Started on	Wednesday, 30 April 2025, 11:23 AM
State	Finished
Completed on	Wednesday, 30 April 2025, 11:43 AM
Time taken	20 mins 5 secs
Grade	**100.00** out of 100.00

Question **1**

Correct

Mark 20.00 out of 20.00

Write a Python Program to find longest common subsequence using Dynamic Programming

Answer: (penalty regime: 0 %)

```
def longest_common_subsequence(X, Y):
    m = len(X)
    n = len(Y)

    dp = [[0] * (n + 1) for _ 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]:
                dp[i][j] = dp[i - 1][j - 1] + 1
            else:
                dp[i][j] = max(dp[i - 1][j], dp[i][j - 1])

    lcs_length = dp[m][n]
    lcs = [''] * lcs_length
    i, j = m, n

    while i > 0 and j > 0:
        if X[i - 1] == Y[j - 1]:
            lcs[lcs_length - 1] = X[i - 1]
            i -= 1
            j -= 1
```

	Input	Expected	Got	
✔	abcbdab bdcaba	Length of LCS is : 4	Length of LCS is : 4	✔
✔	treehouse elephant	Length of LCS is : 3	Length of LCS is : 3	✔
✔	AGGTAB GXTXAYB	Length of LCS is : 4	Length of LCS is : 4	✔

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 %)

```
def LongComSubS(st1, st2):
  ans = 0;
  for a in range(len(st1)):
         for b in range(len(st2)):
            k = 0;
            while ((a + k) < len(st1) and (b + k) < len(st2)
        and st1[a + k] == st2[b + k]):
                k = k + 1;
            ans = max(ans, k);
  return ans;

if __name__ == '__main__':

    A = input()
    B = input()
    i = len(A)
    j = len(B)
    print('Length of Longest Common Substring is', LongComSubS(A, B))
```

	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

Create a Naive recursive python program to find the minimum number of operations to convert str1 to str2

For example:

Input	Result
Python Peithen	Edit Distance 3

Answer: (penalty regime: 0 %)

Reset answer

```
def ed(x,y,m,n):
    if m==0:
        return n
    if n==0:
        return m
    if x[m-1]==y[n-1]:
        return ed(x,y,m-1,n-1)
    return 1+min(ed(x,y,m-1,n-1),ed(x,y,m,n-1),ed(x,y,m-1,n))
x=input()
y=input()
print("Edit Distance",ed(x,y,len(x),len(y)))
```

	Input	Expected	Got	
✔	Python Peithen	Edit Distance 3	Edit Distance 3	✔
✔	food money	Edit Distance 4	Edit Distance 4	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **4**

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 %)

```
def Lps(X):
    n=len(X)
    dp=[[0 for _ in range(n)] for _ in range(n)]
    for x in range(n):
        dp[x][x]=1
    for l in range(2,n+1):
        for i in range(n-l+1):
            j=i+l-1
            if X[i]==X[j]:
                dp[i][j]=dp[i+1][j-1]+2
            else:
                dp[i][j]=max(dp[i+1][j],dp[i][j-1])
    return dp[0][n-1]
X=input()
print("The length of the LPS is",Lps(X))
```

	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

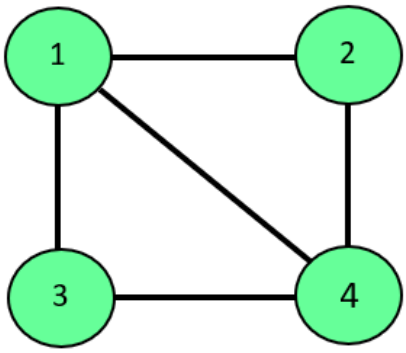
Marks for this submission: 20.00/20.00.

Question **5**

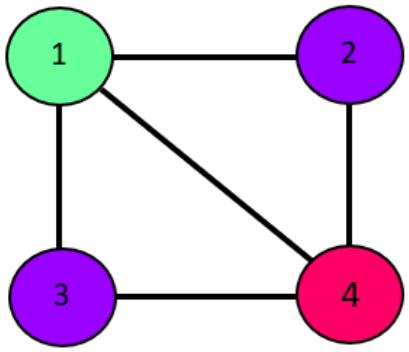
Correct

Mark 20.00 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

```
def isSafe(graph, color):
    for i in range(4):
        for j in range(i + 1, 4):
            if (graph[i][j] and color[j] == color[i]):
                return False
    return True

def graphColoring(graph, m, i, color):

    print('''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''')
def display(color):
    print("Solution Exists:" " Following are the assigned colors ")
    for i in range(4):
        print("Vertex", i+1 ," is given color: ",color[i])
if __name__ == '__main__':
    graph = [
        [ 0, 1, 1, 1 ],
        [ 1, 0, 1, 0 ],
```

	Expected	Got	
✔	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	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	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.