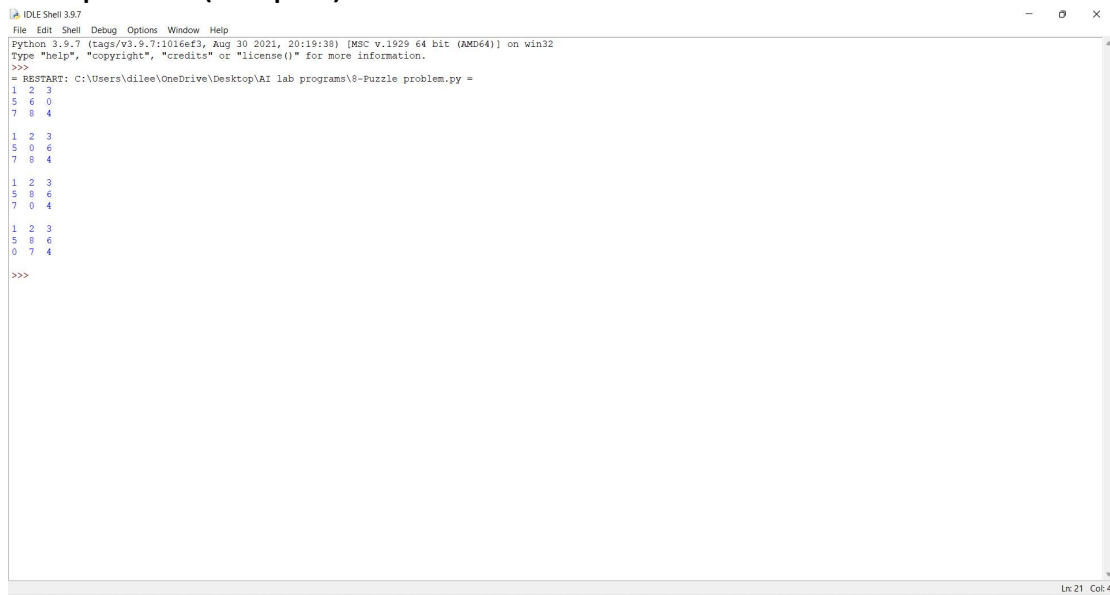


OUTPUTS

1. 8 puzzle (output):



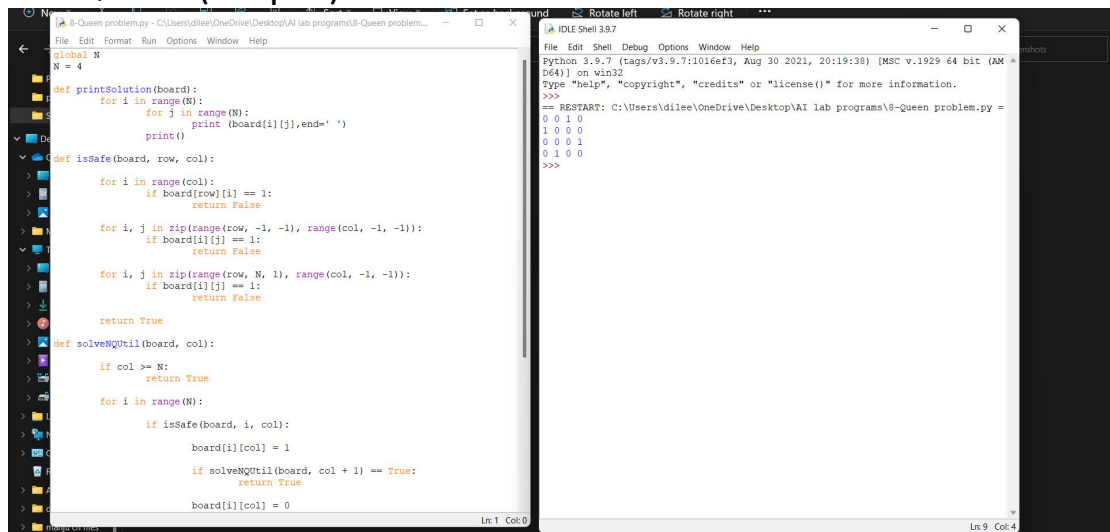
```
Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\dilee\OneDrive\Desktop\AI lab programs\8-Puzzle problem.py =
1 2 3
5 6 0
7 8 4

1 2 3
5 0 6
7 8 4

1 2 3
5 8 6
7 0 4

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

2.8 Queen (output):



```
Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
== RESTART: C:\Users\dilee\OneDrive\Desktop\AI lab programs\8-Queen problem.py =
0 0 1 0
1 0 0 0
0 0 0 1
0 1 0 0
>>>
```

