Ahsanullah University of Science and Technology

Department of Computer Science and Engineering

Program: Bachelor of Science in Computer Science and Engineering

Course No: CSE 4108

Course Title: Artificial Intelligence Lab

Assignment No: 02

Date of Submission: 8/1/2022

Submitted to:

Mr. Faisal Muhammad Shah
Associate Professor, Department of CSE, AUST.

Mr. Md. Siam Ansary
Lecturer, Department of CSE, AUST.

Submitted by,

Name: Atanu Kumar Saha

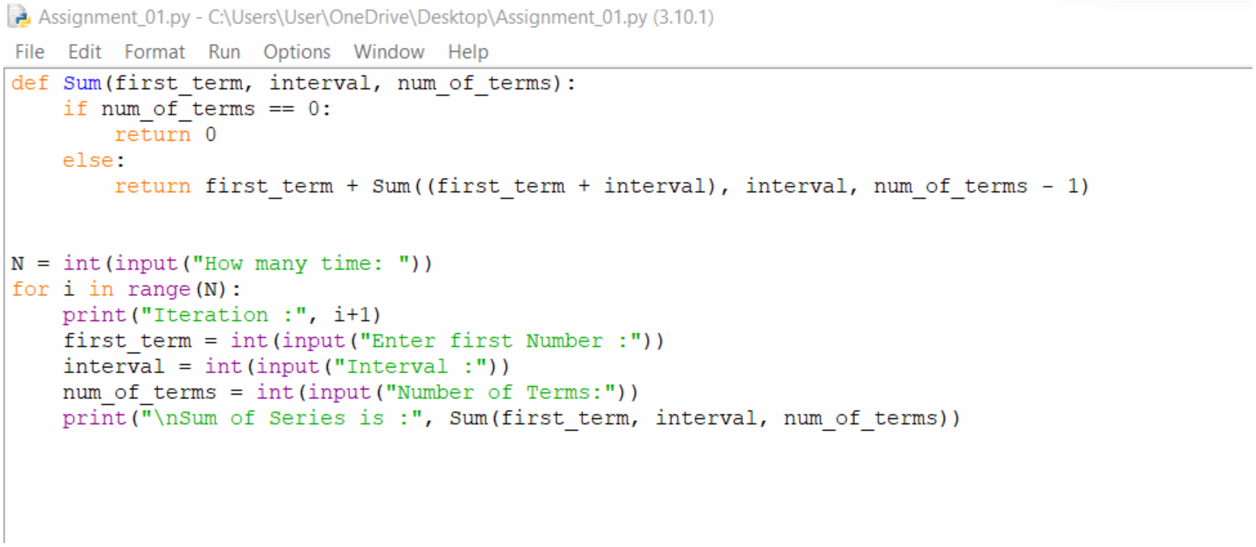
Student ID: 17.02.04.003

Question 1: Define a recursive procedure in Python to find the sum of 1st n terms of an equal-interval series given the 1st term and the interval.

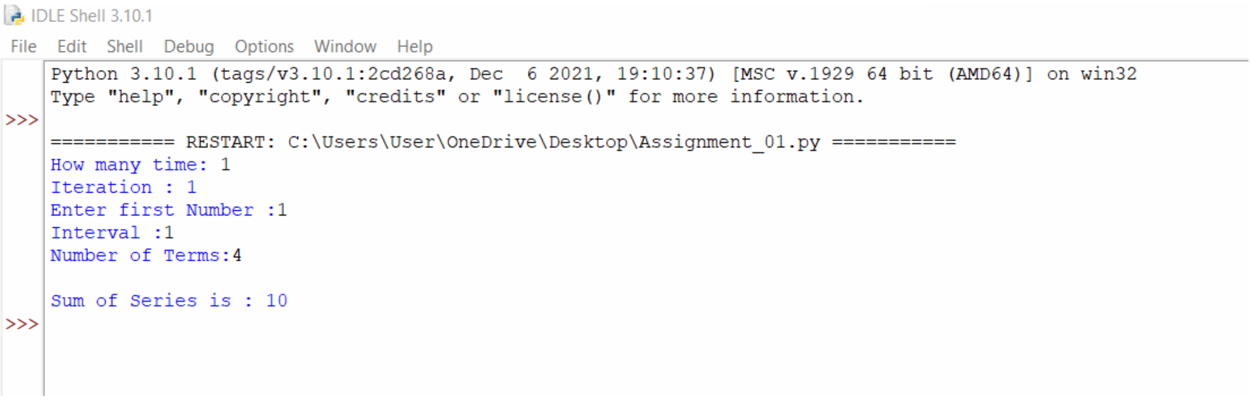
Solution:

