Z==W.
```

OUTPUT:





PRINCIPLES OF ARTIFICIAL INTELLIGENCE
LABORATORY PROGRAMS

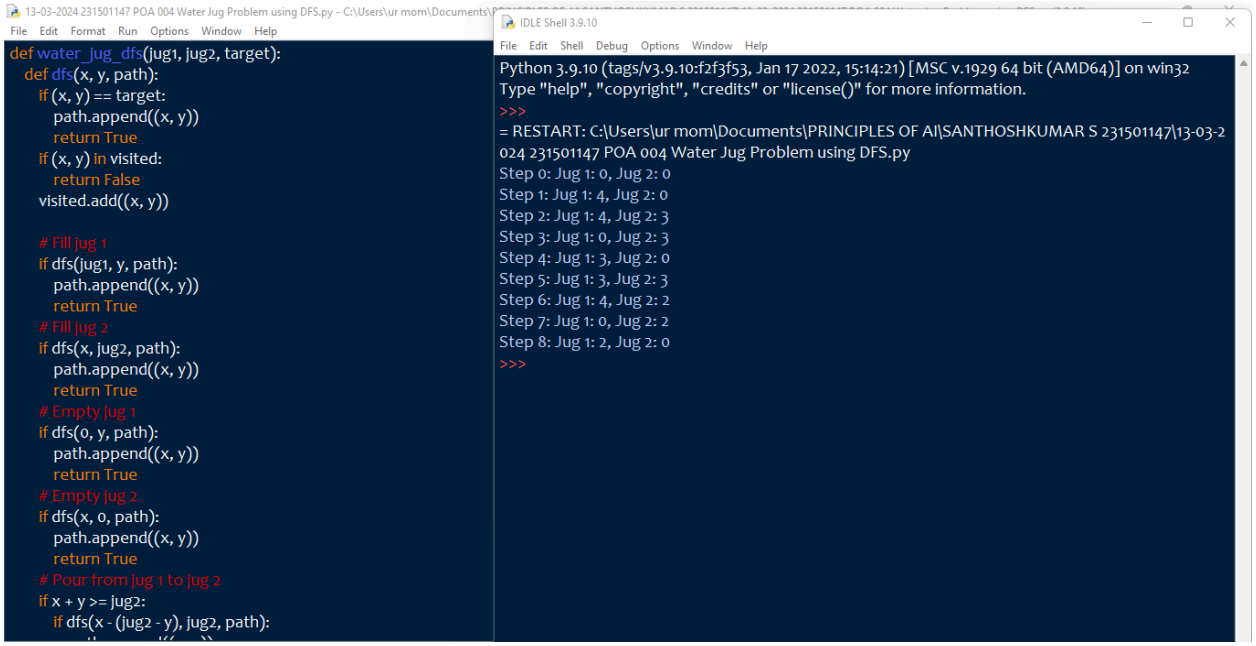
WATER JUG PROBLEM USING DEPTH FIRST SEARCH:

SOURCE CODE:

```
def water_jug_dfs(jug1, jug2, target):
    def dfs(x, y, path):
        if (x, y) == target:
            path.append((x, y))
            return True+-
        if (x, y) in visited:
            return False
        visited.add((x, y))

        # Fill jug 1
        if dfs(jug1, y, path):
            path.append((x, y))
            return True
        # Fill jug 2
        if dfs(x, jug2, path):
            path.append((x, y))
            return True
        # Empty jug 1
        if dfs(0, y, path):
            path.append((x, y))
            return True
        # Empty jug 2
        if dfs(x, 0, path):
            path.append((x, y))
            return True
        # Pour from jug 1 to jug 2
        if x + y >= jug2:
            if dfs(x - (jug2 - y), jug2, path):
                path.append((x, y))
                return True
        else:
```

```
            if dfs(0, x + y, path):
                path.append((x, y))
                return True
        # Pour from jug 2 to jug 1
        if x + y >= jug1:
            if dfs(jug1, y - (jug1 - x), path):
                path.append((x, y))
                return True
        else:
            if dfs(x + y, 0, path):
                path.append((x, y))
                return True
        return False

    visited = set()
    path = []
    if dfs(0, 0, path):
        path.reverse()
        return path
    else:
        return "No solution found."

