Started on	Monday, 12 May 2025, 10:17 AM
State	Finished
Completed on	Monday, 12 May 2025, 11:09 AM
Time taken	51 mins 24 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 common subsequence using Memoization Implementation.

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	✔
✔	SAMPLE SAEMSUNG	Length of LCS is 3	Length of LCS is 3	✔
✔	saveetha sabeetha	Length of LCS is 7	Length of LCS is 7	✔

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 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

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

```
def lcw(X,Y):
    m=len(X)
    n=len(Y)
    maxL=0
    eindex=m
    c=[[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]:
                c[i][j]=c[i-1][j-1]+1
                if c[i][j]>maxL:
                    maxL=c[i][j]
                    eindex=i
    return X[eindex-maxL: eindex]
u=input()
v=input()
print("Longest Common Subword:",lcw(u,v))
```

	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 **3**

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

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

```
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 **4**

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 %)

Ace editor not ready. Perhaps reload page?

Falling back to raw text area.

```
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 **5**

Incorrect

Mark 0.00 out of 20.00

Write a python program to implement knight tour problem using warnsdorff's algorithm

For example:

Test	Input	Result
a.warnsdroff((x,y))	8 8 3 3	board: [21, 32, 17, 30, 39, 36, 15, 42] [18, 29, 20, 35, 16, 41, 54, 37] [33, 22, 31, 40, 53, 38, 43, 14] [28, 19, 34, 1, 44, 49, 60, 55] [23, 2, 27, 52, 61, 56, 13, 50] [8, 5, 24, 45, 48, 51, 62, 59] [3, 26, 7, 10, 57, 64, 47, 12] [6, 9, 4, 25, 46, 11, 58, 63]

Answer: (penalty regime: 0 %)

Reset answer

Ace editor not ready. Perhaps reload page?

Falling back to raw text area.

```
KNIGHT_MOVES = [(2, 1), (1, 2), (-1, 2), (-2, 1), (-2, -1), (-1, -2), (1, -2), (2, -1)]
class KnightTour:
    def __init__(self, board_size):
        self.board_size = board_size  # tuple
        self.board = []
        for i in range(board_size[0]):
            temp = []
            for j in range(board_size[1]):
                temp.append(0)
            self.board.append(temp) # empty cell
        self.move = 1

    def print_board(self):
        print('board:')
        for i in range(self.board_size[0]):
            print(self.board[i])

    def warnsdroff(self, start_pos, GUI=False):
```

Syntax Error(s)

```
Sorry: IndentationError: expected an indented block (__tester__.python3, line 21)
```

Incorrect

Marks for this submission: 0.00/20.00.