Python Code:

```
def Sum(first_term, interval, num_of_terms):
    if num_of_terms == 0:
        return 0
    else:
        return first_term + Sum((first_term + interval), interval, 
num_of_terms - 1)


N = int(input("How many time: "))
for i in range(N):
    print("Iteration :", i+1)
    first_term = int(input("Enter first Number :"))
    interval = int(input("Interval :"))
    num_of_terms = int(input("Number of Terms:"))
    print("\nSum of Series is :", Sum(first_term, interval, 
num_of_terms))
```


Assignment_01.py - C:\Users\User\OneDrive\Desktop\Assignment_01.py (3.10.1)
File
Edit
Format
Run
Options
Window
Help

```
def Sum(first_term, interval, num_of_terms):
    if num_of_terms == 0:
        return 0
    else:
        return first_term + Sum((first_term + interval), interval, num_of_terms - 1)


N = int(input("How many time: "))
for i in range(N):
    print("Iteration :", i+1)
    first_term = int(input("Enter first Number :"))
    interval = int(input("Interval :"))
    num_of_terms = int(input("Number of Terms:"))
    print("\nSum of Series is :", Sum(first_term, interval, num_of_terms))
```



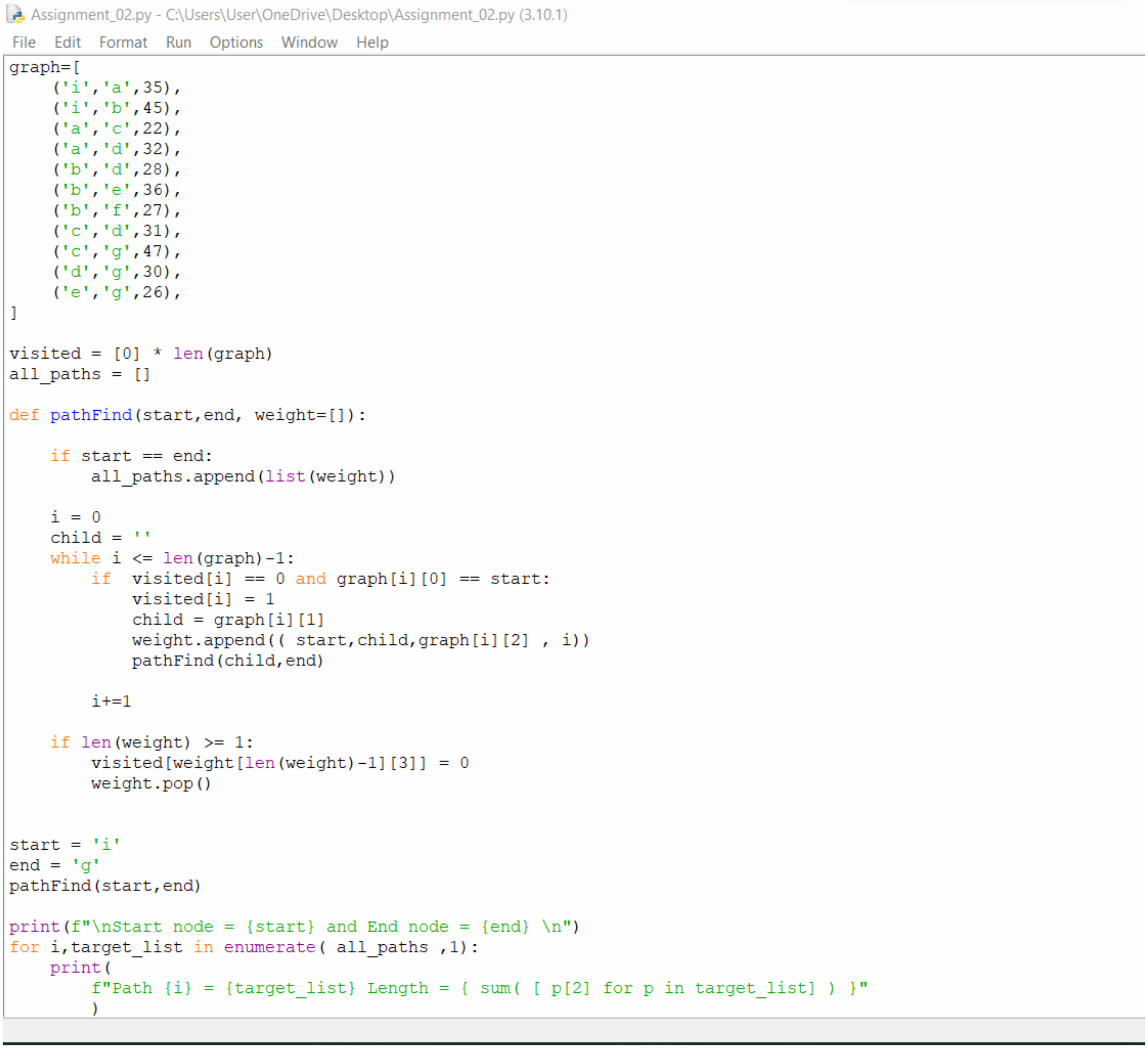
```
Python 3.10.1 (tags/v3.10.1:2cd268a, Dec  6 2021, 19:10:37) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
=========== RESTART: C:\Users\User\OneDrive\Desktop\Assignment_01.py ===========
How many time: 1
Iteration : 1
Enter first Number :1
Interval :1
Number of Terms:4

Sum of Series is : 10
>>>
```

