Started on	Friday, 16 May 2025, 2:21 PM
State	Finished
Completed on	Friday, 16 May 2025, 2:51 PM
Time taken	30 mins 22 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 longest palindromic substring using optimal algorithm Expand around center.

For example:

Test	Input	Result
findLongestPalindromicSubstring(s)	samsunggnusgnusam	sunggnus

Answer: (penalty regime: 0 %)

Reset answer

```
def printSubStr(ss, 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)
s = input()

```

	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.

Question **2**

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 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(LD(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.

Question **3**

Correct

Mark 20.00 out of 20.00

To Write a Python Program to find longest common subsequence using Dynamic Programming

For example:

Input	Result
abcbdab bdcaba	bdab

Answer: (penalty regime: 0 %)

```
def lcs(u, v):
    c = [[-1]*(len(v) + 1) for _ in range(len(u) + 1)]
    for i in range(len(u) + 1):
        c[i][len(v)] = 0
    for j in range(len(v)):
        c[len(u)][j] = 0

    for i in range(len(u) - 1, -1, -1):
        for j in range(len(v) - 1, -1, -1):
            if u[i] == v[j]:
                c[i][j] = 1 + c[i + 1][j + 1]
            else:
                c[i][j] = max(c[i + 1][j], c[i][j + 1])
    return c

def print_lcs(u, v, c):
    i = j = 0
    while not (i == len(u) or j == len(v)):
        if u[i] == v[j]:
            print(u[i], end='')
            i += 1
            j += 1
```

	Input	Expected	Got	
✔	abcbdab bdcaba	bdab	bdab	✔
✔	treehouse elephant	eeh	eeh	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **4**

Not answered

Mark 0.00 out of 20.00

Write a Python program to calculate the harmonic sum of n-1.
Note: The harmonic sum is the sum of reciprocals of the positive integers.

Example:

$$1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\frac{1}{5}+\cdots$$

For example:

Input	Result
5	2.283333333333333
7	2.5928571428571425

Answer: (penalty regime: 0 %)

```
1
```

Question **5**

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]

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.