```
global N
N = 4

def printSolution(board):
    for i in range(N):
        for j in range(N):
            print (board[i][j],end=' ')
        print()

def isSafe(board, row, col):
    for i in range(col):
        if board[row][i] == 1:
            return False

    for i, j in zip(range(row, -1, -1), range(col, -1, -1)):
        if board[i][j] == 1:
            return False

    for i, j in zip(range(row, N, 1), range(col, -1, -1)):
        if board[i][j] == 1:
            return False

    return True

def solveNQUtil(board, col):
    if col >= N:
        return True

    for i in range(N):
        if isSafe(board, i, col):
            board[i][col] = 1
            if solveNQUtil(board, col + 1) == True:
                return True
            board[i][col] = 0
```

3. water jug(output):

The screenshot shows an IDE with two windows. The left window displays the Python code for the 'water jug' problem, which uses a recursive function to find the minimum number of steps to reach a target amount. The right window shows the output of the program, which is a list of steps: 0 0, 4 0, 4 3, 0 3, 3 0, 3 3, 4 2, 0 2, and 0 0.

```
File Edit Format Run Options Window Help
Water Jug Problem.py - C:\Users\dilee\OneDrive\Desktop\AI lab programs\Water Jug Problem.py
from collections import defaultdict
jug1, jug2, aim = 4, 3, 2
visited = defaultdict(lambda: False)

def waterJugSolver(amt1, amt2):
    if (amt1 == aim and amt2 == 0) or (amt2 == aim and amt1 == 0):
        print(amt1, amt2)
        return True
    if visited[(amt1, amt2)] == False:
        print(amt1, amt2)
        visited[(amt1, amt2)] = True
        return (waterJugSolver(0, amt2) or
                waterJugSolver(jug1, amt2) or
                waterJugSolver(amt1, jug2) or
                waterJugSolver(amt1 + min(amt2, (jug1-amt1)),
                                amt2 - min(amt2, (jug1-amt1))) or
                waterJugSolver(amt1 - min(amt1, (jug2-amt2)),
                                amt2 + min(amt1, (jug2-amt2))))
    else:
        return False
print("Steps: ")
waterJugSolver(0, 0)
```

```
Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\dilee\OneDrive\Desktop\AI lab programs\Water Jug Problem.py
Steps:
0 0
4 0
4 3
0 3
3 0
3 3
4 2
0 2
0 0
>>>
```

4. crypt-arithmetic(output):

The screenshot shows an IDE with two windows. The left window displays the Python code for the 'crypt-arithmetic' problem, which uses a recursive function to find the minimum number of steps to reach a target amount. The right window shows the output of the program, which is a list of steps: 0 0, 4 0, 4 3, 0 3, 3 0, 3 3, 4 2, 0 2, and 0 0.

```
File Edit Format Run Options Window Help
Crypt-Arithmetic problem.py - C:\Users\dilee\OneDrive\Desktop\AI lab programs\Crypt-Arith...
global N = 4
def isSolvable(words, result):
    mp = [-1]*(26)
    used = [0]*(10)
    Hash = [0]*(26)
    CharAtFront = [0]*(26)

    uniq = ""
    for word in range(len(words)):
        for i in range(len(words[word])):
            ch = words[word][i]
            Hash[ord(ch) - ord('A')] += pow(10, len(words[word]) - i)
            if mp[ord(ch) - ord('A')] == -1:
                mp[ord(ch) - ord('A')] = 0
                uniq += str(ch)

            if i == 0 and len(words[word]) > 1:
                CharAtFront[ord(ch) - ord('A')] = 1

    for i in range(len(result)):
        ch = result[i]
        Hash[ord(ch) - ord('A')] -= pow(10, len(result) - i - 1)

        if mp[ord(ch) - ord('A')] == -1:
            mp[ord(ch) - ord('A')] = 0
            uniq += str(ch)

        if i == 0 and len(result) > 1:
            CharAtFront[ord(ch) - ord('A')] = 1
```

```
Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\dilee\OneDrive\Desktop\AI lab programs\Crypt-Arithmetic prob...
lem.py
Yes
>>> |
```

5. Missionaries(output):

```

Missionaries Cannibal problem.py - C:\Users\dilee\OneDrive\Desktop\AI lab programs\Missionaries Cannibal problem.py
File Edit Format Run Options Window Help
print("\n")
print("\tGame Start\nNow the task is to move all of them to right side of the river")
print("rules:\n1. The boat can carry at most two people\n2. If cannibals num greater than missionaries, they will eat them")
LM = 3
LC = 3
RM = 0
RC = 0
userM = 0
userC = 0
k = 0
print("\nM M C C C | --- | \n")
try:
    while(True):
        while(Trus):
            print("Left side -> right side river travel")
            uM = int(input("Enter number of Missionaries travel => "))
            uC = int(input("Enter number of Cannibals travel => "))
            if((uM==0)and(uC==0)):
                print("Empty travel not possible")
                print("Re-enter : ")
            elif(((uM+uC) <= 2)and((LM-uM)>=0)and((LC-uC)>=0)):
                break
            else:
                print("Wrong input re-enter : ")
            LM = (LM-uM)
            LC = (LC-uC)
            RM += uM
            RC += uC
        print("\n")
        for i in range(0,LM):
            print("M ",end="")
        for i in range(0,LC):
            print("C ",end="")
        print("| --> | ",end="")
        for i in range(0,RM):
            print("M ",end="")
        for i in range(0,RC):
            print("C ",end="")
        print("\n")
        k += 1
        if k == 10:
            break
    print("Game End")
except:
    print("Invalid input")

```

6. Vaccum(output):