Question 2: Define a recursive procedure in Python to find the length of a path between two vertices of a directed weighted graph.

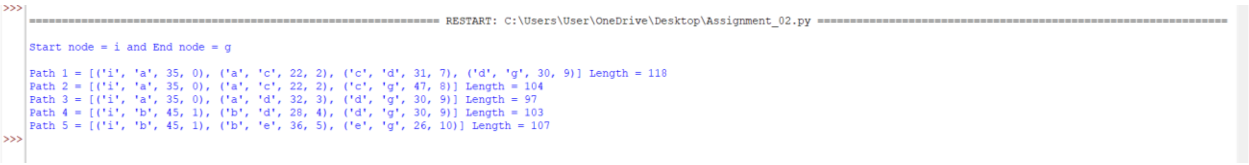
Solution:

Python Code:

```
 graph=[
    ('i','a',35),
    ('i','b',45),
    ('a','c',22),
    ('a','d',32),
    ('b','d',28),
    ('b','e',36),
    ('b','f',27),
    ('c','d',31),
    ('c','g',47),
    ('d','g',30),
    ('e','g',26),
]

visited = [0] * len(graph)
all_paths = []

def pathFind(start,end, weight=[]):

    if start == end:
        all_paths.append(list(weight))

    i = 0
    child = "
    while i <= len(graph)-1:
        if  visited[i] == 0 and graph[i][0] == start:
            visited[i] = 1
            child = graph[i][1]
            weight.append(( start,child,graph[i][2] , i))
            pathFind(child,end)

        i+=1

    if len(weight) >= 1:
        visited[weight[len(weight)-1][3]] = 0
        weight.pop()


start = 'i'
end = 'g'
pathFind(start,end)

print(f"\nStart node = {start} and End node = {end} \n")
for i,target_list in enumerate( all_paths ,1):
    print(
        f"Path {i} = {target_list} Length = { sum( [ p[2] for p in target_list] ) }"
        )
```


