Started on	Monday, 5 May 2025, 9:12 AM
State	Finished
Completed on	Tuesday, 6 May 2025, 8:31 AM
Time taken	23 hours 18 mins
Overdue	21 hours 18 mins
Grade	**80.00** out of 100.00

Question **1**

Correct

Mark 20.00 out of 20.00

You are the king of Pensville where you have $2N2N$ workers.
All workers will be grouped in association of size *2*,so a total of *N* associations have to be formed.
The building speed of the $i^{th}i^{th}$ worker is A_iA_i.
To make an association, you pick up *2* workers. Let the minimum building speed between both workers be *x*, then the association has the resultant building speed *x*.
You have to print the maximum value possible of the sum of building speeds of *N* associations if you make the associations optimally.

Input

First line contains an integer *N*, representing the number of associations to be made.
Next line contains $2N2N$ space separated integers, denoting the building speeds of $2N2N$ workers.

Output

Print the maximum value possible of the sum of building speeds of all the associations.

Sample Input

```
2
1 3 1 2
```

Sample Output

```
3
```

For example:

Input	Result
2 1 3 1 2	3

Answer: (penalty regime: 0 %)

```
def max_sum_of_min_speeds(N, speeds):

    speeds.sort()

    max_sum = 0

    for i in range(0, 2 * N, 2):
        max_sum += speeds[i]

    return max_sum

N = 2
speeds = [1, 3, 1, 2]

print(max_sum_of_min_speeds(N, speeds))
```

	Input	Expected	Got	
✓	2 1 3 1 2	3	3	✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.

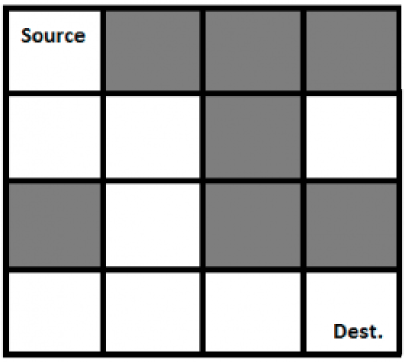
Question **2**

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) ]
```

	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.

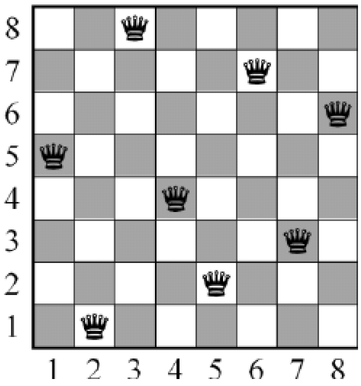
Question **3**

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	✓
✓	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

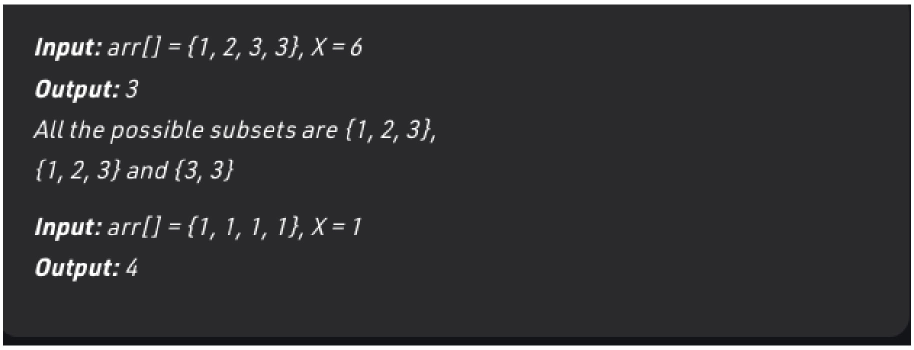
Mark 20.00 out of 20.00

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

```
def subsetSum(arr, n, i,sum, count):
#Write your code here
    if i == n:
        if sum == 0:
            return count + 1
        return count

    # Include the current element in the subset
    count = subsetSum(arr, n, i + 1, sum - arr[i], count)

    # Exclude the current element from the subset
    count = subsetSum(arr, n, i + 1, sum, count)

    return count

arr=[]
size=int(input())
for j in range(size):
    value=int(input())
    value = int(input())
    arr.append(value)
sum = int(input())
n = len(arr)
```

	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**

Not answered

Mark 0.00 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

```python
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):
    ############ Add your code here ##############
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
    graph = Graph(edges, n)
    colorGraph(graph, n)
```