TT DS PYTHON MODULE-22

Started on	Friday, 16 May 2025, 2:47 PM
State	Finished
Completed on	Friday, 16 May 2025, 3:23 PM
Time taken	35 mins 22 secs
Grade	**80.00** out of 100.00

Question **1**

Correct

Mark 20.00 out of 20.00

Flag question

Create a Python program to find longest common substring or subword (LCW) of two strings using dynamic programming with bottor 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 o 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 **2**

Correct

Mark 20.00 out of 20.00

Flag question

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

Answer: (penalty regime: 0 %)

```
def lcs(str1 , str2):
    m = len(str1)
    n = len(str2)
    matrix = [[0]*(n+1) for i in range(m+1)]
    for i in range(m+1):
        for j in range(n+1):
            if i==0 or j==0:
                matrix[i][j] = 0
            elif str1[i-1] == str2[j-1]:
                matrix[i][j] = 1 + matrix[i-1][j-1]
            else:
```

```
                matrix[i][j] = max(matrix[i-1][j] , matrix[i][j-1])
    return matrix[-1][-1]
str1 = input()
str2 = input()
lcs_length = lcs(str1, str2)
print("Length of LCS is : {}".format(lcs_length))
```

	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 **3**

Correct

Mark 20.00 out of 20.00

Flag question

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 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('Edit Distance',LD(str1,str2))
```

	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

Flag question

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 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**

Not answered

Mark 0.00 out of 20.00

Flag question

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

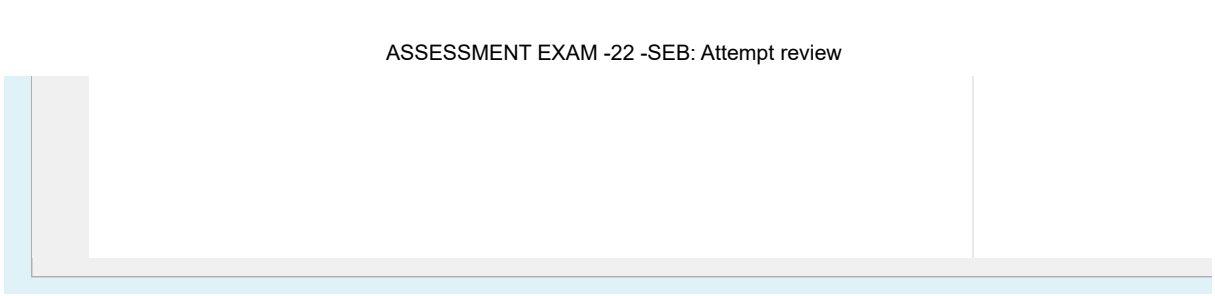
For example:

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 %)

```

```



Finish r