Started on	Friday, 11 April 2025, 10:17 AM
State	Finished
Completed on	Tuesday, 15 April 2025, 1:22 AM
Time taken	3 days 15 hours
Overdue	3 days 13 hours
Grade	**100.00** out of 100.00

Question **1**

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

Ace editor not ready. Perhaps reload page?

Falling back to raw text area.

```
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 **2**

Correct

Mark 20.00 out of 20.00

Write a short recursive Python function that finds the minimum and maximum values in a sequence without using any loops.

For example:

Input	Result
4 51 20 31 47	51
4 12 20 5 6	20

Answer: (penalty regime: 0 %)

Ace editor not ready. Perhaps reload page?
Falling back to raw text area.

```
def maximum_in_list(nums):
    if len(nums) == 1:
        return nums[0]
    else:
        return max(nums[0],maximum_in_list(nums[1:]))
list1=[]
n=int(input())
for i in range(n):
    list1.append(int(input()))
print(maximum_in_list(list1))
```

	Input	Expected	Got	
✔	4 51 20 31 47	51	51	✔

	Input	Expected	Got	
✔	4 12 20 5 6	20	20	✔
✔	6 10 41 75 84 62 35	84	84	✔
✔	3 74 52 20	74	74	✔

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 length of longest common subsequence using naive recursive method

For example:

Input	Result
AGGTAB GXTXAYB	Length of LCS is 4

Answer: (penalty regime: 0 %)

Ace editor not ready. Perhaps reload page?

Falling back to raw text area.

```
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
```

	Input	Expected	Got	
✔	AGGTAB GXTXAYB	Length of LCS is 4	Length of LCS is 4	✔
✔	saveetha engineering	Length of LCS is 2	Length of LCS is 2	✔

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

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 %)

Ace editor not ready. Perhaps reload page?
Falling back to raw text area.

```
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
```

	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 **5**

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

Ace editor not ready. Perhaps reload page?
Falling back to raw text area.

```
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)
```

	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.