```

Missionaries Cannibal problem.py - C:\Users\dilee\OneDrive\Desktop\AI lab programs\Missionaries Cannibal problem.py
File Edit Format Run Options Window Help
import random
def display(room):
    print(room)
room = [
    [1, 1, 1, 1],
    [1, 1, 1, 1],
    [1, 1, 1, 1],
    [1, 1, 1, 1]
]
print("All the room are dirty")
display(room)
x = 0
y = 0
while x < 4:
    while y < 4:
        room[x][y] = random.choice([0,1])
        x+=1
        y+=1
    y=0
    x=0
print("Before cleaning the room I detect all of these random dirt")
display(room)
x = 0
y = 0
z = 0
while x < 4:
    while y < 4:
        if room[x][y] == 1:
            print("Vacuum in this location now, ", x, y)
            room[x][y] = 0
            print("cleaned", x, y)
            z+=1
            y+=1
        else:
            y+=1
        if y == 4:
            y=0
            x+=1
        if x == 4:
            x=0
    print("Room is clean now, Thanks for using : 3710933")
    print("performance = (z/16)*100")
    z = 0

```

7. Bfs(output):

The screenshot shows a Python IDE with two windows. The left window displays a file named `bfs.py` with the following code:

```

import sys
from collections import defaultdict

class Graph:
    def __init__(self):
        self.graph = defaultdict(list)

    def addEdge(self, u, v):
        self.graph[u].append(v)

    def BFS(self, s):
        visited = [False] * (len(self.graph))
        queue = []
        queue.append(s)
        visited[s] = True

        while queue:
            s = queue.pop(0)
            print(s, end=" ")

            for i in self.graph[s]:
                if visited[i] == False:
                    queue.append(i)
                    visited[i] = True

g = Graph()
g.addEdge(0, 1)
g.addEdge(0, 2)
g.addEdge(1, 2)
g.addEdge(2, 0)
g.addEdge(2, 3)
g.addEdge(3, 3)

```

The right window shows the IDLE Shell output:

```

Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\dilee\OneDrive\Desktop\AI lab programs\bfs.py =====
Following is Breadth First Traversal (starting from vertex 2)
2 0 3 1
>>>

```

8. Dfs(output):

The screenshot shows a Python IDE with two windows. The left window displays a file named `dfs.py` with the following code:

```

import sys
from collections import defaultdict

class Graph:
    def __init__(self):
        self.graph = defaultdict(list)

    def addEdge(self, u, v):
        self.graph[u].append(v)

    def DFSUtil(self, v, visited):
        visited.add(v)
        print(v, end=" ")

        for neighbour in self.graph[v]:
            if neighbour not in visited:
                self.DFSUtil(neighbour, visited)

    def DFS(self, v):
        visited = set()
        self.DFSUtil(v, visited)

if __name__ == "__main__":
    g = Graph()
    g.addEdge(0, 1)
    g.addEdge(0, 2)
    g.addEdge(1, 2)
    g.addEdge(2, 0)
    g.addEdge(2, 3)
    g.addEdge(3, 3)

    print("Following is DFS from (starting from vertex 2)")
    g.DFS(2)

```

The right window shows the IDLE Shell output:

```

Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\dilee\OneDrive\Desktop\AI lab programs\dfs.py =====
Following is DFS from (starting from vertex 2)
2 0 1 3
>>>

```

9. travelson(output):

The image shows a Python IDE with two windows. The left window displays a Python script for a Travelling Salesman Problem (TSP) solver. The script defines a function `travellingSalesmanProblem` that takes a graph and a starting node `s` as input. It uses `maxsize` to initialize the minimum path weight and `permutations` to generate all possible paths starting from `s`. The function iterates through these permutations, calculating the total weight of each path and updating the minimum path weight. The right window shows the execution of the script, which prints the minimum path weight.

