Started on	Friday, 9 May 2025, 1:24 PM
State	Finished
Completed on	Friday, 9 May 2025, 2:03 PM
Time taken	38 mins 42 secs
Grade	**80.00** out of 100.00

Question **1**

Correct

Mark 20.00 out of 20.00

Write a python program to implement the quick sort using recursion.

For example:

Input	Result
5 10 30 1 5 6	pivot: 6 pivot: 5 pivot: 10 [1, 5, 6, 10, 30]
6 21 30 4 5 61 70	pivot: 70 pivot: 61 pivot: 5 pivot: 30 [4, 5, 21, 30, 61, 70]

Answer: (penalty regime: 0 %)

```
def part(arr,l,h):
    p=arr[h]
    i=l-1
    print(f"pivot:  {p}")
    for j in range(l,h):
        if(arr[j]<=p):
            i+=1
            arr[i],arr[j]=arr[j],arr[i]
    arr[i+1],arr[h]=arr[h],arr[i+1]
    return i+1
def qs(arr,l,h):
    if(l<h):
        p=part(arr,l,h)
        qs(arr,l,p-1)
        qs(arr,p+1,h)
arr=[]
n=int(input())
for i in range(n):
    arr.append(int(input()))
qs(arr,0,n-1)
print(arr)
```

	Input	Expected	Got	
✔	5 10 30 1 5 6	pivot: 6 pivot: 5 pivot: 10 [1, 5, 6, 10, 30]	pivot: 6 pivot: 5 pivot: 10 [1, 5, 6, 10, 30]	✔
✔	6 21 30 4 5 61 70	pivot: 70 pivot: 61 pivot: 5 pivot: 30 [4, 5, 21, 30, 61, 70]	pivot: 70 pivot: 61 pivot: 5 pivot: 30 [4, 5, 21, 30, 61, 70]	✔
✔	4 20 34 5 10	pivot: 10 pivot: 34 [5, 10, 20, 34]	pivot: 10 pivot: 34 [5, 10, 20, 34]	✔

	Input	Expected	Got	
✔	8 26 14 51 32 20 71 80 9	pivot: 9 pivot: 26 pivot: 20 pivot: 32 pivot: 51 pivot: 71 [9, 14, 20, 26, 32, 51, 71, 80]	pivot: 9 pivot: 26 pivot: 20 pivot: 32 pivot: 51 pivot: 71 [9, 14, 20, 26, 32, 51, 71, 80]	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

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 ):

    # Creating a 4 * 4 2-D list
```

	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.

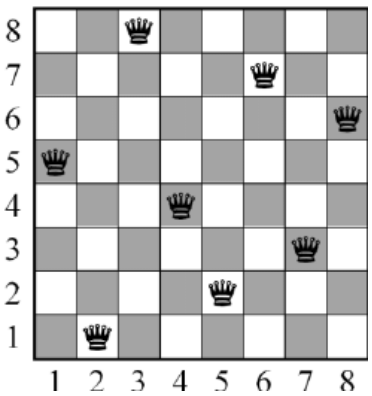
Question **3**

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):

    # Check this row on left side
    for i in range(col):
        if board[row][i] == 1:
            return False

    # Check upper diagonal on left side
    for i, j in zip(range(row, -1, -1),
                    range(col, -1, -1)):
        if board[i][j] == 1:
            return False

```

	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	✔

	Input	Expected	Got	
✔	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

```
from itertools import combinations
def subsetSum(n,arr,x):
    c=0
    for i in range(n):
        for subsetSum in combinations(arr,i):
            if(sum(subsetSum)==x):
                c+=1
    print(c)

arr=[]
n=int(input())
for j in range(n):
    arr.append(int(input()))
x=int(input())
subsetSum(n,arr,x)
```

	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

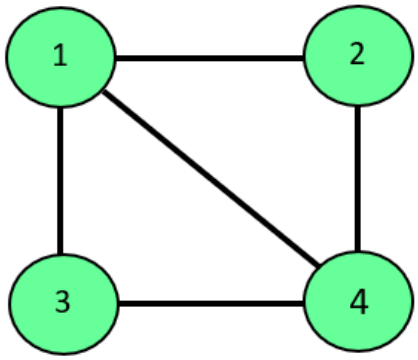
Marks for this submission: 20.00/20.00.

Question **5**

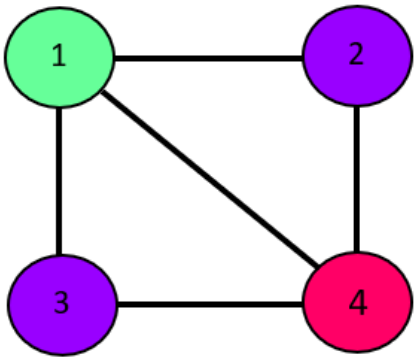
Incorrect

Mark 0.00 out of 20.00

The m-coloring problem states, **"We are given an undirected graph and m number of different colors. We have to check if we can assign colors to the vertices of the graphs in such a way that no two adjacent vertices have the same color."**



0	1	1	1
1	0	0	1
1	0	0	1
1	1	1	0



Node 1 -> color 1
Node 2 -> color 2
Node 3 -> color 2
Node 4-> color 3

For example:

Result
Solution Exists: Following are the assigned colors Vertex 1 is given color: 1 Vertex 2 is given color: 2 Vertex 3 is given color: 3 Vertex 4 is given color: 2

Answer: (penalty regime: 0 %)

Reset answer

```
def isSafe(graph, color):
    for i in range(4):
        for j in range(i + 1, 4):
            if (graph[i][j] and color[j] == color[i]):
                return False
    return True

def graphColoring(graph, m, i, color):

    ############ Add your code here ####################
def display(color):
    print("Solution Exists:" " Following are the assigned colors ")
    for i in range(4):
        print("Vertex", i+1 ," is given color: ",color[i])
if __name__ == '__main__':
    graph = [
        [ 0, 1, 1, 1 ],
        [ 1, 0, 1, 0 ],
        [ 1, 1, 0, 1 ],
        [ 1, 0, 1, 0 ],
    ]
    m = 3 # Number of colors
```

Syntax Error(s)

```
Sorry: IndentationError: expected an indented block (__tester__.python3, line 11)
```

Incorrect

Marks for this submission: 0.00/20.00.