Started on	Friday, 16 May 2025, 10:22 AM
State	Finished
Completed on	Friday, 16 May 2025, 11:16 AM
Time taken	54 mins 8 secs
Grade	**100.00** out of 100.00

Question **1**

Correct

Mark 20.00 out of 20.00

Create a python program to find the longest common subsequence using Memoization Implementation.

For example:

Input	Result
AGGTAB GXTXAYB	Length of LCS is 4

Answer: (penalty regime: 0 %)

```
def lcs(x,y,m,n,dp):
    if m==0 or n==0:
        return 0
    if dp[m][n]!=-1:
        return dp[m][n]
    if x[m-1]==y[n-1]:
        dp[m][n]=1+lcs(x,y,m-1,n-1,dp)
        return dp[m][n]
    dp[m][n]=max(lcs(x,y,m,n-1,dp),lcs(x,y,m-1,n,dp))
    return  dp[m][n]
x=input()
y=input()
dp=[[-1]*(len(y)+1) for i in range(len(x)+1)]
print("Length of LCS is",lcs(x,y,len(x),len(y),dp))
```

	Input	Expected	Got	
✔	AGGTAB GXTXAYB	Length of LCS is 4	Length of LCS is 4	✔
✔	SAMPLE SAEMSUNG	Length of LCS is 3	Length of LCS is 3	✔
✔	saveetha sabeetha	Length of LCS is 7	Length of LCS is 7	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **2**

Correct

Mark 20.00 out of 20.00

LONGEST COMMON SUBSTRING PROBLEM

The longest common substring problem is the problem of finding the longest string (or strings) that is a substring (or are substrings) of two strings.

Answer: (penalty regime: 0 %)

```
def LCS(X, Y, m, n):

    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]
if __name__ == '__main__':
    X = input()
    Y = input()
    m = len(X)
    n = len(Y)
    print('The longest common substring is', LCS(X, Y, m, n))
```

	Input	Expected	Got	
✔	ABC BABA	The longest common substring is AB	The longest common substring is AB	✔
✔	abcdxyz xyzabcd	The longest common substring is abcd	The longest common substring is abcd	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **3**

Correct

Mark 20.00 out of 20.00

Write a Program for Implementing merge sort using python recursion.

For example:

Test	Input	Result
merge_sort(inp_arr)	8 11 31 7 41 101 56 77 2	Input Array: [11, 31, 7, 41, 101, 56, 77, 2] Sorted Array: [2, 7, 11, 31, 41, 56, 77, 101]
merge_sort(inp_arr)	5 61 2 41 80 9	Input Array: [61, 2, 41, 80, 9] Sorted Array: [2, 9, 41, 61, 80]

Answer: (penalty regime: 0 %)

```
def merge_sort(inp_arr):
    n = len(inp_arr)
    current_size = 1

    while current_size < n:
        left = 0
        while left < n - 1:
            mid = min(left + current_size - 1, n - 1)
            right = min(left + 2 * current_size - 1, n - 1)

            merge(inp_arr, left, mid, right)
            left += 2 * current_size

        current_size *= 2


def merge(inp_arr, left, mid, right):
    n1 = mid - left + 1
    n2 = right - mid

    L = [inp_arr[left + i] for i in range(n1)]
    R = [inp_arr[mid + 1 + i] for i in range(n2)]
```

	Test	Input	Expected	Got	
✔	merge_sort(inp_arr)	8 11 31 7 41 101 56 77 2	Input Array: [11, 31, 7, 41, 101, 56, 77, 2] Sorted Array: [2, 7, 11, 31, 41, 56, 77, 101]	Input Array: [11, 31, 7, 41, 101, 56, 77, 2] Sorted Array: [2, 7, 11, 31, 41, 56, 77, 101]	✔
✔	merge_sort(inp_arr)	5 61 2 41 80 9	Input Array: [61, 2, 41, 80, 9] Sorted Array: [2, 9, 41, 61, 80]	Input Array: [61, 2, 41, 80, 9] Sorted Array: [2, 9, 41, 61, 80]	✔

	Test	Input	Expected	Got	
✔	merge_sort(inp_arr)	6 100 30 29 5 600 21	Input Array: [100, 30, 29, 5, 600, 21] Sorted Array: [5, 21, 29, 30, 100, 600]	Input Array: [100, 30, 29, 5, 600, 21] Sorted Array: [5, 21, 29, 30, 100, 600]	✔
✔	merge_sort(inp_arr)	4 21 10 30 5	Input Array: [21, 10, 30, 5] Sorted Array: [5, 10, 21, 30]	Input Array: [21, 10, 30, 5] Sorted Array: [5, 10, 21, 30]	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **4**

Correct

Mark 20.00 out of 20.00

Create a python program to compute the edit distance between two given strings using iterative method.

For example:

Input	Result
kitten sitting	3

Answer: (penalty regime: 0 %)

```
def mind(x,y):
    m=len(x)
    n=len(y)
    dp = [[0] * (n + 1) for _ in range(m + 1)]

    for i in range(m+1):
        for j in range(n+1):
            if i==0:
                dp[i][j]=j
            elif j==0:
                dp[i][j]=i
            elif x[i-1]==y[j-1]:
                dp[i][j]=dp[i-1][j-1]
            else:
                dp[i][j]=min(dp[i-1][j-1],dp[i][j-1],dp[i-1][j])+1
    return dp[m][n]
x=input()
y=input()
print(mind(x,y))
```

	Input	Expected	Got	
✔	kitten sitting	3	3	✔
✔	medium median	2	2	✔

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 the longest palindromic substring using Brute force method in a given string.

For example:

Input	Result
mojologiccigolmojo	logiccigol

Answer: (penalty regime: 0 %)

Reset answer

```
def printSubStr(str, low, high):

    for i in range(low, high + 1):
        print(str[i], end = "")

def longestPalindrome(str):
    n=len(str)
    max_len=0
    start=0
    for i in range(n):
        for j in range(1,n):
            s=str[i:j+1]
            if s==s[::-1]:
                cur=j-i+1
                if cur>max_len:
                    max_len=cur
                    start=i
    printSubStr(str, start, start + max_len - 1)

if __name__ == '__main__':

    str = input()
```

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

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.