```
travelsion.py - C:\Users\dilee\OneDrive\Desktop\AI lab programs\travelsion.py (3.9.7)
File Edit Format Run Options Window Help
from sys import maxsize
from itertools import permutations
V = 4
def travellingSalesmanProblem(graph, s):
    vertex = []
    for i in range(V):
        if i != s:
            vertex.append(i)
    min_path = maxsize
    next_permutation=permutations(vertex)
    for i in next_permutation:
        current_pathweight = 0
        k = s
        for j in i:
            current_pathweight += graph[k][j]
            k = j
        current_pathweight += graph[k][s]
        min_path = min(min_path, current_pathweight)
    return min_path
if __name__ == "__main__":
    graph = [[0, 10, 15, 20], [10, 0, 35, 25],
             [15, 35, 0, 30], [20, 25, 30, 0]]
    s = 0
    print(travellingSalesmanProblem(graph, s))
Ln 1 Col 0
```

```
IDLE Shell 3.9.7
Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\dilee\OneDrive\Desktop\AI lab programs\travelsion.py =====
80
>>>
Ln 6 Col 4
```

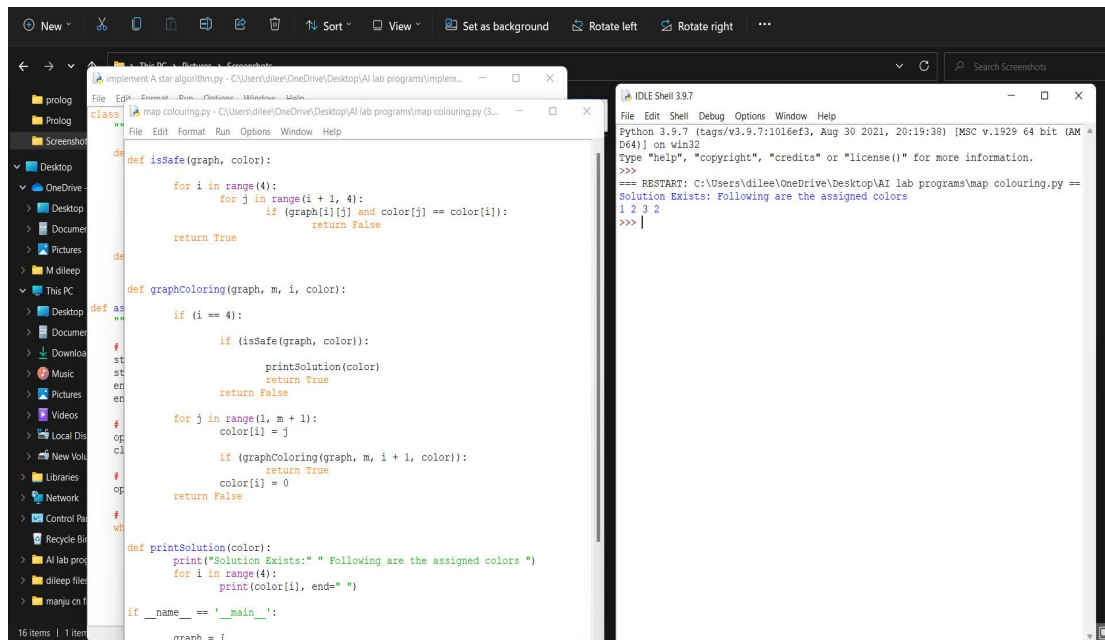
10. A* (output):

The image shows a Python IDE with two windows. The left window displays a Python script for an A* search algorithm. The script defines a function `implement A star algorithm` that takes a start node `g` and an end node `h` as input. It uses a priority queue to explore nodes, calculating the `f` cost (sum of `g` cost and `h` cost). The right window shows the execution of the script, which prints the path from the start node to the end node.