Assignment_02.py - C:\Users\User\OneDrive\Desktop\Assignment_02.py (3.10.1)
File
Edit
Format
Run
Options
Window
Help

```
graph=[
    ('i','a',35),
    ('i','b',45),
    ('a','c',22),
    ('a','d',32),
    ('b','d',28),
    ('b','e',36),
    ('b','f',27),
    ('c','d',31),
    ('c','g',47),
    ('d','g',30),
    ('e','g',26),
]

visited = [0] * len(graph)
all_paths = []

def pathFind(start,end, weight=[]):

    if start == end:
        all_paths.append(list(weight))

    i = 0
    child = ''
    while i <= len(graph)-1:
        if  visited[i] == 0 and graph[i][0] == start:
            visited[i] = 1
            child = graph[i][1]
            weight.append(( start,child,graph[i][2] , i))
            pathFind(child,end)

        i+=1

    if len(weight) >= 1:
        visited[weight[len(weight)-1][3]] = 0
        weight.pop()


start = 'i'
end = 'g'
pathFind(start,end)

print(f"\nStart node = {start} and End node = {end} \n")
for i,target_list in enumerate( all_paths ,1):
    print(
        f"Path {i} = {target_list} Length = { sum( [ p[2] for p in target_list] ) }"
        )
```



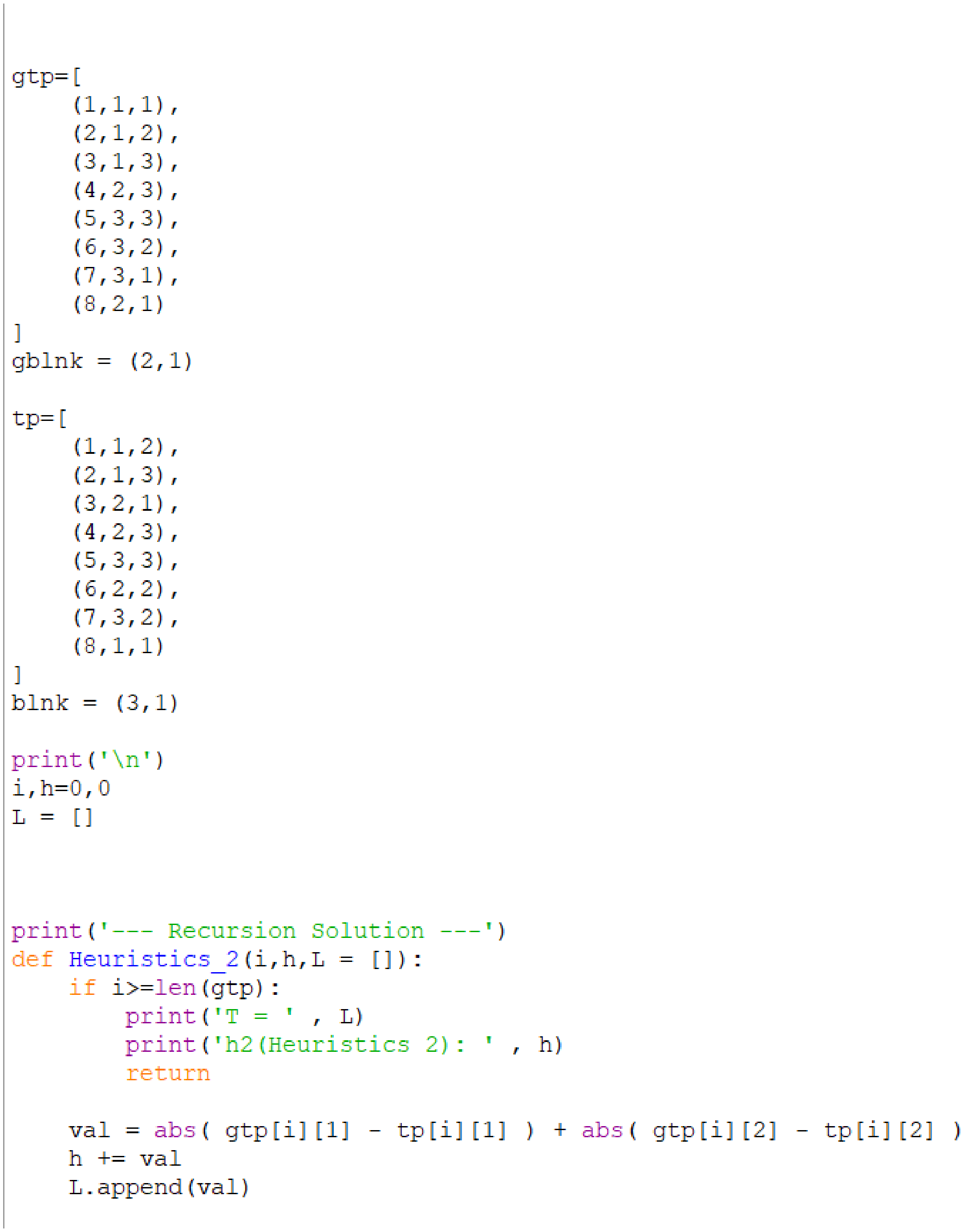
```
>>>
    ============================================================== RESTART: C:\Users\User\OneDrive\Desktop\Assignment_02.py ==============================================================

    Start node = i and End node = g

    Path 1 = [('i', 'a', 35, 0), ('a', 'c', 22, 2), ('c', 'd', 31, 7), ('d', 'g', 30, 9)] Length = 118
    Path 2 = [('i', 'a', 35, 0), ('a', 'c', 22, 2), ('c', 'g', 47, 8)] Length = 104
    Path 3 = [('i', 'a', 35, 0), ('a', 'd', 32, 3), ('d', 'g', 30, 9)] Length = 97
    Path 4 = [('i', 'b', 45, 1), ('b', 'd', 28, 4), ('d', 'g', 30, 9)] Length = 103
    Path 5 = [('i', 'b', 45, 1), ('b', 'e', 36, 5), ('e', 'g', 26, 10)] Length = 107
>>>
```

