Started on	Wednesday, 14 May 2025, 3:35 PM
State	Finished
Completed on	Wednesday, 14 May 2025, 4:02 PM
Time taken	27 mins 30 secs
Grade	**80.00** out of 100.00

Question **1**

Not answered

Mark 0.00 out of 20.00

GRAPH COLORING PROBLEM

Given an undirected graph and a number m, determine if the graph can be coloured with at most m colours such that no two adjacent vertices of the graph are colored with the same color. Here coloring of a graph means the assignment of colors to all vertices.

Input-Output format:

Input:

1. A 2D array graph[V][V] where V is the number of vertices in graph and graph[V][V] is an adjacency matrix representation of the graph. A value graph[i][j] is 1 if there is a direct edge from i to j, otherwise graph[i][j] is 0.
2. An integer m is the maximum number of colors that can be used.

Output:

An array color[V] that should have numbers from 1 to m. color[i] should represent the color assigned to the ith vertex.

Example:

```
Input:
graph = {0, 1, 1, 1},
        {1, 0, 1, 0},
        {1, 1, 0, 1},
        {1, 0, 1, 0}
Output:
Solution Exists:
Following are the assigned colors
 1  2  3  2
Explanation: By coloring the vertices
with following colors, adjacent
vertices does not have same colors
```

```
Input:
graph = {1, 1, 1, 1},
        {1, 1, 1, 1},
        {1, 1, 1, 1},
        {1, 1, 1, 1}
Output: Solution does not exist.
Explanation: No solution exits.
```

Answer: (penalty regime: 0 %)

```

```

Question **2**

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 %)

```python
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 **3**

Correct

Mark 20.00 out of 20.00

LONGEST COMMON SUBSTRING PROBLEM

The longest common substring problem is the problem of finding the longest string (or strings) that is a substring (or are substrings) of two strings.

Answer: (penalty regime: 0 %)

```python
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 X[endingIndex - maxLength: endingIndex]

X = input()
Y = input()
m = len(X)
n = len(Y)
print('The longest common substring is', LCS(X, Y, m, n))

```

	Input	Expected	Got	
✓	ABC BABA	The longest common substring is AB	The longest common substring is AB	✓
✓	abcdxyz xyzabcd	The longest common substring is abcd	The longest common substring is abcd	✓

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 palindromic substring using optimal algorithm Expand around center.

For example:

Test	Input	Result
`findLongestPalindromicSubstring(s)`	samsunggnusgnusam	sunggnus

Answer: (penalty regime: 0 %)

Reset answer

```python
def printSubStr(ss, low, high):
    for i in range(low, high + 1):
        print(s[i], end = "")
def findLongestPalindromicSubstring(s):
    n = len(s)
    maxLength = 1
    start = 0
    for i in range(n):
        for j in range(i, n):
            flag = 1
            for k in range(0, ((j - i) // 2) + 1):
                if (s[i + k] != s[j - k]):
                    flag = 0
            if (flag != 0 and (j - i + 1) > maxLength):
                start = i
                maxLength = j - i + 1
    printSubStr(s, start, start + maxLength - 1)
s = input()
```

	Test	Input	Expected	Got	
✓	`findLongestPalindromicSubstring(s)`	samsunggnusgnusam	sunggnus	sunggnus	✓
✓	`findLongestPalindromicSubstring(s)`	welcomeindiaaidni	indiaaidni	indiaaidni	✓

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 %)

```python
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.