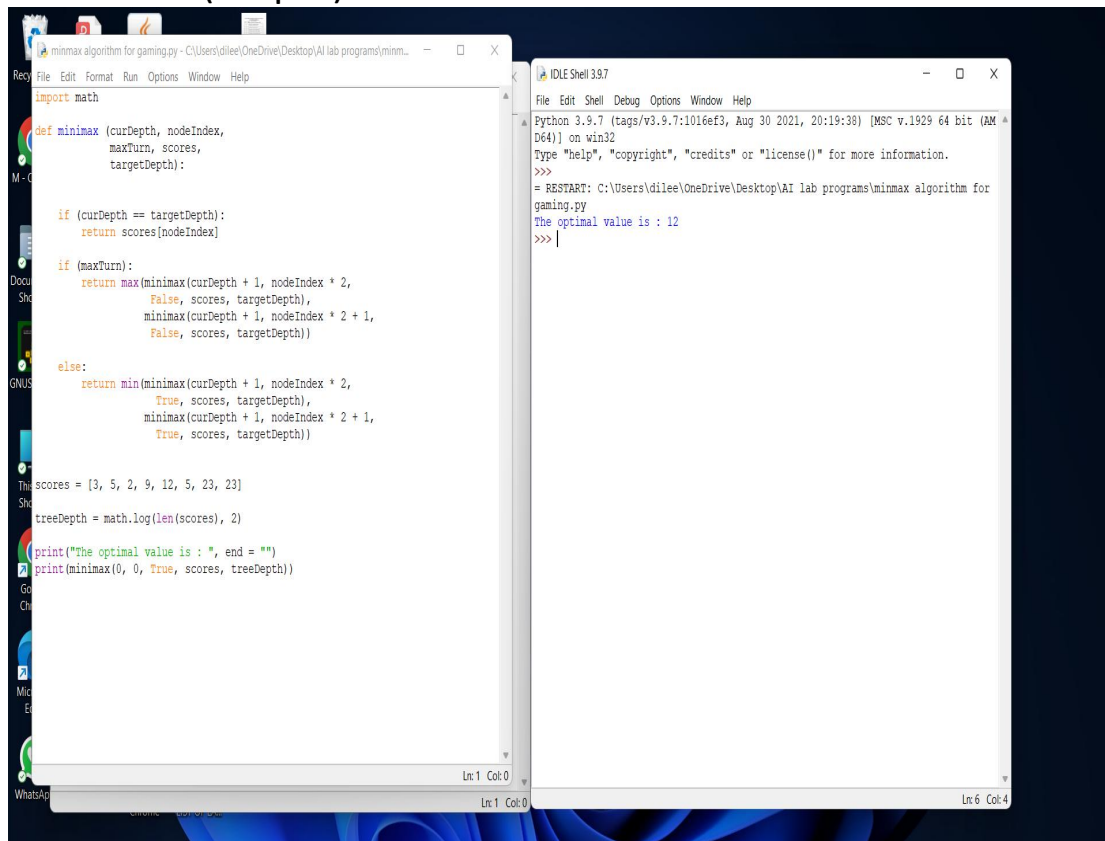
```
SWH-Protog (AMD64 Multi-threaded version 8.4.3)
File Edit Settings Run Debug Help
Welcome to SWH-Protog (threaded, 64 bits, version 8.4.3)
implement A star algorithm.py - C:\Users\dilee\OneDrive\Desktop\AI lab programs\implem...
File Edit Format Run Options Window Help
start_node.g = start_node.h = start_node.f = 0
end_node = Node(None, end)
end_node.g = end_node.h = end_node.f = 0
# Initialize both open and closed list
open_list = []
closed_list = []
# Add the start node
open_list.append(start_node)
# Loop until you find the end
while len(open_list) > 0:
    # Get the current node
    current_node = open_list[0]
    current_index = 0
    for index, item in enumerate(open_list):
        if item.f < current_node.f:
            current_node = item
            current_index = index
    # Pop current off open list, add to closed list
    open_list.pop(current_index)
    closed_list.append(current_node)
    # Found the goal
    if current_node == end_node:
        path = []
        current = current_node
        while current is not None:
            path.append(current.position)
            current = current.parent
        return path[::-1] # Return reversed path
    # Generate children
    children = []
    for new_position in [(0, -1), (0, 1), (-1, 0), (1, 0), (-1, -1), (-1, 1)]:
        # Get node position
Ln 33 Col 0
```

