

Started on	Wednesday, 26 March 2025, 9:16 AM
State	Finished
Completed on	Wednesday, 26 March 2025, 9:57 AM
Time taken	41 mins 36 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 values.

For example:

Input	Result
7	left: [33]
33	Right: [42]
42	left: [9]
9	Right: [37]
37	left: [8]
8	Right: [47]
47	left: [5]
5	Right: []
	left: [33, 42]
	Right: [9, 37]
	left: [8, 47]
	Right: [5]
	left: [9, 33, 37, 42]
	Right: [5, 8, 47]
	[5, 8, 9, 33, 37, 42, 47]
6	left: [10]
10	Right: [3]
3	left: [5]
5	Right: [61]
61	left: [74]
74	Right: [92]
92	left: [3, 10]
	Right: [5, 61]
	left: [74, 92]
	Right: []
	left: [3, 5, 10, 61]
	Right: [74, 92]
	[3, 5, 10, 61, 74, 92]

Answer: (penalty regime: 0 %)

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

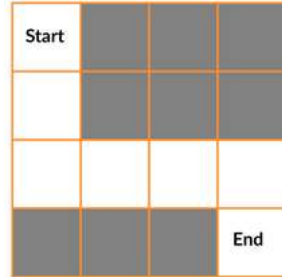
Question **2**

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 $n \times n$. 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

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 ):
    # 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
```

	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.

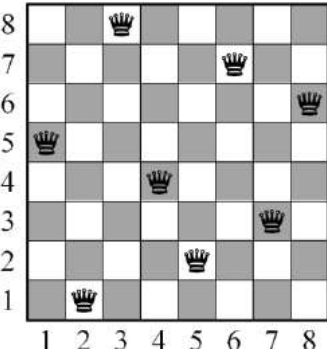
Question 3

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

```
1 global N
2 N = int(input())
3
4 def printSolution(board):
5     for i in range(N):
6         for j in range(N):
7             print(board[i][j], end = " ")
8         print()
9
10 def isSafe(board, row, col):
11
12     # Check this row on left side
13     for i in range(col):
14         if board[row][i] == 1:
15             return False
16
17     # Check upper diagonal on left side
18     for i, j in zip(range(row, -1, -1),
19                     range(col, -1, -1)):
20         if board[i][j] == 1:
21             return False
22
```

	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**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

```

1 def subsetSum(arr, n, i, sum, count):
2     if(i==n):
3         if(sum==0):
4             count+=1
5         return count
6     count=subsetSum(arr,n,i+1,sum-arr[i],count)
7     count=subsetSum(arr,n,i+1,sum,count)
8     return count
9
10
11
12
13
14
15 arr=[]
16 size=int(input())
17 for j in range(size):
18     value=int(input())
19     arr.append(value)
20 sum = int(input())

```

```
21 | n = len(arr)
22 |
```

	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**

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 i th 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 exists.

Answer: (penalty regime: 0 %)

```
1 class Graph():
2     def __init__(self,vertices):
3         self.V=vertices
4         self.graph=[[0 for column in range(vertices)]for row in range(vertices)]
5     def isSafe(self,v,colour,c):
6         for i in range(self.V):
7             if self.graph[v][i]==1 and colour[i]==c:
8                 return False
9         return True
10    def graphColourUtil(self,m,colour,v):
11        if v==self.V:
12            return True
13        for c in range(1,m+1):
14            if self.isSafe(v,colour,c)==True:
15                colour[v]=c
16                if self.graphColourUtil(m,colour,v+1)==True:
17                    return True
18                colour[v]=0
19    def graphColouring(self, m):
20        colour = [0] * self.V
21        if self.graphColourUtil(m, colour, 0) == None:
```

22

return False



	Test	Expected	Got	
✓	<pre>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)</pre>	<p>Solution exist and Following are the assigned colours:</p> <p>1 2 3 2</p>	<p>Solution exist and Following are the assigned colours:</p> <p>1 2 3 2</p>	✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.