

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: 08 / 01 / 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: Nusrat Jahan

Student ID: 180104020

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.

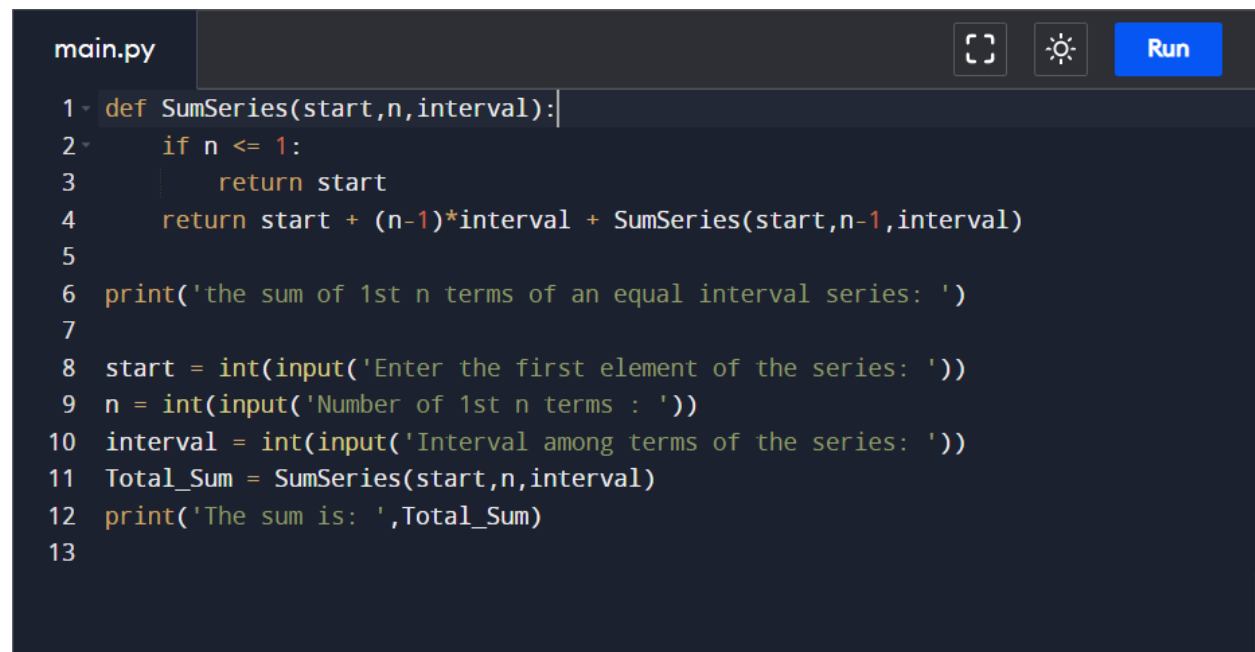
Solution:

Python Code:

```
def SumSeries(start,n,interval):
    if n <= 1:
        return start
    return start + (n-1)*interval + SumSeries(start,n-1,interval)

print('the sum of 1st n terms of an equal interval series: ')

start = int(input('Enter the first element of the series: '))
n = int(input('Number of 1st n terms : '))
interval = int(input('Interval among terms of the series: '))
Total_Sum = SumSeries(start,n,interval)
print('The sum is: ',Total_Sum)
```

A screenshot of a Python IDE window titled 'main.py'. The code is as follows:

```
1 def SumSeries(start,n,interval):
2     if n <= 1:
3         return start
4     return start + (n-1)*interval + SumSeries(start,n-1,interval)
5
6 print('the sum of 1st n terms of an equal interval series: ')
7
8 start = int(input('Enter the first element of the series: '))
9 n = int(input('Number of 1st n terms : '))
10 interval = int(input('Interval among terms of the series: '))
11 Total_Sum = SumSeries(start,n,interval)
12 print('The sum is: ',Total_Sum)
13
```

The IDE interface includes a file explorer on the left, a toolbar with icons for running, debugging, and testing, and a 'Run' button on the right.

