Started on	Friday, 14 February 2025, 9:16 AM
State	Finished
Completed on	Friday, 14 February 2025, 9:45 AM
Time taken	28 mins 45 secs
Grade	**80.00** out of 100.00

Question **1**

Not answered

Mark 0.00 out of 20.00

Write a python program to implement merge sort without using recursive function on the given list of float values.

For example:

Input	Result
5 6.2 4.1 3.2 5.6 7.4	left: [6.2] Right: [4.1] left: [3.2] Right: [5.6] left: [7.4] Right: [] left: [4.1, 6.2] Right: [3.2, 5.6] left: [7.4] Right: [] left: [3.2, 4.1, 5.6, 6.2] Right: [7.4] [3.2, 4.1, 5.6, 6.2, 7.4]
6 3.2 8.9 4.5 6.2 1.5 8.0	left: [3.2] Right: [8.9] left: [4.5] Right: [6.2] left: [1.5] Right: [8.0] left: [3.2, 8.9] Right: [4.5, 6.2] left: [1.5, 8.0] Right: [] left: [3.2, 4.5, 6.2, 8.9] Right: [1.5, 8.0] [1.5, 3.2, 4.5, 6.2, 8.0, 8.9]

Answer: (penalty regime: 0 %)

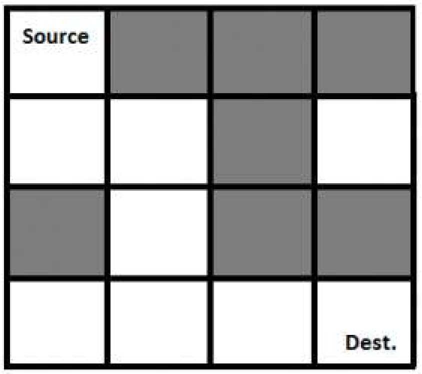
Question **2**

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

```python
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");
```

	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.

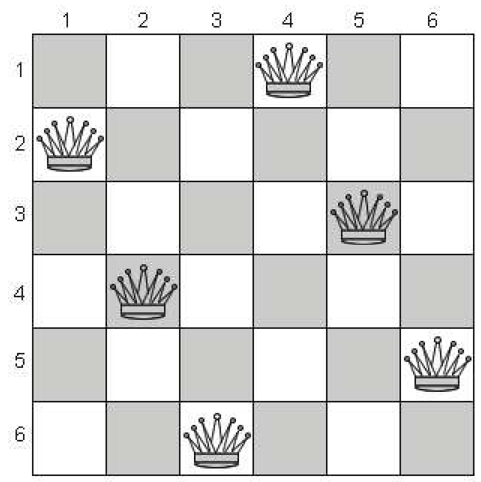
Question **3**

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 6

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

6	0 0 0 1 0 0
	1 0 0 0 0 0
	0 0 0 0 1 0
	0 1 0 0 0 0
	0 0 0 0 0 1
	0 0 1 0 0 0

Answer: (penalty regime: 0 %)

Reset answer

```
                if solveNQUtil(board,col+1):
                    return True
                board[i][col]=0
        return False
def solveNQ():
    board = [ [0, 0, 0, 0, 0, 0, 0, 0],
              [0, 0, 0, 0, 0, 0, 0, 0],
              [0, 0, 0, 0, 0, 0, 0, 0],
              [0, 0, 0, 0, 0, 0, 0, 0],
              [0, 0, 0, 0, 0, 0, 0, 0],
              [0, 0, 0, 0, 0, 0, 0, 0],
              [0, 0, 0, 0, 0, 0, 0, 0],
              [0, 0, 0, 0, 0, 0, 0, 0]]

    if solveNQUtil(board, 0) == False:
        print ("Solution does not exist")
        return False

    printSolution(board)
    return True
solveNQ()

```

	Input	Expected	Got	
✓	2	Solution does not exist	Solution does not exist	✓
✓	3	Solution does not exist	Solution does not exist	✓
✓	6	0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0	0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 1 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

Given a set of positive integers, and a value sum, determine that the sum of the subset of a given set is equal to the given sum.

Write the program for subset sum problem.

INPUT

1.no of elements

2.Input the given elements

3.Get the target sum

OUTPUT

True , if subset with required sum is found

False , if subset with required sum is not found

For example:

Input	Result
5 4 16 5 23 12 9	4 16 5 23 12 True,subset found

Answer: (penalty regime: 0 %)

Reset answer

```
    if SubsetSum(a, i + 1, sum, target, n):
        return True

    return False

a=[]
size=int(input())
for i in range(size):
    x=int(input())
    a.append(x)

target=int(input())
n=len(a)
if(SubsetSum(a,0,0,target,n)==True):
    for i in range(size):
        print(a[i])
    print("True,subset found")
else:
    for i in range(size):
        print(a[i])
    print("False,subset not found")

```

	Input	Expected	Got	
✓	5 4 16 5 23 12 9	4 16 5 23 12 True,subset found	4 16 5 23 12 True,subset found	✓
✓	4 1 2 3 4 11	1 2 3 4 False,subset not found	1 2 3 4 False,subset not found	✓
✓	7 10 7 5 18 12 20 15 35	10 7 5 18 12 20 15 True,subset found	10 7 5 18 12 20 15 True,subset found	✓

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 %)

```

    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):
                colour[v] = c
                if self.graphColourUtil(m, colour, v + 1):
                    return True
                colour[v] = 0
```

```python

    def graphColouring(self, m):
        colour = [0] * self.V
        if not self.graphColourUtil(m, colour, 0):
            print("Solution does not exist")
            return False

        print("Solution exist and Following are the assigned colours:")
        for c in colour:
            print(c, end=' ')
        return True
```

	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.