Question 3: Write a program in Python to calculate the heuristic for 8 puzzle problem where the heuristic is the Manhattan distance of the tiles.

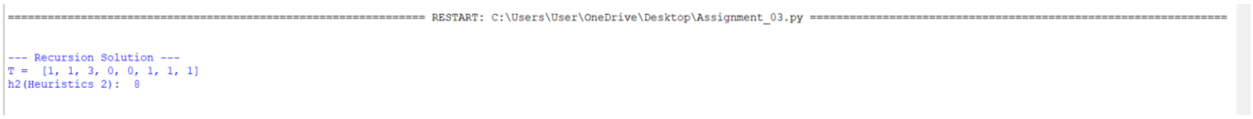
Solution:

Python Code:

```
'''
goal status :
1    2    3
8    _    4
7    6    5

current status :
8    1    2
3    6    4
_    7    5

'''


gtp=[
    (1,1,1),
    (2,1,2),
    (3,1,3),
    (4,2,3),
    (5,3,3),
    (6,3,2),
    (7,3,1),
    (8,2,1)
]
gblnk = (2,1)

tp=[
    (1,1,2),
    (2,1,3),
    (3,2,1),
    (4,2,3),
    (5,3,3),
    (6,2,2),
    (7,3,2),
    (8,1,1)
]
blnk = (3,1)

print('\n')
i,h=0,0
L = []



print('--- Recursion Solution ---')
def Heuristics_2(i,h,L = []):
    if i>=len(gtp):
        print('T = ' , L)
        print('h2(Heuristics 2): ' , h)
        return

    val = abs( gtp[i][1] - tp[i][1] ) + abs( gtp[i][2] - tp[i][2] )
    h += val
    L.append(val)

    i += 1
    Heuristics_2(i , h)


Heuristics_2(0,0)
```



```python
gtp=[
    (1,1,1),
    (2,1,2),
    (3,1,3),
    (4,2,3),
    (5,3,3),
    (6,3,2),
    (7,3,1),
    (8,2,1)
]
gblnk = (2,1)

tp=[
    (1,1,2),
    (2,1,3),
    (3,2,1),
    (4,2,3),
    (5,3,3),
    (6,2,2),
    (7,3,2),
    (8,1,1)
]
blnk = (3,1)

print('\n')
i,h=0,0
L = []



print('--- Recursion Solution ---')
def Heuristics_2(i,h,L = []):
    if i>=len(gtp):
        print('T = ' , L)
        print('h2(Heuristics 2): ' , h)
        return

    val = abs( gtp[i][1] - tp[i][1] ) + abs( gtp[i][2] - tp[i][2] )
    h += val
    L.append(val)
```



```
================================================================ RESTART: C:\Users\User\OneDrive\Desktop\Assignment_03.py ================================================================


--- Recursion Solution ---
T =  [1, 1, 3, 0, 0, 1, 1, 1]
h2(Heuristics 2):  8
```

