Started on	Wednesday, 26 March 2025, 9:24 AM
State	Finished
Completed on	Wednesday, 26 March 2025, 11:52 AM
Time taken	2 hours 28 mins
Overdue	28 mins 3 secs
Grade	**80.00** out of 100.00

Question **1**

Not answered

Mark 0.00 out of 20.00

Write a Python program to calculate the harmonic sum of n-1.
Note: The harmonic sum is the sum of reciprocals of the positive integers.

Example:

$$1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\frac{1}{5}+\cdots$$

For example:

Input	Result
5	2.283333333333333
7	2.5928571428571425

Answer: (penalty regime: 0 %)

```

```

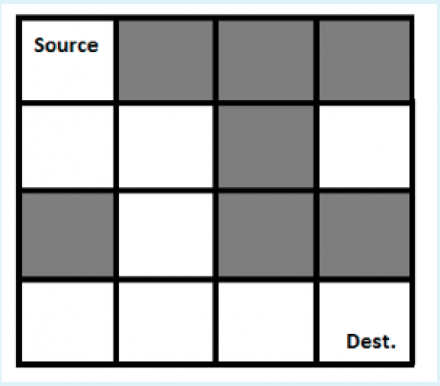
Question **2**

Correct

Mark 20.00 out of 20.00

Rat In A Maze Problem

You are given a maze in the form of a matrix of size n * n. Each cell is either clear or blocked denoted by 1 and 0 respectively. A rat sits at the top-left cell and there exists a block of cheese at the bottom-right cell. Both these cells are guaranteed to be clear. You need to find if the rat can get the cheese if it can move only in one of the two directions - down and right. It can't move to blocked cells.



Provide the solution for the above problem Consider n=4)

The output (Solution matrix) must be 4*4 matrix with value "1" which indicates the path to destination and "0" for the cell indicating the absence of the path to destination.

Answer: (penalty regime: 0 %)

Reset answer

```
N = 4
def printSolution( sol ):
    for i in sol:
        for j in i:
            print(str(j) + " ", end ="")
        print("")
def isSafe( maze, x, y ):
    if x >= 0 and x < N and y >= 0 and y < N and maze[x][y] == 1:
        return True
    return False
def solveMaze( maze ):
    sol = [ [ 0 for j in range(4) ] for i in range(4) ]
    if solveMazeUtil(maze, 0, 0, sol) == False:
        print("Solution doesn't exist");
        return False
    printSolution(sol)
    return True
def solveMazeUtil(maze, x, y, sol):
    if x==N-1 and y==N-1  and maze[x][y]==1:
        sol[x][y]=1
        return True
    if isSafe( maze, x, y ):
```

	Expected	Got	
✓	1 0 0 0 1 1 0 0 0 1 0 0 0 1 1 1	1 0 0 0 1 1 0 0 0 1 0 0 0 1 1 1	✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.

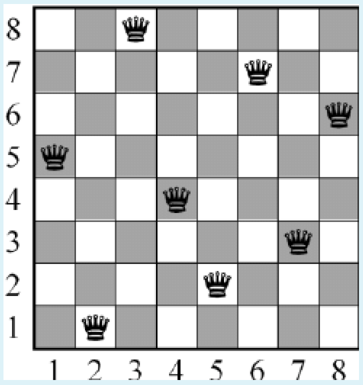
Question **3**

Correct

Mark 20.00 out of 20.00

You are given an integer **N**. For a given **N** x **N** chessboard, find a way to place **'N'** queens such that no queen can attack any other queen on the chessboard.

A queen can be attacked when it lies in the same row, column, or the same diagonal as any of the other queens. **You have to print one such configuration**.



Note :

Get the input from the user for N . The value of N must be from 1 to 8

If solution exists Print a binary matrix as output that has 1s for the cells where queens are placed

If there is no solution to the problem print "Solution does not exist"

For example:

Input	Result

5	1 0 0 0 0
	0 0 0 1 0
	0 1 0 0 0
	0 0 0 0 1
	0 0 1 0 0

Answer: (penalty regime: 0 %)

Reset answer

```
global N
N = int(input())
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:
```

	Input	Expected	Got	

✓	5	`1 0 0 0 0`
		`0 0 0 1 0`
		`0 1 0 0 0`
		`0 0 0 0 1`
		`0 0 1 0 0`
			`1 0 0 0 0`
			`0 0 0 1 0`
			`0 1 0 0 0`
			`0 0 0 0 1`
			`0 0 1 0 0`
				✓

✓	2	`Solution does not exist`	`Solution does not exist`	✓

✓	8	`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`
			`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`
				✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.

