Started on	Wednesday, 14 May 2025, 11:24 AM
State	Finished
Completed on	Wednesday, 14 May 2025, 11:53 AM
Time taken	29 mins 3 secs
Grade	**100.00** 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 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 subst 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 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=len(u)
    n=len(v)
    max=0
    end=m
    dp=[[0 for i in range(n+1)] for j 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]:
                dp[i][j]=dp[i-1][j-1]+1
                if dp[i][j]>max:
                    max=dp[i][j]
                    end=i
    return u[end-max:end]

u=input()
v=input()
print("Longest Common Subword:",lcw(u,v))
```

	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.

Question **2**

Correct

Mark 20.00 out of 20.00

Create a python program to find the longest palindromic substring using Brute force method in a given string.

For example:

Input	Result
mojologiccigolmojo	logiccigol

Answer: (penalty regime: 0 %)

Reset answer

```
class Solution(object):
    def longestPalindrome(self, s):
        def expand(l, r):
            while l >= 0 and r < len(s) and s[l] == s[r]:
                l -= 1
                r += 1
            return s[l+1:r]

        longest = ""
        for i in range(len(s)):
            odd = expand(i, i)
            even = expand(i, i+1)

            if len(odd) > len(longest) or (len(odd) == len(longest) and odd < longest):
                longest = odd
            if len(even) > len(longest) or (len(even) == len(longest) and even < longest):
                longest = even

        return longest

ob1 = Solution()
str1 = input().strip()
```

	Input	Expected	Got	
✔	mojologiccigolmojo	logiccigol	logiccigol	✔
✔	sampleelpams	pleelp	pleelp	✔

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 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]:
        count=0
    else:
        count=1
    res=min([LD(s[:-1],t)+1,LD(s,t[:-1])+1,LD(s[:-1],t[:-1])+count])
    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 **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(x,y):
    m=len(x)
    n=len(y)
    dp=[["" for _ in range (n+1)]for _ 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]:
                dp[i][j]=dp[i-1][j-1]+x[i-1]
            else:
                dp[i][j] = dp[i - 1][j] if len(dp[i - 1][j]) > len(dp[i][j - 1]) else dp[i][j - 1

    return dp[m][n]

str1=input()
str2=input()
print(f"{lcs(str1,str2)}")
```

	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

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 %)

```
def quick_sort_3partition(nums,st,en):
    if en-st>1:
        p=partition(nums,st,en)
        quick_sort_3partition(nums,st,p)
        quick_sort_3partition(nums,p+1,en)
def partition(nums,st,en):
    pivot=nums[st]
    i=st+1
    j=en-1

    while True:
        while(i<=j and nums[i]<=pivot):
            i=i+1
        while(i<=j and nums[j]>=pivot):
            j=j-1
        if i<=j:
            nums[i],nums[j]=nums[j],nums[i]
        else:
            nums[st],nums[j]=nums[j],nums[st]
            return j


```

	Test	Input	Expected	Got
✔	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]	Original list: [4, 3, 5, 1, 2] After applying Random Pivot Quick Sort the said list becomes: [1, 2, 3, 4, 5]

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

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.