Shell



Clear

```
the sum of 1st n terms of an equal interval series:  
Enter the first element of the series: 1  
Number of 1st n terms : 5  
Interval among terms of the series: 2  
The sum is: 25  
> |
```

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

Python Code:

```
def pathlength(src,des,cost = 0):  
    if(src,des) in edges:  
        print(str(cost + edgeValue[(src,des)]) + ' ' )  
  
    for(i,j) in edges:  
        if i == src:  
            pathlength(j,des, cost + edgeValue[(i,j)])  
  
edges = [('a', 'b'), ('a', 'c'), ('b', 'e'), ('b', 'f'), ('c', 'f'),  
('c', 'g'), ('c', 'h'), ('e', 'f'), ('e', 'i'), ('f', 'i')]  
  
edgeValue = {('a', 'b') : 30, ('a', 'c') : 45, ('b', 'e') : 20, ('b',  
'f') : 39,  
              ('c', 'f') : 21, ('c', 'g') : 31, ('c', 'h') : 27, ('e',  
'f') : 35,  
              ('e', 'i') : 47, ('f', 'i') : 20}  
  
src = str(input('Enter Starting Node: '))  
des = str(input('Enter Ending Node: '))  
print('The length of path is: ' )  
pathlength(src, des)
```

```
main.py   Run

1
2 def pathlength(src,des,cost = 0):
3     if(src,des) in edges:
4         print(str(cost + edgeValue[(src,des)]) + ' ')
5
6     for(i,j) in edges:
7         if i == src:
8             pathlength(j,des, cost + edgeValue[(i,j)])
9
10 edges = [('a', 'b'), ('a', 'c'), ('b', 'e'), ('b', 'f'), ('c', 'f'), ('c', 'g'),
11          ('c', 'h'), ('e', 'f'), ('e', 'i'), ('f', 'i')]
12
13 edgeValue = {('a', 'b') : 30, ('a', 'c') : 45, ('b', 'e') : 20, ('b', 'f') : 39,
14              ('c', 'f') : 21, ('c', 'g') : 31, ('c', 'h') : 27, ('e', 'f') : 35,
15              ('e', 'i') : 47, ('f', 'i') : 20}
16
17 src = str(input('Enter Starting Node: '))
18 des = str(input('Enter Ending Node: '))
19 print('The length of path is: ')
20 pathlength(src, des)
```

```
Shell Clear

Enter Starting Node: a
Enter Ending Node: e
The length of path is:
50
> |
```

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.

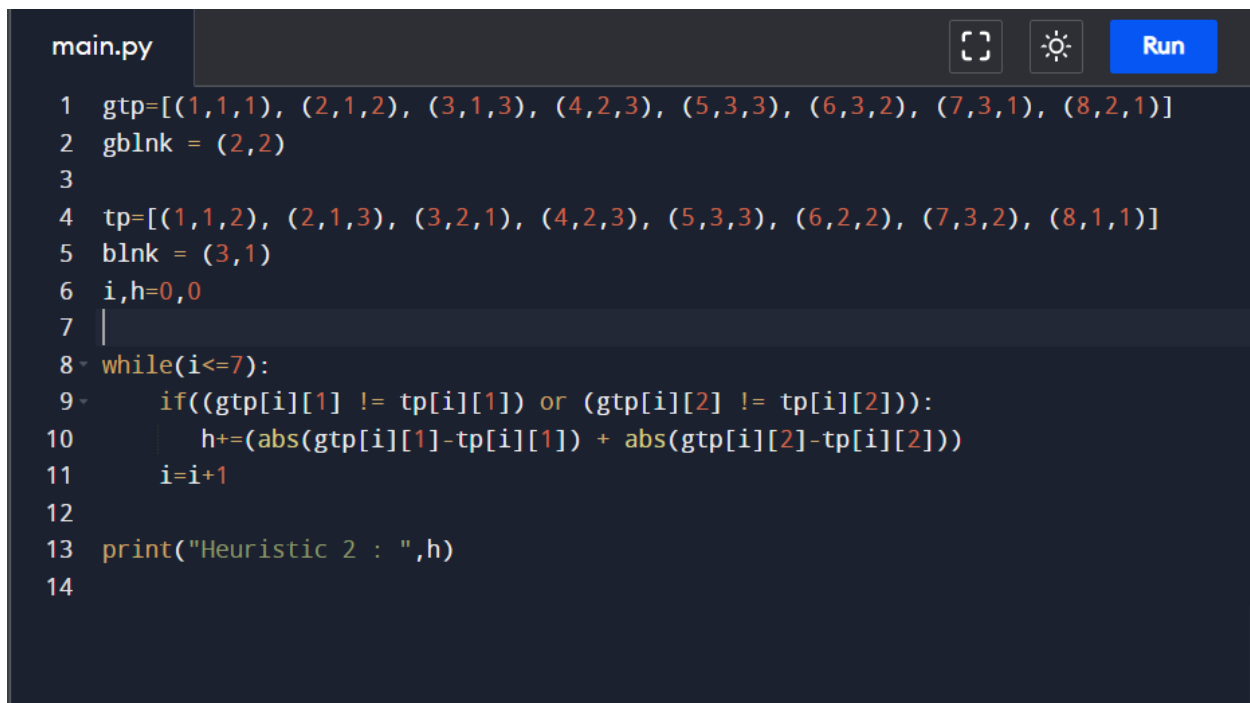
Python Code:

```
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 = (2,2)

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)
i,h=0,0

while(i<=7):
    if((gtp[i][1] != tp[i][1]) or (gtp[i][2] != tp[i][2])):
        h+=(abs(gtp[i][1]-tp[i][1]) + abs(gtp[i][2]-tp[i][2]))
    i=i+1

print("Heuristic 2 : ",h)
```