Question **4**

Correct

Mark 20.00 out of 20.00

SUBSET SUM PROBLEM

COUNT OF SUBSETS WITH SUM EQUAL TO X

Given an array arr[] of length N and an integer X, the task is to find the number of subsets with a sum equal to X.

Examples:

Input: *arr[] = {1, 2, 3, 3}, X = 6*
Output: *3*
All the possible subsets are {1, 2, 3},
{1, 2, 3} and {3, 3}

Input: *arr[] = {1, 1, 1, 1}, X = 1*
Output: *4*

THE INPUT

1.No of numbers

2.Get the numbers

3.Sum Value

For example:

Input	Result
4 2 4 5 9 15	1
6 3 34 4 12 3 2 7	2

Answer: (penalty regime: 0 %)

Reset answer

```python
def subsetSum(arr, n, i,sum, count):
    if(i==n):
        if(sum==0):
            count+=1
        return count
    count=subsetSum(arr,n,i+1,sum-arr[i],count)
    count=subsetSum(arr,n,i+1,sum,count)
    return count
arr=[]
size=int(input())
for j in range(size):
    value=int(input())
    arr.append(value)
sum = int(input())
n = len(arr)
print(subsetSum(arr, n, 0, sum, 0))
```

	Input	Expected	Got	
✓	4 2 4 5 9 15	1	1	✓
✓	6 10 20 25 50 70 90 80	2	2	✓
✓	5 4 16 5 23 12 9	1	1	✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.

Question **5**

Correct

Mark 20.00 out of 20.00

GRAPH COLORING PROBLEM

Given an undirected graph and a number m, determine if the graph can be coloured with at most m colours such that no two adjacent vertices of the graph are colored with the same color. Here coloring of a graph means the assignment of colors to all vertices.

Input-Output format:

Input:

1. A 2D array graph[V][V] where V is the number of vertices in graph and graph[V][V] is an adjacency matrix representation of the graph. A value graph[i][j] is 1 if there is a direct edge from i to j, otherwise graph[i][j] is 0.
2. An integer m is the maximum number of colors that can be used.

Output:

An array color[V] that should have numbers from 1 to m. color[i] should represent the color assigned to the ith vertex.

Example:

```
Input:
graph = {0, 1, 1, 1},
        {1, 0, 1, 0},
        {1, 1, 0, 1},
        {1, 0, 1, 0}
Output:
Solution Exists:
Following are the assigned colors
 1  2  3  2
Explanation: By coloring the vertices
with following colors, adjacent
vertices does not have same colors
```

```
Input:
graph = {1, 1, 1, 1},
        {1, 1, 1, 1},
        {1, 1, 1, 1},
        {1, 1, 1, 1}
Output: Solution does not exist.
Explanation: No solution exits.
```

Answer: (penalty regime: 0 %)

```
class Graph():
    def __init__(self,vertices):
        self.V=vertices
        self.graph=[[0 for column in range(vertices)]for row in range(vertices)]

    def isSafe(self,v,colour,c):
        for i in range(self.V):
            if self.graph[v][i]==1 and colour[i]==c:
                return False
        return True
    def graphColourUtil(self,m,colour,v):
        if v==self.V:
            return True
        for c in range(1,m+1):
            if self.isSafe(v,colour,c)==True:
                colour[v]=c
                if self.graphColourUtil(m,colour,v+1)==True:
                    return True
                colour[v]=0

    def graphColouring(self, m):
        colour = [0] * self.V
```

	Test	Expected	Got	

✓	`g = Graph(4)`
	`g.graph = [[0, 1, 1, 1], [1, 0, 1, 0], [1, 1, 0, 1], [1, 0, 1, 0]]`
	`m = 3`
	`g.graphColouring(m)`
		`Solution exist and Following are the assigned colours:`
		`1 2 3 2`
			`Solution exist and Following are the assigned colours:`
			`1 2 3 2`
				✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.