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.

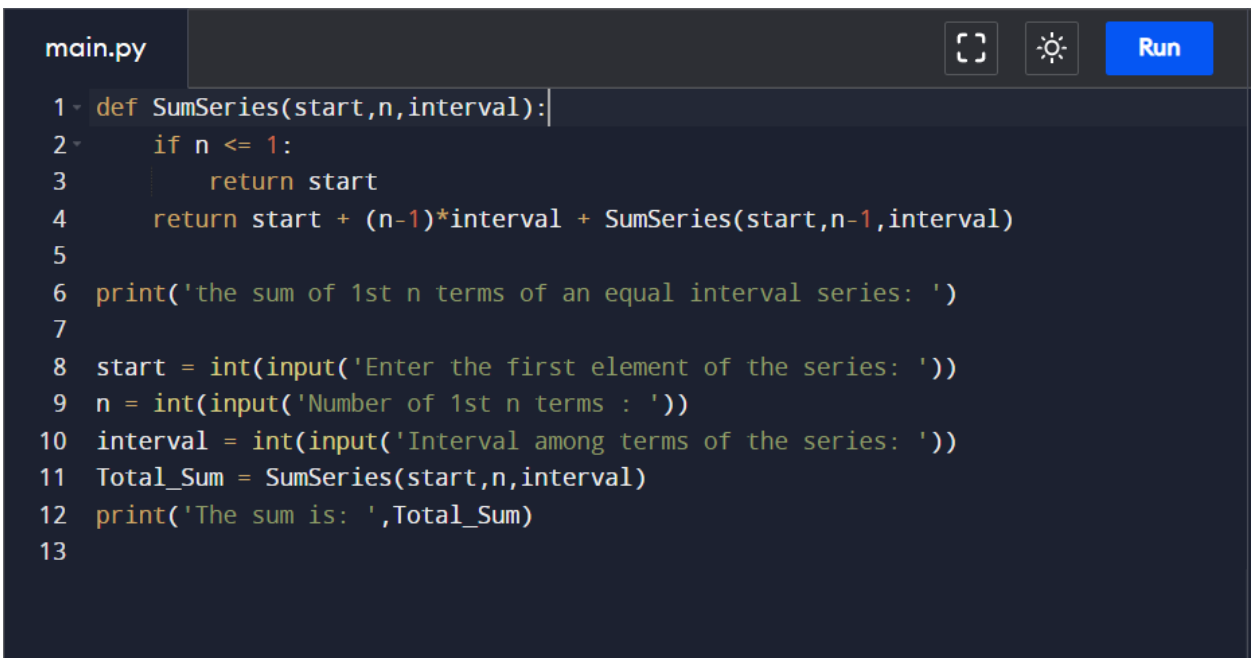
Solution:

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)
```



```
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.

<u>Python Code:</u>

```
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

```python

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)
```

Shell

```
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 theheuristic is the Manhattan distance of the tiles.

<u>Python Code:</u>

```
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)
```

main.py

Run

```
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)
```

```
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.

<u>Python Code:</u>

```
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

```

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
```

```python
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)
```

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
>
```