A screenshot of a Python IDE window titled 'main.py'. The code is as follows:

```
1 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)]
2 gblnk = (2,2)
3
4 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)]
5 blnk = (3,1)
6 i,h=0,0
7
8 while(i<=7):
9     if((gtp[i][1] != tp[i][1]) or (gtp[i][2] != tp[i][2])):
10         h+=(abs(gtp[i][1]-tp[i][1]) + abs(gtp[i][2]-tp[i][2]))
11     i=i+1
12
13 print("Heuristic 2 : ",h)
14
```

The IDE interface includes a file explorer on the left showing 'main.py', a toolbar on the right with icons for file operations and a 'Run' button, and a dark-themed code editor with line numbers and syntax highlighting.

```
Shell Clear  
Heuristic 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.

Python Code:

```
def hCol(pos):  
    for i in range(len(pos)):  
        countHorizontal(i, pos)  
  
def duCol(pos):  
    for i in range(len(pos)):  
        countDiagonalUp(i, pos)  
def ddCol(pos):  
    for i in range(len(pos)):  
        countDiagonalDown(i, pos)  
  
def countHorizontal(point, pos):  
    global count  
    for i in range(point + 1, len(pos)):  
        if pos[i] == pos[point]:  
            count += 1  
  
def countDiagonalUp(point, pos):  
    global count  
    for i in range(point + 1, len(pos)):  
        if pos[i] == pos[point] + i - point:  
            count += 1  
  
def countDiagonalDown(point, pos):  
    global count  
    for i in range(point + 1, len(pos)):  
        if pos[i] == pos[point] - i + point:  
            count += 1
```

```

count = 0
pos = [0]*8
for i in range(8):
    pos[i] = int(input('Enter Position for Queen in Column ' + str(i+1) +
': '))

hCol(pos)
duCol(pos)
ddCol(pos)
print('Total Number of Collisions: ', count)

```

main.py

Run

```

1
2 def hCol(pos):
3     for i in range(len(pos)):
4         countHorizontal(i, pos)
5
6 def duCol(pos):
7     for i in range(len(pos)):
8         countDiagonalUp(i, pos)
9 def ddCol(pos):
10    for i in range(len(pos)):
11        countDiagonalDown(i, pos)
12
13 def countHorizontal(point, pos):
14     global count
15     for i in range(point + 1, len(pos)):
16         if pos[i] == pos[point]:
17             count += 1
18
19 def countDiagonalUp(point, pos):
20     global count
21     for i in range(point + 1, len(pos)):
22         if pos[i] == pos[point] + i - point:
23             count += 1

```

```

25 def countDiagonalDown(point, pos):
26     global count
27     for i in range(point + 1, len(pos)):
28         if pos[i] == pos[point] - i + point:
29             count += 1
30
31     count = 0
32     pos = [0]*8
33     for i in range(8):
34         pos[i] = int(input('Enter Position for Queen in Column ' + str(i+1) + ': '))
35
36     hCol(pos)
37     duCol(pos)
38     ddCol(pos)
39     print('Total Number of Collisions: ', count)

```

Shell

Clear

```

Enter Position for Queen in Column 1: 6
Enter Position for Queen in Column 2: 1
Enter Position for Queen in Column 3: 5
Enter Position for Queen in Column 4: 7
Enter Position for Queen in Column 5: 4
Enter Position for Queen in Column 6: 3
Enter Position for Queen in Column 7: 8
Enter Position for Queen in Column 8: 1
Total Number of Collisions: 5
> |

```