Started on	Friday, 11 April 2025, 10:53 AM
State	Finished
Completed on	Wednesday, 23 April 2025, 12:41 PM
Time taken	12 days 1 hour
Overdue	11 days 23 hours
Grade	**80.00** out of 100.00

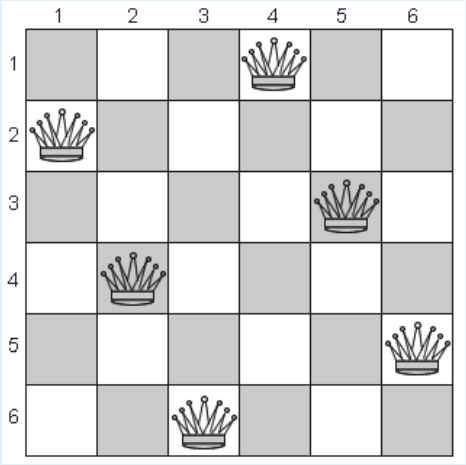
Question **1**

Not answered

Mark 0.00 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 %)

```
1
```

	Input	Expected	Got	

Testing was aborted due to error.
Your code must pass all tests to earn any marks. Try again.

Show differences

Incorrect

Marks for this submission: 0.00/20.00.

Question **2**

Correct

Mark 20.00 out of 20.00

Create a Python program to find longest common substring or subword (LCW) of two strings using dynamic programming with bottom-up approach.

A string r is a substring or subword of a string s if r is contained within s. A string r is a common substring of s and t if r is a substring of both s and t. A string r is a longest common substring or subword (LCW) of s and t if there is no string that is longer than r and is a common substring of s and t. The problem is to find an LCW of two given strings.

For example:

Test	Input	Result
lcw(u, v)	bisect trisect	Longest Common Subword: isect

Answer: (penalty regime: 0 %)

Reset answer

```
def lcw(X,Y):
    m = len(X)
    n = len(Y)
    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 X[endingIndex - maxLength: endingIndex]

u = input()
v = input()
print("Longest Common Subword:", lcw(u,v))
```

	Test	Input	Expected	Got	
✔	lcw(u, v)	bisect trisect	Longest Common Subword: isect	Longest Common Subword: isect	✔
✔	lcw(u, v)	director conductor	Longest Common Subword: ctor	Longest Common Subword: ctor	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **3**

Correct

Mark 20.00 out of 20.00

Given a string s, return *the longest palindromic substring* in s.

Example 1:

```
Input: s = "babad"
Output: "bab"
Explanation: "aba" is also a valid answer.
```

Example 2:

```
Input: s = "cbbd"
Output: "bb"
```

For example:

Test	Input	Result
ob1.longestPalindrome(str1)	ABCBCB	BCBCB

Answer: (penalty regime: 0 %)

Reset answer

```
class Solution(object):
    def longestPalindrome(self, s):
        dp = [[False for i in range(len(s))] for i in range(len(s))]
        for i in range(len(s)):
            dp[i][i] = True
        max_length = 1
        start = 0
        for l in range(2,len(s)+1):
            for i in range(len(s)-l+1):
                end = i+l
                if l==2:
                    if s[i] == s[end-1]:
                        dp[i][end-1]=True
                        max_length = l
                        start = i
                else:
                    if s[i] == s[end-1] and dp[i+1][end-2]:
                        dp[i][end-1]=True
                        max_length = l
                        start = i
        return s[start:start+max_length]
ob1 = Solution()
```

	Test	Input	Expected	Got	
✔	ob1.longestPalindrome(str1)	ABCBCB	BCBCB	BCBCB	✔
✔	ob1.longestPalindrome(str1)	BABAD	ABA	ABA	✔

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 compute the edit distance between two given strings using iterative method.

For example:

Input	Result
kitten sitting	3

Answer: (penalty regime: 0 %)

```
def LD(s, t):
    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 **5**

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 %)

```
def lcs(X, Y, m, n, dp):
    if (m == 0 or n == 0):
        return 0
    if (dp[m][n] != -1):
        return dp[m][n]
    if X[m - 1] == Y[n - 1]:
        dp[m][n] = 1 + lcs(X, Y, m - 1, n - 1, dp)
        return dp[m][n]
    dp[m][n] = max(lcs(X, Y, m, n - 1, dp),lcs(X, Y, m - 1, n, dp))
    return dp[m][n]
X =input()  #"AGGTAB"
Y =input() #"GXTXAYB"
m = len(X)
n = len(Y)
dp = [[-1 for i in range(n + 1)]for j in range(m + 1)]
print(f"Length of LCS is {lcs(X, Y, m, n, dp)}")
```

	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.