Started on	Tuesday, 20 May 2025, 8:11 AM
State	Finished
Completed on	Tuesday, 20 May 2025, 8:47 AM
Time taken	36 mins 41 secs
Grade	**100.00** out of 100.00

Question **1**

Correct

Mark 20.00 out of 20.00

Create a python program to compute the edit distance between two given strings using iterative method.

For example:

Input	Result
kitten sitting	3

Answer: (penalty regime: 0 %)

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

```
def LD(s, t):
    #########  Add your code here ###########
    if s == "":
        return len(t)
    if t == "":
        return len(s)
    if s[-1] == t[-1]:
        cost = 0
    else:
        cost = 1
    res = min([LD(s[:-1], t)+1, LD(s, t[:-1])+1, LD(s[:-1], t[:-1]) + cost])
    return res
str1=input()
str2=input()
print(LD(str1,str2))
```

	Input	Expected	Got	
✓	kitten sitting	3	3	✓
✓	medium median	2	2	✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.

Question **2**

Correct

Mark 20.00 out of 20.00

LONGEST COMMON SUBSTRING PROBLEM

Given two strings 'X' and 'Y', find the length of the longest common substring.

Answer: (penalty regime: 0 %)

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

```
def LCS(X, Y, m, n):
    maxLength = 0
    endingIndex = m
    lookup = [[0 for x in range(n + 1)] for y in range(m + 1)]
    for i in range(1, m + 1):
        for j in range(1, n + 1):
            if X[i - 1] == Y[j - 1]:
                lookup[i][j] = lookup[i - 1][j - 1] + 1
                if lookup[i][j] > maxLength:
                    maxLength = lookup[i][j]
                    endingIndex = i
    return len(X[endingIndex - maxLength: endingIndex])

X = input()
Y = input()
m = len(X)
n = len(Y)
print('Length of Longest Common Substring is', LCS(X, Y, m, n))
```

	Input	Expected	Got	
✓	ABC BABA	Length of Longest Common Substring is 2	Length of Longest Common Substring is 2	✓
✓	abcdxyz xyzabcd	Length of Longest Common Substring is 4	Length of Longest Common Substring is 4	✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.

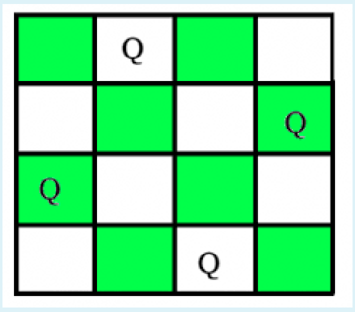
Question **3**

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 4

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

4	0 0 1 0
	1 0 0 0
	0 0 0 1
	0 1 0 0

Answer: (penalty regime: 0 %)

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):

    # 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),
```

	Input	Expected	Got	
✓	4	0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0	0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0	✓

	Input	Expected	Got	
✓	2	Solution does not exist	Solution does not exist	✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.

Question **4**

Correct

Mark 20.00 out of 20.00

Create a python program to find the longest common subsequence using Memoization Implementation.

For example:

Input	Result
AGGTAB GXTXAYB	Length of LCS is 4

Answer: (penalty regime: 0 %)

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

```
def lcs(u, v):
    """Return c where c[i][j] contains length of LCS of u[i:] and v[j:]."""
    c = [[-1]*(len(v) + 1) for _ in range(len(u) + 1)]
    for i in range(len(u) + 1):
        c[i][len(v)] = 0
    for j in range(len(v)):
        c[len(u)][j] = 0

    for i in range(len(u) - 1, -1, -1):
        for j in range(len(v) - 1, -1, -1):
            if u[i] == v[j]:
                c[i][j] = 1 + c[i + 1][j + 1]
            else:
                c[i][j] = max(c[i + 1][j], c[i][j + 1])
    return c

def print_lcs(u, v, c):
    """Print one LCS of u and v using table c."""
```

	Input	Expected	Got	
✓	AGGTAB GXTXAYB	Length of LCS is 4	Length of LCS is 4	✓
✓	SAMPLE SAEMSUNG	Length of LCS is 3	Length of LCS is 3	✓
✓	saveetha sabeetha	Length of LCS is 7	Length of LCS is 7	✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.

Question **5**

Correct

Mark 20.00 out of 20.00

LONGEST PALINDROMIC SUBSEQUENCE

Given a sequence, find the length of the longest palindromic subsequence in it.

For example:

Input	Result
ABBDCACB	The length of the LPS is 5

Answer: (penalty regime: 0 %)

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

```
dp = [[-1 for i in range(1001)]for j in range(1001)]
def lps(s1, s2, n1, n2):
    if (n1 == 0 or n2 == 0):
        return 0
    if (dp[n1][n2] != -1):
        return dp[n1][n2]
    if (s1[n1 - 1] == s2[n2 - 1]):
        dp[n1][n2] = 1 + lps(s1, s2, n1 - 1, n2 - 1)
        return dp[n1][n2]
    else:
        dp[n1][n2] = max(lps(s1, s2, n1 - 1, n2), lps(s1, s2, n1, n2 - 1))
        return dp[n1][n2]
seq = input()
n = len(seq)
s2 = seq
s2 = s2[::-1]
print(f"The length of the LPS is",lps(s2, seq, n, n))
```

	Input	Expected	Got	
✓	ABBDCACB	The length of the LPS is 5	The length of the LPS is 5	✓
✓	BBABCBCAB	The length of the LPS is 7	The length of the LPS is 7	✓
✓	cbbd	The length of the LPS is 2	The length of the LPS is 2	✓
✓	abbab	The length of the LPS is 4	The length of the LPS is 4	✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.