Started on	Wednesday, 9 April 2025, 10:08 AM
State	Finished
Completed on	Wednesday, 9 April 2025, 10:31 AM
Time taken	23 mins 41 secs
Grade	**80.00** out of 100.00

Question **1**

Incorrect

Mark 0.00 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 minmax(x,y,result)
```

Syntax Error(s)

```
  File "__tester__.python3", line 1
    def minmax(x,y,result)
                          ^
SyntaxError: invalid syntax
```

Incorrect

Marks for this submission: 0.00/20.00.

Question **2**

Correct

Mark 20.00 out of 20.00

LONGEST PALINDROMIC SUBSEQUENCE

Given a sequence, find the length of the longest palindromic subsequence in it.

For example:

Input	Result
ABBDCACB	The length of the LPS is 5

Answer: (penalty regime: 0 %)

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

```
def Lps(X):
    n=len(X)
    dp=[[0 for _ in range(n)] for _ in range(n)]
    for x in range(n):
        dp[x][x]=1
    for l in range(2,n+1):
        for i in range(n-l+1):
            j=i+l-1
            if X[i]==X[j]:
                dp[i][j]=dp[i+1][j-1]+2
            else:
                dp[i][j]=max(dp[i+1][j],dp[i][j-1])
    return dp[0][n-1]
X=input()
print("The length of the LPS is",Lps(X))
```

	Input	Expected	Got	
✔	ABBDCACB	The length of the LPS is 5	The length of the LPS is 5	✔
✔	BBABCBCAB	The length of the LPS is 7	The length of the LPS is 7	✔
✔	cbbd	The length of the LPS is 2	The length of the LPS is 2	✔
✔	abbab	The length of the LPS is 4	The length of the LPS is 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 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(x,y,m,n):
    if m==0 or n==0:
        return 0
    elif x[m-1]==y[n-1]:
        return 1+lcs(x,y,m-1,n-1)
    else:
        return max(lcs(x,y,m,n-1),lcs(x,y,m-1,n))
X = input()
Y = input()
print ("Length of LCS is ", lcs(X , Y, len(X), len(Y)) )
```

	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

Create a Python program to find longest common substring or subword (LCW) of two strings using dynamic programming with bottom-up approach.

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)	bisect trisect	Longest Common Subword: isect

Answer: (penalty regime: 0 %)

Reset answer

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

```
def lcw(x,y):
    m=len(x)
    n=len(y)
    dp=[[0]*(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]=1+dp[i-1][j-1]
            else:
                dp[i][j]=0
    maxl=0
    end_i=0
    end_j=0
    for i in range(1,len(x)+1):
        for j in range(1,len(y)+1):
            if dp[i][j]>maxl:
                maxl=dp[i][j]
                end_i=i
```

	Test	Input	Expected	Got	
✔	lcw(u, v)	bisect trisect	Longest Common Subword: isect	Longest Common Subword: isect	✔
✔	lcw(u, v)	director conductor	Longest Common Subword: ctor	Longest Common Subword: ctor	✔

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 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

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

```
def edit_distance(str1, str2, a, b):
        dp = [[0 for x in range(b + 1)] for x in range(a + 1)]
        for i in range(a + 1):
                for j in range(b + 1):
                        if i == 0:
                                dp[i][j] = j # Min. operations = j

                        elif j == 0:
                                dp[i][j] = i # Min. operations = i

                        elif str1[i-1] == str2[j-1]:
                                dp[i][j] = dp[i-1][j-1]
                        else:
                                dp[i][j] = 1 + min(dp[i][j-1],dp[i-1][j],dp[i-1][j-1])

        return dp[a][b]
str1 = input()
str2 = input()
```

	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.