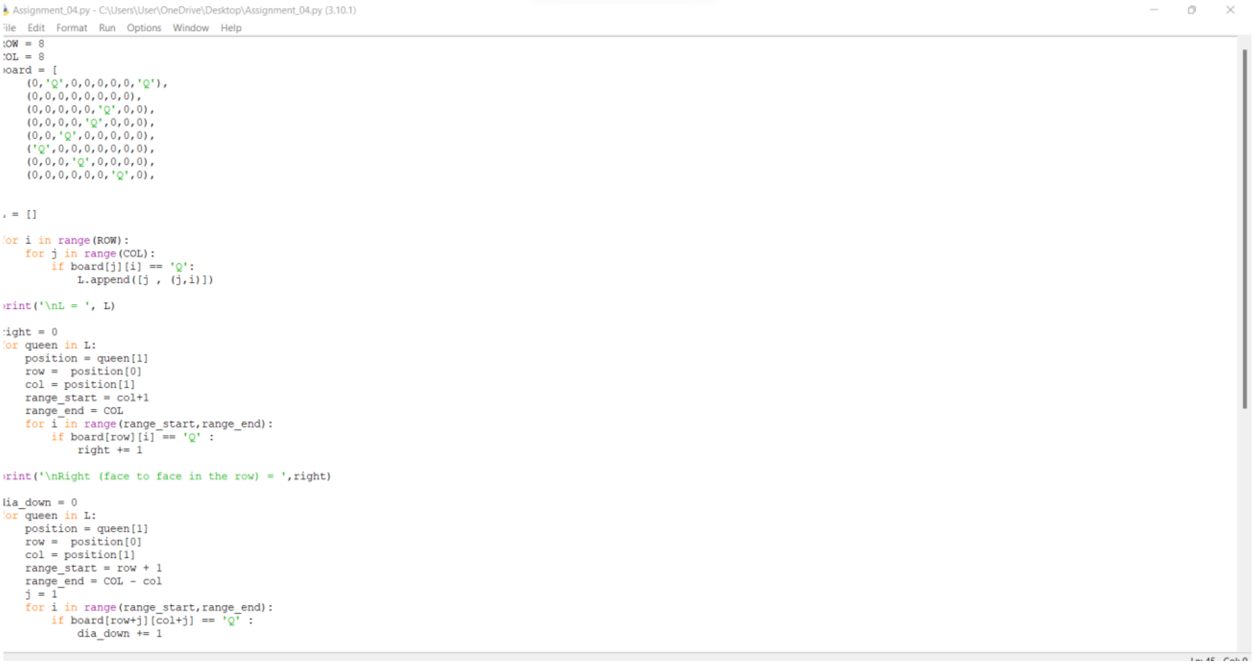
Question 4: Write a program in Python to calculate the heuristic for 8 queen problem where the heuristic is the number of attacking pairs..

