Started on	Tuesday, 29 April 2025, 1:45 PM
State	Finished
Completed on	Tuesday, 29 April 2025, 5:58 PM
Time taken	4 hours 13 mins
Overdue	2 hours 13 mins
Grade	**80.00** out of 100.00

Question **1**

Correct

Mark 20.00 out of 20.00

Create a python program to find the Edit distance between two strings using dynamic programming.

For example:

Input	Result
Cats Rats	No. of Operations required : 1

Answer: (penalty regime: 0 %)

Reset answer

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

```
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('No. of Operations required :',LD(str1,str2))
```

	Input	Expected	Got	
✔	Cats Rats	No. of Operations required : 1	No. of Operations required : 1	✔
✔	Saturday Sunday	No. of Operations required : 3	No. of Operations required : 3	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **2**

Correct

Mark 20.00 out of 20.00

Create a python program to find the length of longest common subsequence using naive recursive method

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(S1, S2, m, n):
    if m == 0 or n == 0:
        return 0
    if S1[m - 1] == S2[n - 1]:
        return 1 + lcs(S1, S2, m - 1, n - 1)
    else:
        return max(lcs(S1, S2, m, n - 1),
                   lcs(S1, S2, m - 1, n))
S1 = input()
S2 = input()
m = len(S1)
n = len(S2)

print("Length of LCS is ", lcs(S1, S2, m, n))
```

	Input	Expected	Got	
✔	AGGTAB GXTXAYB	Length of LCS is 4	Length of LCS is 4	✔
✔	saveetha engineering	Length of LCS is 2	Length of LCS is 2	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **3**

Not answered

Mark 0.00 out of 20.00

Write a python program to implement merge sort using iterative approach on the given list of values.

For example:

Test	Input	Result
Merge_Sort(S)	6 4 2 3 1 6 5	The Original array is: [4, 2, 3, 1, 6, 5] Array after sorting is: [1, 2, 3, 4, 5, 6]
Merge_Sort(S)	5 2 6 4 3 1	The Original array is: [2, 6, 4, 3, 1] Array after sorting is: [1, 2, 3, 4, 6]

Answer: (penalty regime: 0 %)

Ace editor not ready. Perhaps reload page?

Falling back to raw text area.

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 %)

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

```
def longest_palindromic_subsequence(s):
    n = len(s)
    dp = [[0] * n for _ in range(n)]
    for i in range(n):
        dp[i][i] = 1
    for length in range(2, n + 1):
        for i in range(n - length + 1):
            j = i + length - 1
            if s[i] == s[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]

sequence = input()
print("The length of the LPS is", longest_palindromic_subsequence(sequence))
```

	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

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

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

```
def lcw(u, v):
    m = len(u)
    n = len(v)
    dp = [[0] * (n + 1) for _ in range(m + 1)]
    length_lcw = 0
    lcw_i = 0
    for i in range(1, m + 1):
        for j in range(1, n + 1):
            if u[i - 1] == v[j - 1]:
                dp[i][j] = dp[i - 1][j - 1] + 1
                if dp[i][j] > length_lcw:
                    length_lcw = dp[i][j]
                    lcw_i = i - length_lcw
    return length_lcw, lcw_i

u = input()
v = input()
length_lcw, lcw_i = 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.