Started on	Tuesday, 29 April 2025, 1:49 PM
State	Finished
Completed on	Tuesday, 29 April 2025, 5:57 PM
Time taken	4 hours 7 mins
Overdue	2 hours 7 mins
Grade	**80.00** out of 100.00

Question **1**

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(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)
print('Longest Common Subword: ', end='')
if length_lcw > 0:
    print(u[lcw_i:lcw_i + length_lcw])
else:
```

	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 **2**

Correct

Mark 20.00 out of 20.00

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

For example:

Input	Result
abcbdab bdcaba	bdab

Answer: (penalty regime: 0 %)

```
def longest_common_subsequence(X, Y):
    dp = [[""] * (len(Y) + 1) for _ in range(len(X) + 1)]
    for i in range(1, len(X) + 1):
        for j in range(1, len(Y) + 1):
            if X[i - 1] == Y[j - 1]:
                dp[i][j] = dp[i - 1][j - 1] + X[i - 1]
            else:
                if len(dp[i - 1][j]) > len(dp[i][j - 1]):
                    dp[i][j] = dp[i - 1][j]
                else:
                    dp[i][j] = dp[i][j - 1]
    return dp[-1][-1]


X = input()
Y = input()
lcs = longest_common_subsequence(X, Y)
print(lcs)
```

	Input	Expected	Got	
✔	abcbdab bdcaba	bdab	bdab	✔
✔	treehouse elephant	eeh	eeh	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **3**

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

```
def expand(s, low, high):
    length = len(s)
    while low >= 0 and high < length and s[low] == s[high]:
        low = low - 1
        high = high + 1
    return s[low + 1:high]

def findLongestPalindromicSubstring(s):
    if not s or not len(s):
        return ''

    max_palindrome = ''
    for i in range(len(s)):
        odd_palindrome = expand(s, i, i)
        if len(odd_palindrome) > len(max_palindrome):
            max_palindrome = odd_palindrome

        even_palindrome = expand(s, i, i + 1)
        if len(even_palindrome) > len(max_palindrome):
            max_palindrome = even_palindrome

    return max_palindrome
```

	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 **4**

Not answered

Mark 0.00 out of 20.00

Write a python program for the implementation of merge sort on the given list of float values.

For example:

Input	Result
5 6.3 2.3 1.5 8.9 4.5	Given array is 6.3 2.3 1.5 8.9 4.5 Sorted array is 1.5 2.3 4.5 6.3 8.9
6 2.3 6.5 4.9 8.7 6.2 2.1	Given array is 2.3 6.5 4.9 8.7 6.2 2.1 Sorted array is 2.1 2.3 4.9 6.2 6.5 8.7

Answer: (penalty regime: 0 %)

```
1
```

Question **5**

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

```
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.