Started on	Wednesday, 2 April 2025, 2:13 PM
State	Finished
Completed on	Wednesday, 2 April 2025, 2:54 PM
Time taken	40 mins 47 secs
Grade	**100.00** 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 partition(arr, low, high):

    pivot = arr[high]
    print(f"pivot:  {pivot}")
    i = low - 1

    for j in range(low, high):
        if arr[j] < pivot:
            i += 1
            swap(arr, i, j)

    swap(arr, i + 1, high)
    return i + 1

def swap(arr, i, j):
    arr[i], arr[j] = arr[j], arr[i]

def quickSort(arr, low, high):
    if low < high:

        pi = partition(arr, low, high)

```

	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]	✔

	Input	Expected	Got	
✔	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]	✔
✔	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.

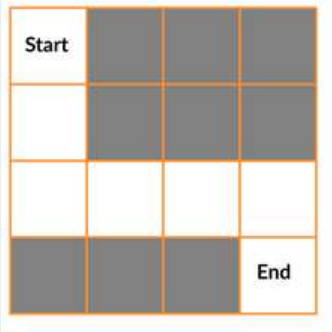
Question **2**

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
    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
```

	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.

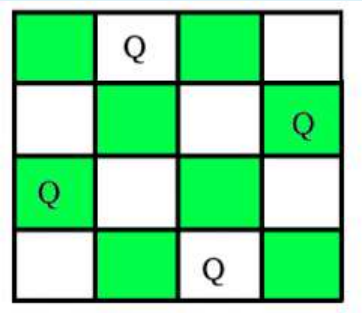
Question **3**

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 4

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
4	0 0 1 0 1 0 0 0 0 0 0 1 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	
✔	4	0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0	0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0	✔
✔	2	Solution does not exist	Solution does not exist	✔

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

We are given a list of n numbers and a number x, the task is to write a python program to find out all possible subsets of the list such that their sum is x.

Examples:

Input: *arr = [2, 4, 5, 9], x = 15*

Output: *[2, 4, 9]*

15 can be obtained by adding 2, 4 and 9 from the given list.

Input : *arr = [10, 20, 25, 50, 70, 90], x = 80*

Output : *[10, 70]*

[10, 20, 50]

80 can be obtained by adding 10 and 70 or by adding 10, 20 and 50 from the given list.

THE INPUT

1.No of numbers

2.Get the numbers

3.Sum Value

For example:

Input	Result
4 2 4 5 9 15	[2, 4, 9]
5 4 16 5 23 12 9	[4, 5]

Answer: (penalty regime: 0 %)

Reset answer

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


```

```
n=int(input())
arr=[]
for i in range(0,n):
    a=int(input())
    arr.append(a)
x = int(input())

subsetSum(n, arr, x)
```

	Input	Expected	Got	
✔	4 2 4 5 9 15	[2, 4, 9]	[2, 4, 9]	✔
✔	6 10 20 25 50 70 90 80	[10, 70] [10, 20, 50]	[10, 70] [10, 20, 50]	✔
✔	5 4 16 5 23 12 9	[4, 5]	[4, 5]	✔

Passed all tests! ✔

Correct

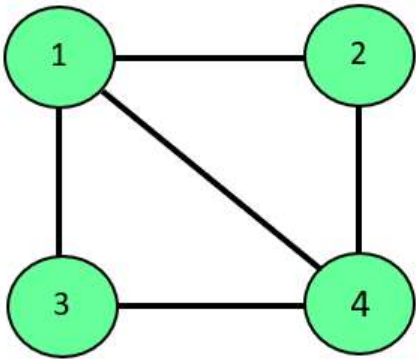
Marks for this submission: 20.00/20.00.

Question **5**

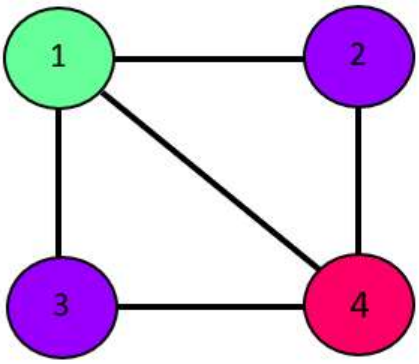
Correct

Mark 20.00 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, i, c):
    # Check all adjacent vertices to see if any have the same color
    for j in range(4):
        if graph[i][j] == 1 and color[j] == c:  # There's an edge and color is same
            return False
    return True

def graphColoring(graph, m, i, color):
    # If all vertices are assigned a color, return True
    if i == 4:
        return True

    # Try different colors for vertex i
    for c in range(1, m + 1):
        if isSafe(graph, color, i, c):
            color[i] = c  # Assign color c to vertex i
```

```
            if graphColoring(graph, m, i + 1, color):  # Recursively assign colors to the next vertic
                return True
            color[i] = 0  # Backtrack if no solution is found

    return False  # If no color can be assigned to vertex i, return False
```

	Expected	Got	
✔	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	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	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.