Started on	Tuesday, 15 July 2025, 1:36 PM
State	Finished
Completed on	Tuesday, 15 July 2025, 6:45 PM
Time taken	5 hours 8 mins
Overdue	3 hours 8 mins
Grade	**80.00** out of 100.00

Question **1**

Incorrect

Mark 0.00 out of 20.00

Write a Python Program to calculate the GCD of the given two numbers using Recursive function

For example:

Input	Result
49 35	7
25 90	5

Answer: (penalty regime: 0 %)

```
from itertools import combinations

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

inputs = list(map(int, input().split()))
arr = inputs[:-1]
x = inputs[-1]

subsetSum(arr, x)
```

	Input	Expected	
✖	49 35	7	✖

Some hidden test cases failed, too.

Your code must pass all tests to earn any marks. Try again.

Incorrect

Marks for this submission: 0.00/20.00.

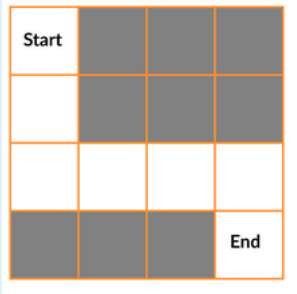
Question **2**

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.

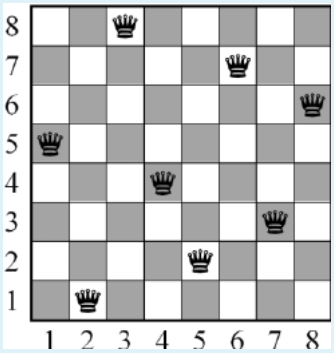
Question **3**

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

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

Greedy coloring doesn't always use the minimum number of colors possible to color a graph. For a graph of maximum degree x, greedy coloring will use at most x+1 color. Greedy coloring can be arbitrarily bad;

Create a python program to implement graph colouring using Greedy algorithm.

For example:

Test	Result
colorGraph(graph, n)	Color assigned to vertex 0 is BLUE Color assigned to vertex 1 is GREEN Color assigned to vertex 2 is BLUE Color assigned to vertex 3 is RED Color assigned to vertex 4 is RED Color assigned to vertex 5 is GREEN

Answer: (penalty regime: 0 %)

Reset answer

```
class Graph:
    def __init__(self, edges, n):
        self.adjList = [[] for _ in range(n)]

        # add edges to the undirected graph
        for (src, dest) in edges:
            self.adjList[src].append(dest)
            self.adjList[dest].append(src)
def colorGraph(graph, n):
    result = [-1]*n
    available = [False]*n
    result[0] = 1
    for u in range(1, n):
        for v in graph.adjList[u]:
            if result[v] != -1:
                available[result[v]] = True
        color = 1
        while color <= n:
            if not available[color]:
                break
            color += 1
        result[u] = color
```

	Test	Expected	Got	
✔	colorGraph(graph, n)	Color assigned to vertex 0 is BLUE Color assigned to vertex 1 is GREEN Color assigned to vertex 2 is BLUE Color assigned to vertex 3 is RED Color assigned to vertex 4 is RED Color assigned to vertex 5 is GREEN	Color assigned to vertex 0 is BLUE Color assigned to vertex 1 is GREEN Color assigned to vertex 2 is BLUE Color assigned to vertex 3 is RED Color assigned to vertex 4 is RED Color assigned to vertex 5 is GREEN	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.