Solution:

Python Code:

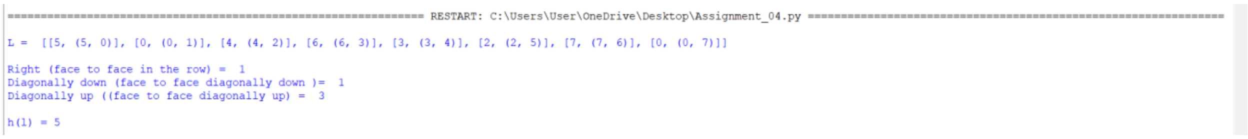
```
ROW = 8
COL = 8
board = [
    (0,'Q',0,0,0,0,0,'Q'),
    (0,0,0,0,0,0,0,0),
    (0,0,0,0,0,'Q',0,0),
    (0,0,0,0,'Q',0,0,0),
    (0,0,'Q',0,0,0,0,0),
    ('Q',0,0,0,0,0,0,0,0),
    (0,0,0,'Q',0,0,0,0),
    (0,0,0,0,0,0,'Q',0),
]

L = []

for i in range(ROW):
    for j in range(COL):
        if board[j][i] == 'Q':
            L.append([j , (j,i)])

print('\nL = ', L)

right = 0
for queen in L:
    position = queen[1]
    row =  position[0]
    col = position[1]
    range_start = col+1
    range_end = COL
    for i in range(range_start,range_end):
        if board[row][i] == 'Q' :
            right += 1

print('\nRight (face to face in the row) = ',right)
```

```
dia_down = 0
for queen in L:
    position = queen[1]
    row =  position[0]
    col = position[1]
    range_start = row + 1
    range_end = COL - col
    j = 1
    for i in range(range_start,range_end):
        if board[row+j][col+j] == 'Q' :
            dia_down += 1

        j += 1

print('Diagonally down (face to face diagonally down )= ',dia_down)



dia_up = 0
for queen in L:
    position = queen[1]
    row =  position[0]
    col = position[1]
    range_start = 0
    range_end =  row
    j = 1

    for i in reversed(range(range_start,range_end)):
        if i == -1 or col+j == COL:
            break
        if board[i][col+j] == 'Q' :
            dia_up += 1
        j += 1


print('Diagonally up ((face to face diagonally up) = ',dia_up)

print(f"\nh(l) = {right + dia_down + dia_up}")
```



Assignment_04.py - C:\Users\User\OneDrive\Desktop\Assignment_04.py (3.10.1)

```
OW = 8
OL = 8
oard = [
    (0,'Q',0,0,0,0,0,'Q'),
    (0,0,0,0,0,0,0,0),
    (0,0,0,0,0,'Q',0,0),
    (0,0,0,0,'Q',0,0,0),
    (0,0,'Q',0,0,0,0,0),
    ('Q',0,0,0,0,0,0,0),
    (0,0,0,'Q',0,0,0,0),
    (0,0,0,0,0,0,'Q',0),


 = []

or i in range(ROW):
    for j in range(COL):
        if board[j][i] == 'Q':
            L.append([j , (j,i)])

rint('\nL = ', L)

ight = 0
or queen in L:
    position = queen[1]
    row =  position[0]
    col = position[1]
    range_start = col+1
    range_end = COL
    for i in range(range_start,range_end):
        if board[row][i] == 'Q' :
            right += 1

rint('\nRight (face to face in the row) = ',right)

ia_down = 0
or queen in L:
    position = queen[1]
    row =  position[0]
    col = position[1]
    range_start = row + 1
    range_end = COL - col
    j = 1
    for i in range(range_start,range_end):
        if board[row+j][col+j] == 'Q' :
            dia_down += 1
```



```
=============================================================== RESTART: C:\Users\User\OneDrive\Desktop\Assignment_04.py ===============================================================

L =  [[5, (5, 0)], [0, (0, 1)], [4, (4, 2)], [6, (6, 3)], [3, (3, 4)], [2, (2, 5)], [7, (7, 6)], [0, (0, 7)]]

Right (face to face in the row) =  1
Diagonally down (face to face diagonally down )=  1
Diagonally up ((face to face diagonally up) =  3

h(1) = 5
```