Started on	Monday, 5 May 2025, 10:51 AM
State	Finished
Completed on	Monday, 5 May 2025, 3:48 PM
Time taken	4 hours 56 mins
Overdue	2 hours 56 mins
Grade	**80.00** out of 100.00

Question **1**
Correct
Mark 20.00 out of 20.00
Flag question

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

```python

    for i in range(length):

        palindrome1 = expand(s, i, i)

        palindrome2 = expand(s, i, i + 1)


        if len(palindrome1) > len(palindrome2) and len(palindrome1) > (end - start):
            start = i - len(palindrome1) // 2
            end = i + len(palindrome1) // 2
        elif len(palindrome2) > (end - start):
            start = i - len(palindrome2) // 2 + 1
            end = i + len(palindrome2) // 2

    return s[start:end + 1]

if __name__ == '__main__':

    s = input()

    print(findLongestPalindromicSubstring(s))
```

	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**
Not answered
Mark 0.00 out of 20.00
Flag question

Write a Python Program to print the fibonacci series upto n_terms using Recursion.

For example:

Input	Result

`10`	`Fibonacci series:`
	`0`
	`1`
	`1`
	`2`
	`3`
	`5`
	`8`
	`13`
	`21`
	`34`

`5`	`Fibonacci series:`
	`0`
	`1`
	`1`
	`2`
	`3`

`7`	`Fibonacci series:`
	`0`
	`1`
	`1`
	`2`
	`3`
	`5`
	`8`

Answer: (penalty regime: 0 %)

```

```

Question 3
Correct
Mark 20.00 out of 20.00
Flag question

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

Answer: (penalty regime: 0 %)

```python
def longest_common_subsequence_length(X, Y):
    m = len(X)
    n = len(Y)

    # Create a 2D table to store the lengths of LCSs
    dp = [[0] * (n + 1) for _ in range(m + 1)]

    # Fill the dp table using bottom-up approach
    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])

    return dp[m][n]

# Example input
X = input()
Y = input()

result = longest_common_subsequence_length(X, Y)
```

	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
Flag question

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 %)

```python

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
s=input()
t=input()
print(LD(s,t))

```

	Input	Expected	Got	
	kitten sitting	3	3	
	medium median	2	2	

Passed all tests!

Correct

Marks for this submission: 20.00/20.00.

Question **5**

Correct

Mark 20.00 out of 20.00

Flag question

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]
if __name__ == '__main__':
    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.