```
IDLE Shell 3.9.7
Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\dilee\OneDrive\Desktop\AI lab programs\implement A star algo...
rithm.py
(10, 0), (1, 1), (2, 2), (3, 3), (4, 3), (5, 4), (6, 5), (7, 6)]
>>>
Ln 6 Col 4
```

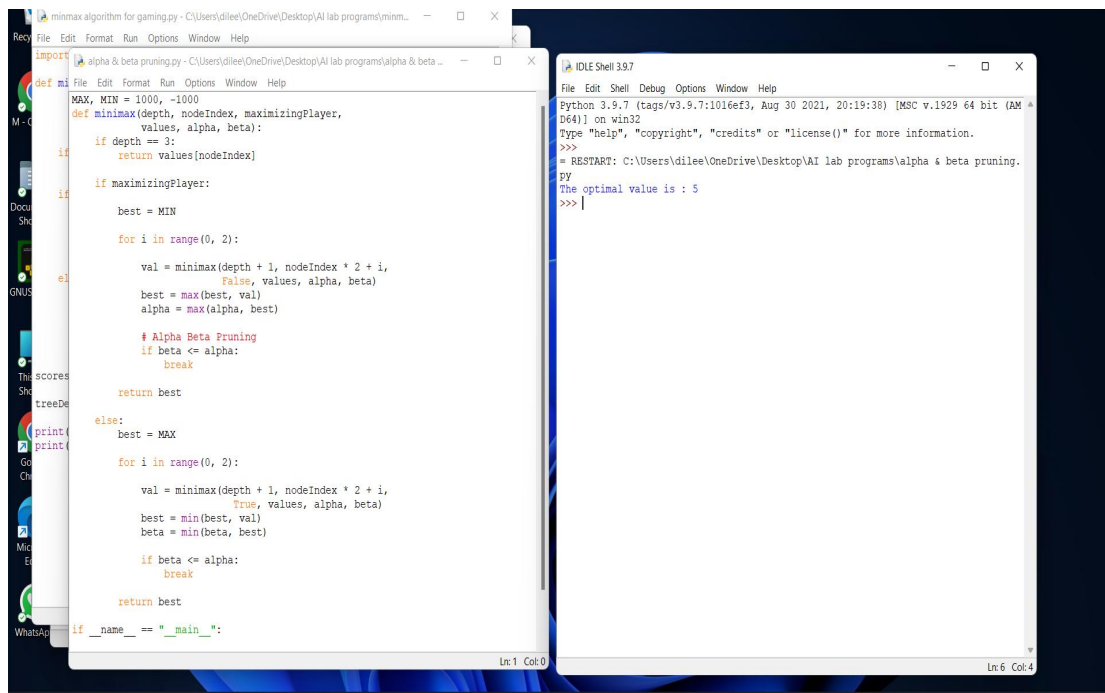
11. map color(output):



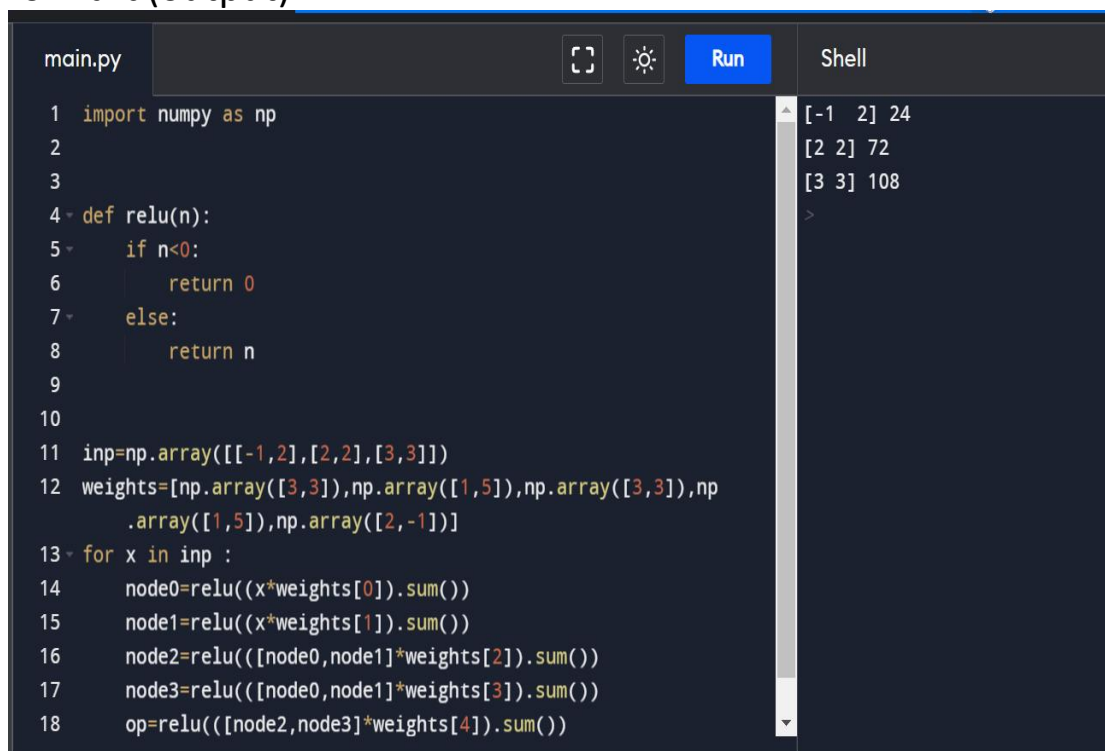
13.minimax(output):



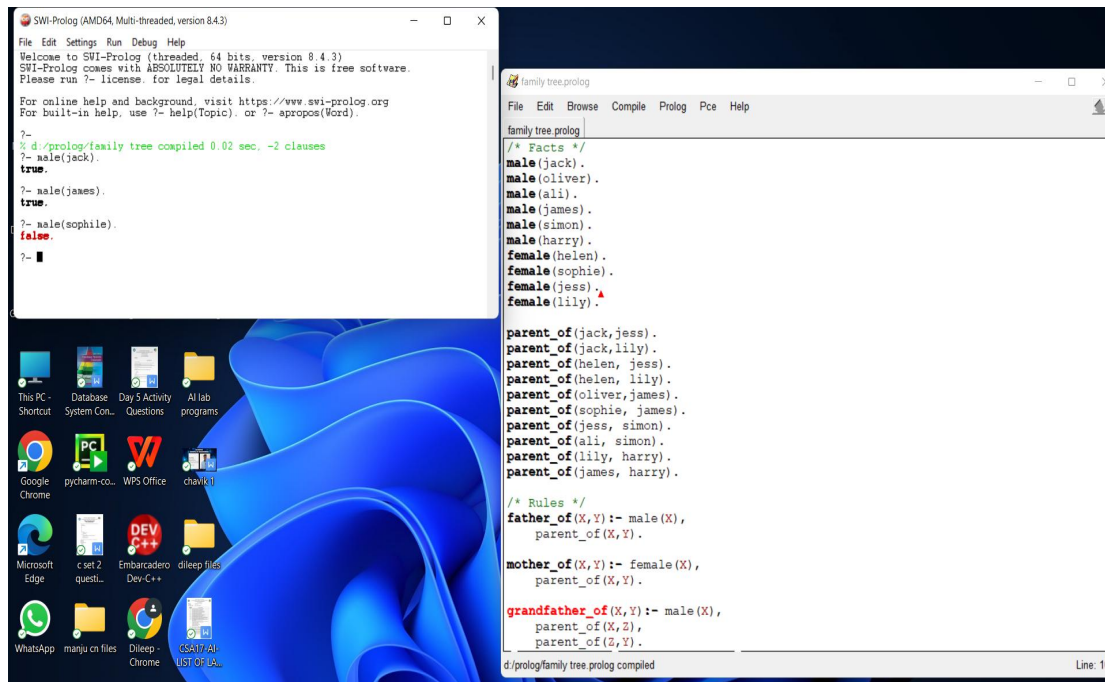
14. alpha Beta(output):



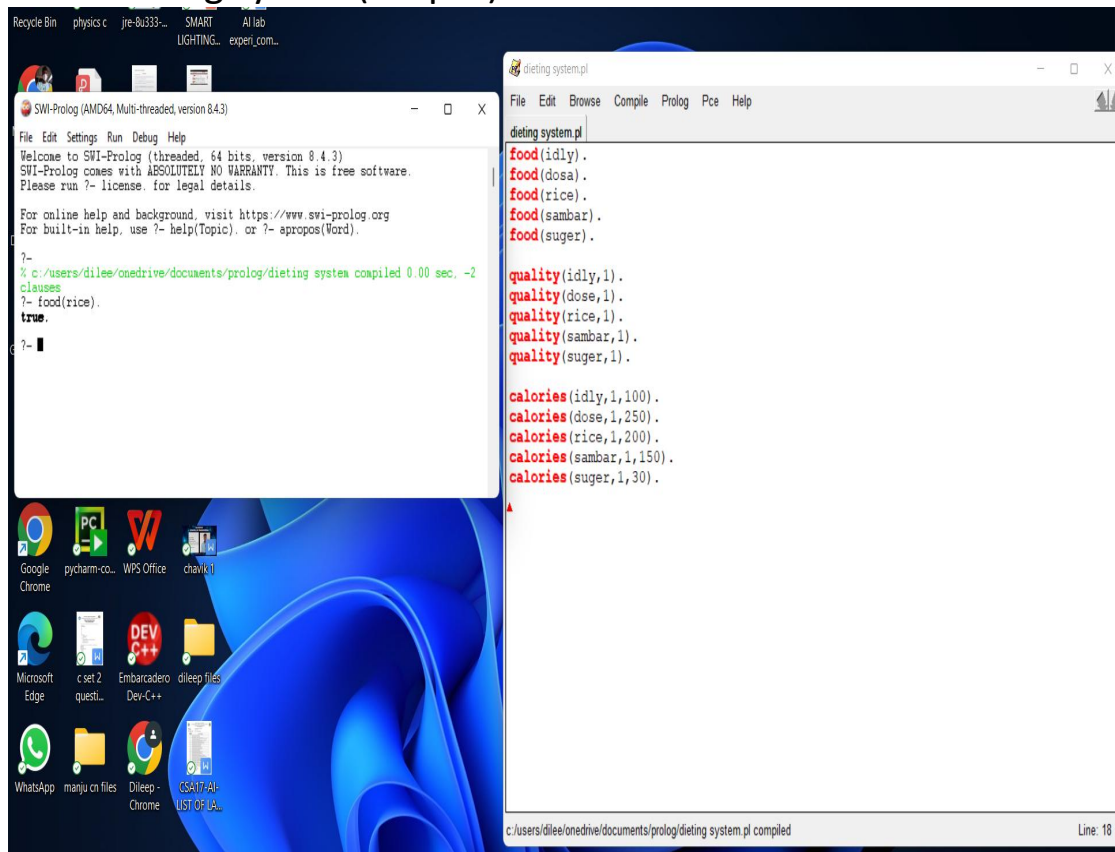
16.feed
