Started on	Friday, 4 July 2025, 1:31 PM
State	Finished
Completed on	Friday, 4 July 2025, 1:53 PM
Time taken	22 mins 5 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

```python
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

Write a Python Program to print factorial of a number recursively.

For example:

Input	Result
5	`Factorial of number 5 = 120`
6	`Factorial of number 6 = 720`

Answer: (penalty regime: 0 %)

```
def Factorial(n):
    if n==0 or n==1:
        return 1
    else:
        return n * Factorial(n-1)
n=int(input())

print("Factorial of number",n,"=",Factorial(n));
```

	Input	Expected	Got	
✓	5	`Factorial of number 5 = 120`	`Factorial of number 5 = 120`	✓
✓	6	`Factorial of number 6 = 720`	`Factorial of number 6 = 720`	✓
✓	7	`Factorial of number 7 = 5040`	`Factorial of number 7 = 5040`	✓
✓	8	`Factorial of number 8 = 40320`	`Factorial of number 8 = 40320`	✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.

Question **3**

Incorrect

Mark 0.00 out of 20.00

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

Answer: (penalty regime: 0 %)

```python
def longest_common_subsequence(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	bdab	✖
✖	treehouse elephant	Length of LCS is : 3	eeh	✖
✖	AGGTAB GXTXAYB	Length of LCS is : 4	GTAB	✖

Some hidden test cases failed, too.
Your code must pass all tests to earn any marks. Try again.

Show differences

Incorrect

Marks for this submission: 0.00/20.00.

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 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(u, v, c, i, j):
    if c[i][j] >= 0:
        return c[i][j]

    if i == len(u) or j == len(v):
        q = 0
    elif u[i] != v[j]:
        q = 0
    else:
        q = 1 + lcw_starting_at(u, v, c, i + 1, j + 1)
```

	Test	Input	Expected	Got	
✓	`lcw(u, v)`	potato tomato	Longest Common Subword: ato	Longest Common Subword: ato	✓
✓	`lcw(u, v)`	snakegourd bottlegourd	Longest Common Subword: egourd	Longest Common Subword: egourd	✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.

Question **5**

Correct

Mark 20.00 out of 20.00

Given a string s, return *the longest palindromic substring* in s.

Example 1:

```
Input: s = "babad"
Output: "bab"
Explanation: "aba" is also a valid answer.
```

Example 2:

```
Input: s = "cbbd"
Output: "bb"
```

For example:

Test	Input	Result
ob1.longestPalindrome(str1)	ABCBCB	BCBCB

Answer: (penalty regime: 0 %)

Reset answer

```
class Solution(object):
    def longestPalindrome(self, s):
        dp = [[False for i in range(len(s))] for i in range(len(s))]
        for i in range(len(s)):
            dp[i][i] = True
        max_length = 1
        start = 0
        for l in range(2,len(s)+1):
            for i in range(len(s)-l+1):
                end = i+l

                if l==2:
                    if s[i] == s[end-1]:
                        dp[i][end-1]=True
                        max_length = l
                        start = i
                else:
                    if s[i] == s[end-1] and dp[i+1][end-2]:
                        dp[i][end-1]=True
                        max_length = l
                        start = i
        return s[start:start+max_length]
```

	Test	Input	Expected	Got	
✓	ob1.longestPalindrome(str1)	ABCBCB	BCBCB	BCBCB	✓
✓	ob1.longestPalindrome(str1)	BABAD	ABA	ABA	✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.