Started on	Monday, 19 May 2025, 10:19 AM
State	Finished
Completed on	Monday, 19 May 2025, 12:20 PM
Time taken	2 hours 1 min
Overdue	1 min 27 secs
Grade	**80.00** out of 100.00

Question **1**

Not answered

Mark 0.00 out of 20.00

Write a Python program to sort unsorted numbers using Multi-key quicksort

For example:

Test	Input	Result

`quick_sort_3partition(nums, 0, len(nums)-1)`	5
	4
	3
	5
	1
	2
		`Original list:`
		`[4, 3, 5, 1, 2]`
		`After applying Random Pivot Quick Sort the said list becomes:`
		`[1, 2, 3, 4, 5]`

`quick_sort_3partition(nums, 0, len(nums)-1)`	6
	21
	10
	3
	65
	4
	8
		`Original list:`
		`[21, 10, 3, 65, 4, 8]`
		`After applying Random Pivot Quick Sort the said list becomes:`
		`[3, 4, 8, 10, 21, 65]`

Answer: (penalty regime: 0 %)

```
1
```

Question **2**

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 **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 printSubStr(s, 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 **4**

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 lcs(str1, str2):
    m, n = len(str1), len(str2)
    table = [[0] * (n+1) for _ in range(m+1)]
    for i in range(1, m+1):
        for j in range(1, n+1):
            if str1[i-1] == str2[j-1]:
                table[i][j] = 1 + table[i-1][j-1]
            else:
                table[i][j] = max(table[i-1][j], table[i][j-1])
    lcs = ""
    i, j = m, n
    while i > 0 and j > 0:
        if str1[i-1] == str2[j-1]:
            lcs = str1[i-1] + lcs
            i -= 1
            j -= 1
        elif table[i-1][j] > table[i][j-1]:
            i -= 1
        else:
            j -= 1
    return lcs
str1=input()
```

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

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):
    m, n = len(u), len(v)
    table = [[0] * (n+1) for _ in range(m+1)]
    for i in range(1, m+1):
        for j in range(1, n+1):
            if u[i-1] == v[j-1]:
                table[i][j] = 1 + table[i-1][j-1]
            else:
                table[i][j] = max(table[i-1][j], table[i][j-1])
    lcw = ""
    i, j = m, n
    while i > 0 and j > 0:
        if u[i-1] == v[j-1]:
            lcw = u[i-1] + lcw
            i -= 1
            j -= 1
        elif table[i][j] >= table[i][j]:
            i -= 1
        else:
            j -= 1
    return lcw
u=input()
```

	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.