Started on	Friday, 20 September 2024, 9:23 AM
State	Finished
Completed on	Friday, 20 September 2024, 9:46 AM
Time taken	22 mins 18 secs
Grade	**80.00** out of 100.00

Question **1**

Correct

Mark 20.00 out of 20.00

Flag question

Write a python program to calculate the length of the given string using recursion

For example:

Test	Input	Result
length(str)	saveetha	length of saveetha is 8
length(str)	engineering	length of engineering is 11

Answer: (penalty regime: 0 %)

```
1 str=input()
2 def length(str):
3     print("length of",str,"is",len(str))
```

	Test	Input	Expected	Got	
	length(str)	saveetha	length of saveetha is 8	length of saveetha is 8	
	length(str)	engineering	length of engineering is 11	length of engineering is 11	
	length(str)	Welcome	length of Welcome is 7	length of Welcome is 7	

Passed all tests!

Correct

Marks for this submission: 20.00/20.00.

Question **2**

Correct

Mark 20.00 out of 20.00

Flag question

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

	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

Flag question

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 que 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 on 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

Flag question

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

```
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())
```

	Input	Expected	Got	
	5	4	4	

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

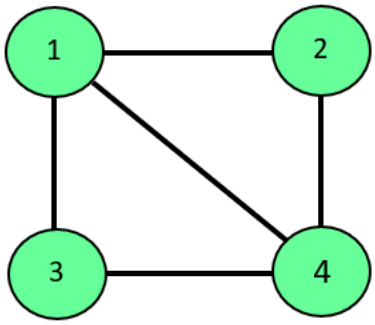
Question **5**

Incorrect

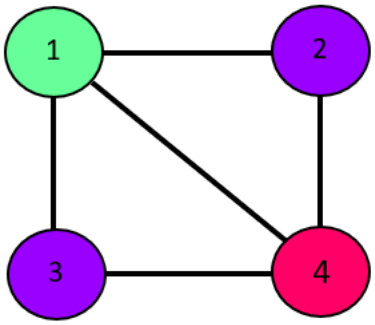
Mark 0.00 out of 20.00

Flag question

The m-coloring problem states, **"We are given an undirected graph and m number of different colors. We have to check if we ca 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

```
        for i in range(self.V):
            if self.graph[v][i]==1 and colour[i]==c:
                return False
        return True
    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)== True:
                colour[v]=c
```

```
                if self.graphColourUtil(m,colour,v+1)==True:
                    return True
                colour[v]=0
    def graphColouring(self,m):
        colour=[0]*self.V
        if self.graphColourUtil(m,colour,0)==None:
            return False
        print("Solution exist and Following are the assigned colours:")
        for c in colour:
            print(c,end=' ')
        return True
```

	Expected
	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

Your code must pass all tests to earn any marks. Try again.

Incorrect

Marks for this submission: 0.00/20.00.

Finish r