
Started on Saturday, 8 February 2025, 11:10 AM

State Finished

Completed on Saturday, 8 February 2025, 11:14 AM

Time taken 4 mins 6 secs

Grade **80.00** out of 100.00

Question **1**

Correct

Mark 20.00 out of 20.00

Write a python program to implement quick sort on the given array values.

For example:

Input	Result
5	left: []
21	right: []
40	left: []
50	right: []
30	left: []
13	right: []
	left: [30]
	right: [50]
	left: [13]
	right: [30, 40, 50]
	[13, 21, 30, 40, 50]
6	left: []
7	right: []
5	left: [4]
21	right: []
63	left: []
4	right: []
9	left: []
	right: []
	left: [9]
	right: [63]
	left: [4, 5]
	right: [9, 21, 63]
	[4, 5, 7, 9, 21, 63]

Answer: (penalty regime: 0 %)

```

1 def quick(l):
2     if l==[]:
3         return []
4     pivot=l[0:1]
5     left=quick([x for x in l[1:] if x<l[0]])
6     right=quick([x for x in l[1:] if x>=l[0]])
7     print("left: ",left)
8     print("right: ",right)
9     return left+pivot+right
10 a=[]
11 n=int(input())
12 for i in range(n):
13     a.append(int(input()))
14 print(quick(a))

```

Input	Expected	Got	
-------	----------	-----	--

	Input	Expected	Got	
✓	5 21 40 50 30 13	left: [] right: [] left: [] right: [] left: [] right: [] left: [30] right: [50] left: [13] right: [30, 40, 50] [13, 21, 30, 40, 50]	left: [] right: [] left: [] right: [] left: [] right: [] left: [30] right: [50] left: [13] right: [30, 40, 50] [13, 21, 30, 40, 50]	✓
✓	6 7 5 21 63 4 9	left: [] right: [] left: [4] right: [] left: [] right: [] left: [] right: [] left: [9] right: [63] left: [4, 5] right: [9, 21, 63] [4, 5, 7, 9, 21, 63]	left: [] right: [] left: [4] right: [] left: [] right: [] left: [] right: [] left: [9] right: [63] left: [4, 5] right: [9, 21, 63] [4, 5, 7, 9, 21, 63]	✓
✓	4 20 35 60 12	left: [] right: [] left: [] right: [] left: [] right: [60] left: [12] right: [35, 60] [12, 20, 35, 60]	left: [] right: [] left: [] right: [] left: [] right: [60] left: [12] right: [35, 60] [12, 20, 35, 60]	✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.

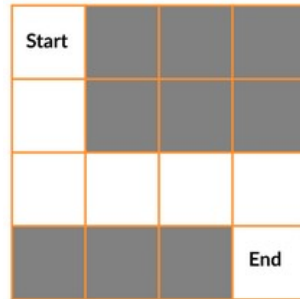
Question **2**

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 $n \times n$. 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

```

1 N = 4
2
3 def printSolution( sol ):
4
5     for i in sol:
6         for j in i:
7             print(str(j) + " ", end = "")
8             print("")
9
10
11 def isSafe( maze, x, y ):
12
13     if x >= 0 and x < N and y >= 0 and y < N and maze[x][y] == 1:
14         return True
15
16     return False
17
18
19 def solveMaze( maze ):
20
21     # Creating a 4 * 4 2-D list
22     sol = [ [ 0 for j in range(4) ] for i in range(4) ]

```

Expected	Got

	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.

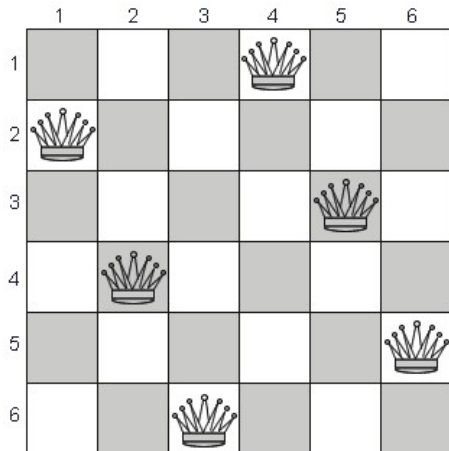
Question 3

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

```

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), range(col, -1, -1)):
19         if board[i][j] == 1:
20             return False

```

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

Answer: (penalty regime: 0 %)

Reset answer

```
1 def SubsetSum(a,i,sum,target,n):
2
3 # Write your code here
4     if i == n:
5         return sum == target
6
7     # Include the current element in the subset
8     if SubsetSum(a, i + 1, sum + a[i], target, n):
9         return True
10
11    # Exclude the current element from the subset
12    if SubsetSum(a, i + 1, sum, target, n):
13        return True
14
15    return False
16
17 a=[]
18 size=int(input())
19 for i in range(size):
20     x=int(input())
21     a.append(x)
22
```

Input	Expected	Got
-------	----------	-----

	Input	Expected	Got	
✓	5 4 16 5 23 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

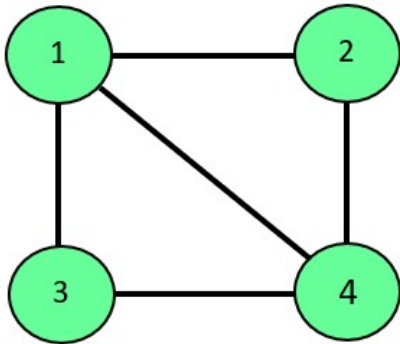
Marks for this submission: 20.00/20.00.

Question 5

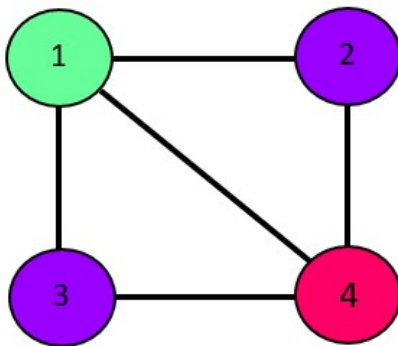
Not answered

Mark 0.00 out of 20.00

The m-coloring problem states, "We are given an undirected graph and m number of different colors. We have to check if we can assign colors to the vertices of the graphs in such a way that no two adjacent vertices have the same color."



0	1	1	1
1	0	0	1
1	0	0	1
1	1	1	0



Node 1 -> color 1

Node 2 -> color 2

Node 3 -> color 2

Node 4 -> color 3

For example:

Result

Solution Exists: Following are the assigned colors
 Vertex 1 is given color: 1
 Vertex 2 is given color: 2
 Vertex 3 is given color: 3
 Vertex 4 is given color: 2

Answer: (penalty regime: 0 %)

Reset answer

```

1 def isSafe(graph, color):
2     for i in range(4):
3         for j in range(i + 1, 4):
4             if (graph[i][j] and color[j] == color[i]):
5                 return False
6     return True
7
8 def graphColoring(graph, m, i, color):
9
10    ##### Add your code here #####
11 def display(color):
12     print("Solution Exists:" " Following are the assigned colors ")
13     for i in range(4):
14         print("Vertex", i+1, " is given color: ",color[i])
15 if __name__ == '__main__':
16     graph = [
17         [ 0, 1, 1, 1 ],
18         [ 1, 0, 0, 1 ],
19         [ 1, 0, 0, 1 ],
20         [ 1, 1, 1, 0 ]

```

Syntax Error(s)

Sorry: IndentationError: expected an indented block (__tester__.python3, line 11)

Incorrect

Marks for this submission: 0.00/20.00.