jug1_capacity = 4
jug2_capacity = 3
target_amount = (2, 0)
solution_path = water_jug_dfs(jug1_capacity, jug2_capacity, target_amount)
if solution_path != "No solution found.":
    for step, (x, y) in enumerate(solution_path):
        print(f"Step {step}: Jug 1: {x}, Jug 2: {y}")
else:
    print("No solution found.")
```

OUTPUT:


13-03-2024 231501147 POA 004 Water Jug Problem using DFS.py - C:\Users\ur mom\Documents\
File Edit Format Run Options Window Help
def water_jug_dfs(jug1, jug2, target):
def dfs(x, y, path):
if (x, y) == target:
path.append((x, y))
return True
if (x, y) in visited:
return False
visited.add((x, y))
# Fill jug 1
if dfs(jug1, y, path):
path.append((x, y))
return True
# Fill jug 2
if dfs(x, jug2, path):
path.append((x, y))
return True
# Empty jug 1
if dfs(0, y, path):
path.append((x, y))
return True
# Empty jug 2
if dfs(x, 0, path):
path.append((x, y))
return True
# Pour from jug 1 to jug 2
if x + y >= jug2:
if dfs(x - (jug2 - y), jug2, path):
IDLE Shell 3.9.10
File Edit Shell Debug Options Window Help
Python 3.9.10 (tags/v3.9.10:f2f3f53, Jan 17 2022, 15:14:21) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\ur mom\Documents\PRINCIPLES OF AI\SANTHOSHKUMAR S 231501147\13-03-2024 231501147 POA 004 Water Jug Problem using DFS.py
Step 0: Jug 1: 0, Jug 2: 0
Step 1: Jug 1: 4, Jug 2: 0
Step 2: Jug 1: 4, Jug 2: 3
Step 3: Jug 1: 0, Jug 2: 3
Step 4: Jug 1: 3, Jug 2: 0
Step 5: Jug 1: 3, Jug 2: 3
Step 6: Jug 1: 4, Jug 2: 2
Step 7: Jug 1: 0, Jug 2: 2
Step 8: Jug 1: 2, Jug 2: 0
>>>

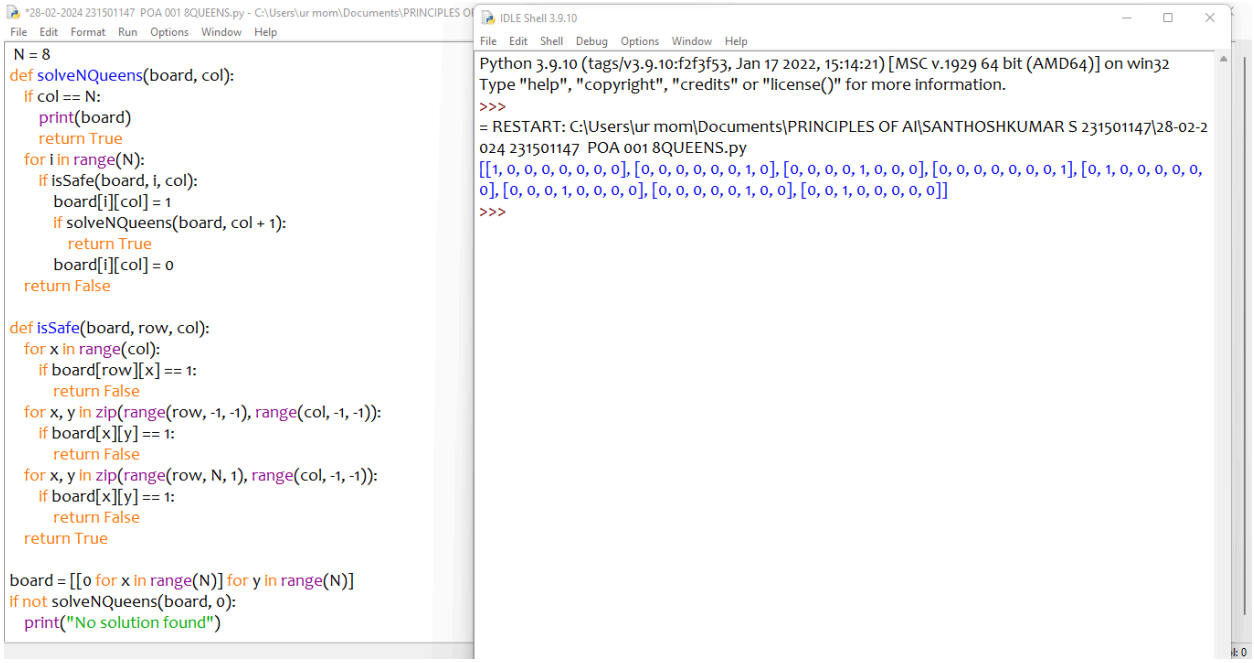
PRINCIPLES OF ARTIFICIAL INTELLIGENCE
LABORATORY PROGRAMS

8 QUEENS PROBLEM:

SOURCE CODE:

```
N = 8
def solveNQueens(board, col):
    if col == N:
        print(board)
        return True
    for i in range(N):
        if isSafe(board, i, col):
            board[i][col] = 1
            if solveNQueens(board, col + 1):
                return True
            board[i][col] = 0
    return False

def isSafe(board, row, col):
    for x in range(col):
        if board[row][x] == 1:
            return False
    for x, y in zip(range(row, -1, -1), range(col, -1, -1)):
        if board[x][y] == 1:
            return False
    for x, y in zip(range(row, N, 1), range(col, -1, -1)):
        if board[x][y] == 1:
            return False
    return True

board = [[0 for x in range(N)] for y in range(N)]
if not solveNQueens(board, 0):
    print("No solution found")
```

OUTPUT:


*28-02-2024 231501147 POA 001 8QUEENS.py - C:\Users\ur mom\Documents\PRINCIPLES OF
File Edit Format Run Options Window Help
N = 8
def solveNQueens(board, col):
if col == N:
print(board)
return True
for i in range(N):
if isSafe(board, i, col):
board[i][col] = 1
if solveNQueens(board, col + 1):
return True
board[i][col] = 0
return False
def isSafe(board, row, col):
for x in range(col):
if board[row][x] == 1:
return False
for x, y in zip(range(row, -1, -1), range(col, -1, -1)):
if board[x][y] == 1:
return False
for x, y in zip(range(row, N, 1), range(col, -1, -1)):
if board[x][y] == 1:
return False
return True
board = [[0 for x in range(N)] for y in range(N)]
if not solveNQueens(board, 0):
print("No solution found")
IDLE Shell 3.9.10
File Edit Shell Debug Options Window Help
Python 3.9.10 (tags/v3.9.10:f2f3f53, Jan 17 2022, 15:14:21) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\ur mom\Documents\PRINCIPLES OF AI\SANTHOSHKUMAR S 231501147\28-02-2024 231501147 POA 001 8QUEENS.py
[[1, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 1, 0], [0, 0, 0, 0, 1, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 1], [0, 1, 0, 0, 0, 0, 0, 0], [0, 0, 0, 1, 0, 0, 0, 0], [0, 0, 0, 0, 0, 1, 0, 0], [0, 0, 1, 0, 0, 0, 0, 0]]
>>>
Ln: 0