Started on	Friday, 23 May 2025, 11:28 AM
State	Finished
Completed on	Friday, 23 May 2025, 1:23 PM
Time taken	1 hour 54 mins
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 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)
    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 (str[i + k] != str[j - k]):
                    flag = 0

            if (flag != 0 and (j - i + 1) > maxLength):
                start = i
                maxLength = j - i + 1

    printSubStr(str, start, start + maxLength - 1)

```

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

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **2**

Correct

Mark 20.00 out of 20.00

Write a Program for Implementing merge sort on float values using python recursion.

For example:

Test	Input	Result
merge_sort(inp_arr)	5 3.2 1.6 9.5 4.3 4.55	Input Array: [3.2, 1.6, 9.5, 4.3, 4.55] Sorted Array: [1.6, 3.2, 4.3, 4.55, 9.5]
merge_sort(inp_arr)	6 3.2 1.2 5.3 9.6 8.5 7.4	Input Array: [3.2, 1.2, 5.3, 9.6, 8.5, 7.4] Sorted Array: [1.2, 3.2, 5.3, 7.4, 8.5, 9.6]

Answer: (penalty regime: 0 %)

```
def merge_sort(inp_arr):
    if len(inp_arr)>1:
        mid = 0 + len(inp_arr) // 2
        L = inp_arr[:mid]
        R = inp_arr[mid:]
        merge_sort(L)
        merge_sort(R)
        i = j = k =0
        while i <len(L) and j <len(R):
            if L[i] < R[j]:
                inp_arr[k] = L[i]
                i +=1
            else:
                inp_arr[k] = R[j]
                j +=1
            k +=1
        while i <len(L):
            inp_arr[k] = L[i]
            i += 1
            k +=1
        while j < len(R):
            inp_arr[k] = R[j]
```

	Test	Input	Expected	Got	
✔	merge_sort(inp_arr)	5 3.2 1.6 9.5 4.3 4.55	Input Array: [3.2, 1.6, 9.5, 4.3, 4.55] Sorted Array: [1.6, 3.2, 4.3, 4.55, 9.5]	Input Array: [3.2, 1.6, 9.5, 4.3, 4.55] Sorted Array: [1.6, 3.2, 4.3, 4.55, 9.5]	✔
✔	merge_sort(inp_arr)	6 3.2 1.2 5.3 9.6 8.5 7.4	Input Array: [3.2, 1.2, 5.3, 9.6, 8.5, 7.4] Sorted Array: [1.2, 3.2, 5.3, 7.4, 8.5, 9.6]	Input Array: [3.2, 1.2, 5.3, 9.6, 8.5, 7.4] Sorted Array: [1.2, 3.2, 5.3, 7.4, 8.5, 9.6]	✔

	Test	Input	Expected	Got	
✔	merge_sort(inp_arr)	4 3.2 1.5 6.9 8.0	Input Array: [3.2, 1.5, 6.9, 8.0] Sorted Array: [1.5, 3.2, 6.9, 8.0]	Input Array: [3.2, 1.5, 6.9, 8.0] Sorted Array: [1.5, 3.2, 6.9, 8.0]	✔

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 %)

```
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

LONGEST COMMON SUBSTRING PROBLEM

Given two strings 'X' and 'Y', find the length of the longest common substring.

Answer: (penalty regime: 0 %)

```
def LCSubStr(X, Y, m, n):
    LCSuff = [[0 for k in range(n+1)] for l in range(m+1)]
    result = 0

    for i in range(m + 1):
        for j in range(n + 1):
            if (i == 0 or j == 0):
                LCSuff[i][j] = 0
            elif (X[i-1] == Y[j-1]):
                LCSuff[i][j] = LCSuff[i-1][j-1] + 1
                result = max(result, LCSuff[i][j])
            else:
                LCSuff[i][j] = 0
    return result

X = input()
Y = input()

m = len(X)
n = len(Y)

print('Length of Longest Common Substring is',
```

	Input	Expected	Got	
✔	ABC BABA	Length of Longest Common Substring is 2	Length of Longest Common Substring is 2	✔
✔	abcdxyz xyzabcd	Length of Longest Common Substring is 4	Length of Longest Common Substring is 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 compute the edit distance between two given strings using iterative method.

For example:

Input	Result
kitten sitting	3

Answer: (penalty regime: 0 %)

```
def edit_distance(string1, string2):
    if len(string1) > len(string2):
        difference = len(string1) - len(string2)
        string1[:difference]
    elif len(string2) > len(string1):
        difference = len(string2) - len(string1)
        string2[:difference]
    else:
        difference = 0
    for i in range(len(string1)):
        if string1[i] != string2[i]:
            difference += 1
    return difference

str1=input()
str2=input()
print(edit_distance(str1,str2))
```

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

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.