Started on	Friday, 12 April 2024, 5:47 PM
State	Finished
Completed on	Friday, 12 April 2024, 8:03 PM
Time taken	2 hours 15 mins
Overdue	15 mins 41 secs
Grade	**80.00** out of 100.00

Question **1**

Not answered

Mark 0.00 out of 20.00

Write a python program to implement quick sort on the given values and print the sorted list and pivot value of each iteration.

For example:

Input	Result
5 41 21 6 34 8	Input List [41, 21, 6, 34, 8] pivot: 41 pivot: 8 pivot: 21 Sorted List [6, 8, 21, 34, 41]
4 5 2 49 3	Input List [5, 2, 49, 3] pivot: 5 pivot: 3 Sorted List [2, 3, 5, 49]

Answer: (penalty regime: 0 %)

```

```

Question **2**

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 %)

```python
dp = [[-1 for i in range(1001)]for j in range(1001)]
def lps(s1, s2, n1, n2):

    if (n1 == 0 or n2 == 0):
        return 0

    if (dp[n1][n2] != -1):
        return dp[n1][n2]

    if (s1[n1 - 1] == s2[n2 - 1]):
        dp[n1][n2] = 1 + lps(s1, s2, n1 - 1, n2 - 1)
        return dp[n1][n2]
    else:
        dp[n1][n2] = max(lps(s1, s2, n1 - 1, n2), lps(s1, s2, n1, n2 - 1))
        return dp[n1][n2]
seq = input()
n = len(seq)

s2 = seq
s2 = s2[::-1]
print(f"The length of the LPS is {lps(s2, seq, n, n)}")

```

	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 **3**

Correct

Mark 20.00 out of 20.00

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

Answer: (penalty regime: 0 %)

```python
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 **4**

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 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 **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 top-down approach or memoization.

Problem Description

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)	potato tomato	Longest Common Subword: ato

Answer: (penalty regime: 0 %)

Reset answer

```
def lcw(u, v):
    c = [[-1]*(len(v) + 1) for _ in range(len(u) + 1)]
    lcw_i = lcw_j = -1
    length_lcw = 0
    for i in range(len(u)):
        for j in range(len(v)):
            temp = lcw_starting_at(u, v, c, i, j)
            if length_lcw < temp:
                length_lcw = temp
                lcw_i = i
                lcw_j = j
    return length_lcw, lcw_i, lcw_j
def lcw_starting_at(u, v, c, i, j):
    if c[i][j] >= 0:
        return c[i][j]
    if i == len(u) or j == len(v):
        q = 0
    elif u[i] != v[j]:
        q = 0
    else:
        q = 1 + lcw_starting_at(u, v, c, i + 1, j + 1)
    c[i][j] = q
```

	Test	Input	Expected	Got	
✓	lcw(u, v)	potato tomato	Longest Common Subword: ato	Longest Common Subword: ato	✓
✓	lcw(u, v)	snakegourd bottlegourd	Longest Common Subword: egourd	Longest Common Subword: egourd	✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.