Started on	Friday, 25 April 2025, 8:46 AM
State	Finished
Completed on	Friday, 25 April 2025, 9:20 AM
Time taken	34 mins 13 secs
Grade	**100.00** out of 100.00

Question **1**

Correct

Mark 20.00 out of 20.00

Write a recursive python function to perform merge sort on the unsorted list of float values.

For example:

Test	Input	Result
mergesort(li)	5 3.2 1.5 1.6 1.7 8.9	[1.5, 1.6, 1.7, 3.2, 8.9]
mergesort(li)	6 3.1 2.3 6.5 4.5 7.8 9.2	[2.3, 3.1, 4.5, 6.5, 7.8, 9.2]

Answer: (penalty regime: 0 %)

Ace editor not ready. Perhaps reload page?
Falling back to raw text area.

```
def mergesort(li):
    if len(li) > 1:
        mid = len(li) // 2
        L = li[:mid]
        R = li[mid:]
        mergesort(L)
        mergesort(R)
        i = j = k = 0
        while i < len(L) and j < len(R):
            if L[i] < R[j]:
                li[k] = L[i]
                i += 1
            else:
                li[k] = R[j]
                j += 1
            k += 1
```

	Test	Input	Expected	Got	
✔	mergesort(li)	5 3.2 1.5 1.6 1.7 8.9	[1.5, 1.6, 1.7, 3.2, 8.9]	[1.5, 1.6, 1.7, 3.2, 8.9]	✔

	Test	Input	Expected	Got	
✔	mergesort(li)	6 3.1 2.3 6.5 4.5 7.8 9.2	[2.3, 3.1, 4.5, 6.5, 7.8, 9.2]	[2.3, 3.1, 4.5, 6.5, 7.8, 9.2]	✔
✔	mergesort(li)	4 3.1 2.3 6.5 4.1	[2.3, 3.1, 4.1, 6.5]	[2.3, 3.1, 4.1, 6.5]	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

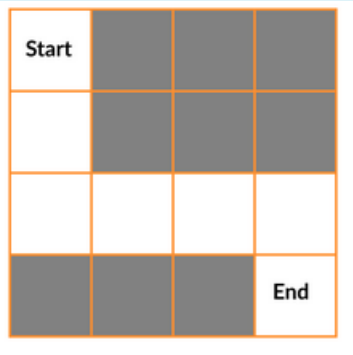
Question **2**

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 respectiv
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
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 ca
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 c
indicating the absence of the path to destination.**

Answer: (penalty regime: 0 %)

Reset answer

Ace editor not ready. Perhaps reload page?
Falling back to raw text area.

```
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):
```

	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.

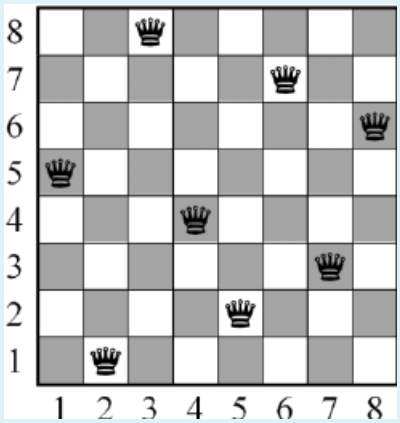
Question **3**

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 othe 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 pri 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

Ace editor not ready. Perhaps reload page?
Falling back to raw text area.

```
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)):
```

	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

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

Ace editor not ready. Perhaps reload page?
Falling back to raw text area.

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

	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

Greedy coloring doesn't always use the minimum number of colors possible to color a graph. For a graph of maximum deg
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

Ace editor not ready. Perhaps reload page?
Falling back to raw text area.

```
class Graph:
    def __init__(fish, edges, n):
        fish.adjList = [[] for _ in range(n)]
        for (src, dest) in edges:
            fish.adjList[src].append(dest)
            fish.adjList[dest].append(src)
def colorGraph(graph, n):
    print('''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''')
if __name__ == '__main__':
    colors = ['', 'BLUE', 'GREEN', 'RED', 'YELLOW', 'ORANGE', 'PINK',
              'BLACK', 'BROWN', 'WHITE', 'PURPLE', 'VOILET']
    edges = [(0, 1), (0, 4), (0, 5), (4, 5), (1, 4), (1, 3), (2, 3), (2, 4)]
    n = 6
```

	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.