Started on	Monday, 28 April 2025, 2:16 PM
State	Finished
Completed on	Monday, 28 April 2025, 2:55 PM
Time taken	39 mins 20 secs
Grade	**80.00** out of 100.00

Question **1**

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 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

            elif j == 0:
                dp[i][j] = 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()
print('No. of Operations required :',edit_distance(str1, str2, len(str1), len(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 **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

```python
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.

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 longest(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] = dp[i - 1][j - 1] + 1
            else:
                dp[i][j] = max(dp[i - 1][j], dp[i][j - 1])

    lcs_length = dp[m][n]
    lcs = [''] * lcs_length
    i, j = m, n

    while i > 0 and j > 0:
        if X[i - 1] == Y[j - 1]:
            lcs[lcs_length - 1] = X[i - 1]
            i -= 1
            j -= 1
```

	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**

Incorrect

Mark 0.00 out of 20.00

Create a Python program to find longest common substring or subword (LCW) of two strings using dynamic programming with top-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 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)`	`potato` `tomato`	`Longest Common Subword: ato`

Answer: (penalty regime: 0 %)

Reset answer

```python
def lcw(u, v):
    c = [[-1]*(len(v) + 1) for _ in range(len(u) + 1)]
    lcw_i = lcw_j = -1
    length_lcw = 0
    for i in range(len(u)):
        for j in range(len(v)):
            temp = lcw_starting_at(u, v, c, i, j)
            if length_lcw < temp:
                length_lcw = temp
                lcw_i = i
                lcw_j = j
    return length_lcw, lcw_i, lcw_j
def lcw_starting_at(str1, str2, a, b,c):
    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

            elif j == 0:
                dp[i][j] = i

```

Syntax Error(s)

```
Sorry: TabError: inconsistent use of tabs and spaces in indentation (__tester__.python3, line 15)
```

Incorrect

Marks for this submission: 0.00/20.00.

Question **5**

Correct

Mark 20.00 out of 20.00

Write a python program to implement quick sort on the given float array values.

For example:

Input	Result
5 6.9 8.3 2.1 1.5 6.4	left: [] right: [] left: [] right: [] left: [1.5] right: [6.4] left: [] right: [] left: [1.5, 2.1, 6.4] right: [8.3] [1.5, 2.1, 6.4, 6.9, 8.3]
6 3.1 2.4 5.6 4.3 6.2 7.8	left: [] right: [] left: [] right: [] left: [] right: [] left: [] right: [7.8] left: [4.3] right: [6.2, 7.8] left: [2.4] right: [4.3, 5.6, 6.2, 7.8] [2.4, 3.1, 4.3, 5.6, 6.2, 7.8]

Answer: (penalty regime: 0 %)

```
def quickSort(arr):
    if arr==[]:
        return arr
    pivot=arr[0:1]
    left=quickSort([x for x in arr[1:] if x<pivot[0]])
    right=quickSort([x for x in arr[1:] if x>=pivot[0]])
    print("left: ",left)
    print("right: ",right)
    return left+pivot+right

l=[float(input()) for i in range(int(input()))]
s=quickSort(l)
print(s)
```

	Input	Expected	Got	
✓	5 6.9 8.3 2.1 1.5 6.4	left: [] right: [] left: [] right: [] left: [1.5] right: [6.4] left: [] right: [] left: [1.5, 2.1, 6.4] right: [8.3] [1.5, 2.1, 6.4, 6.9, 8.3]	left: [] right: [] left: [] right: [] left: [1.5] right: [6.4] left: [] right: [] left: [1.5, 2.1, 6.4] right: [8.3] [1.5, 2.1, 6.4, 6.9, 8.3]	✓
✓	6 3.1 2.4 5.6 4.3 6.2 7.8	left: [] right: [] left: [] right: [] left: [] right: [] left: [] right: [7.8] left: [4.3] right: [6.2, 7.8] left: [2.4] right: [4.3, 5.6, 6.2, 7.8] [2.4, 3.1, 4.3, 5.6, 6.2, 7.8]	left: [] right: [] left: [] right: [] left: [] right: [] left: [] right: [7.8] left: [4.3] right: [6.2, 7.8] left: [2.4] right: [4.3, 5.6, 6.2, 7.8] [2.4, 3.1, 4.3, 5.6, 6.2, 7.8]	✓
✓	8 1.2 1.3 4.2 5.3 6.4 7.3 6.8 9.2	left: [] right: [] left: [] right: [] left: [6.8] right: [9.2] left: [] right: [6.8, 7.3, 9.2] left: [] right: [6.4, 6.8, 7.3, 9.2] left: [] right: [5.3, 6.4, 6.8, 7.3, 9.2] left: [] right: [4.2, 5.3, 6.4, 6.8, 7.3, 9.2] left: [] right: [1.3, 4.2, 5.3, 6.4, 6.8, 7.3, 9.2] [1.2, 1.3, 4.2, 5.3, 6.4, 6.8, 7.3, 9.2]	left: [] right: [] left: [] right: [] left: [6.8] right: [9.2] left: [] right: [6.8, 7.3, 9.2] left: [] right: [6.4, 6.8, 7.3, 9.2] left: [] right: [5.3, 6.4, 6.8, 7.3, 9.2] left: [] right: [4.2, 5.3, 6.4, 6.8, 7.3, 9.2] left: [] right: [1.3, 4.2, 5.3, 6.4, 6.8, 7.3, 9.2] [1.2, 1.3, 4.2, 5.3, 6.4, 6.8, 7.3, 9.2]	✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.