forward(output):



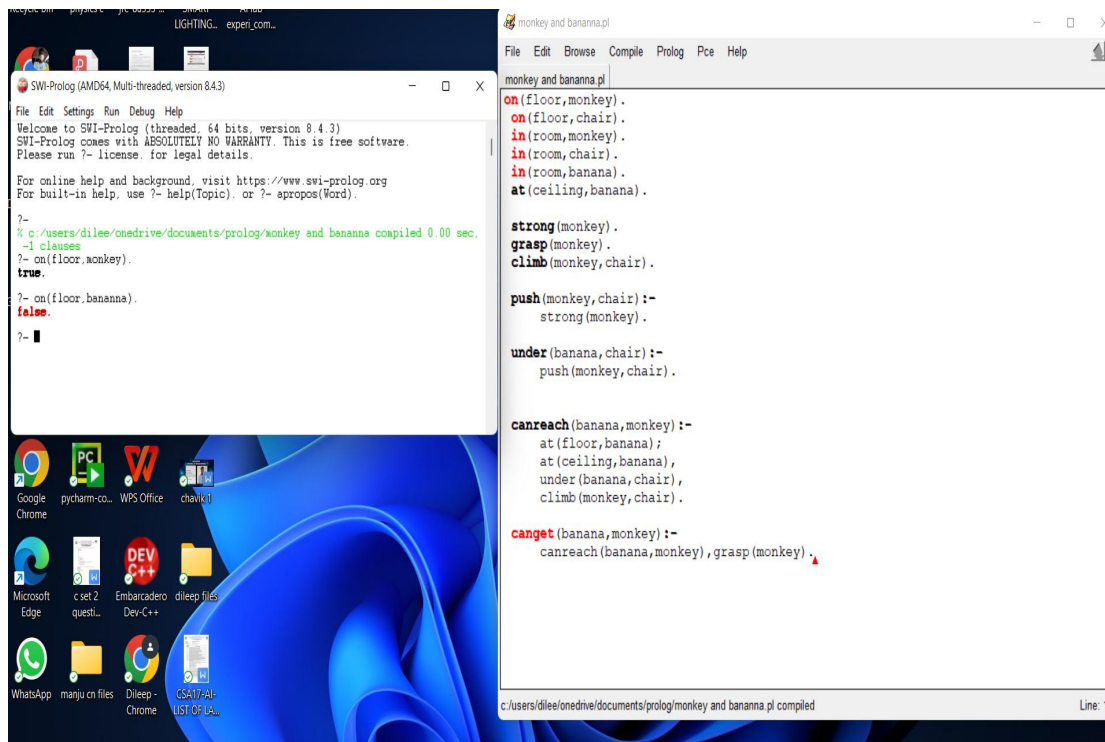
17.family tree(output):



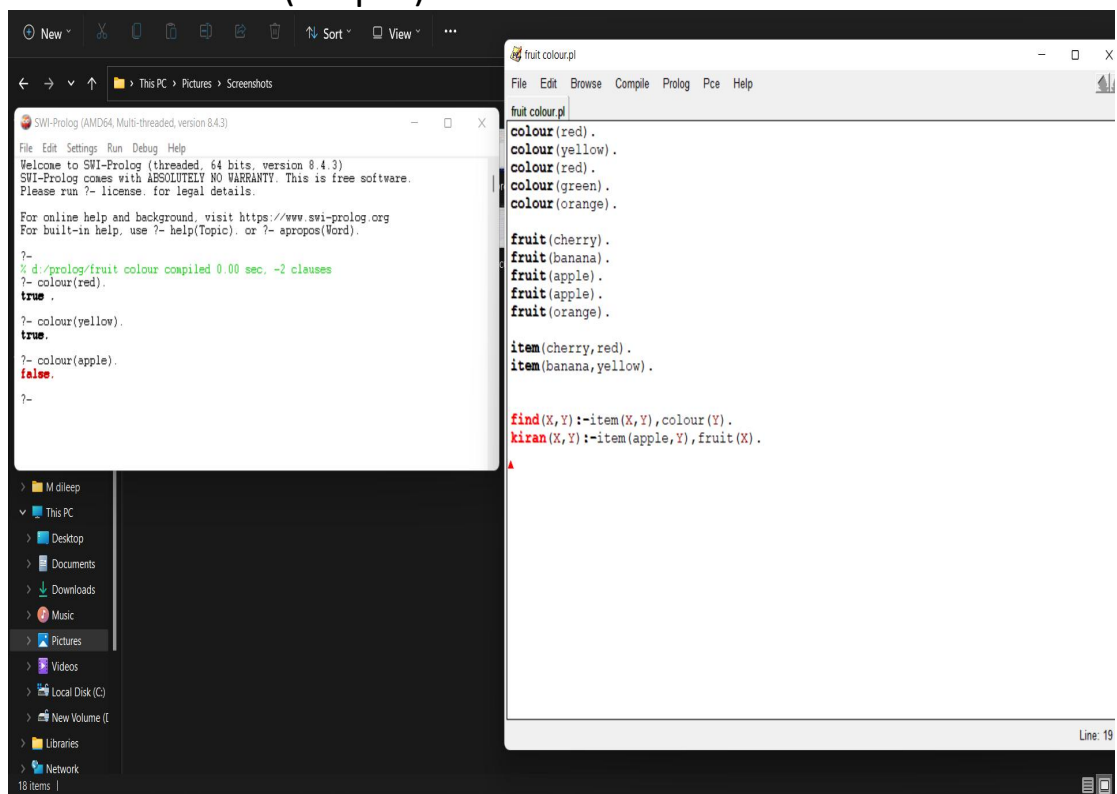
18. dieting system(output):



19. monkey &Bananna(output):



20. Fruit color(output):



21. Bfs prolog(output):

