Started on	Saturday, 21 September 2024, 3:01 PM
State	Finished
Completed on	Saturday, 21 September 2024, 3:15 PM
Time taken	14 mins 27 secs
Grade	**100.00** out of 100.00

Question **1**

Correct

Mark 20.00 out of 20.00

Create a python program to find the Hamiltonian path using Depth First Search for traversing the graph .

For example:

Test	Result
`hamiltonian.findCycle()`	`['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'A']` `['A', 'H', 'G', 'F', 'E', 'D', 'C', 'B', 'A']`

Answer: (penalty regime: 0 %)

Reset answer

```
class Hamiltonian:
    def __init__(self, start):
        self.start = start
        self.cycle = []
        self.hasCycle = False

    def findCycle(self):
        self.cycle.append(self.start)
        self.solve(self.start)

    def solve(self, vertex):
        ############  Add your code here ##############
        #Start here
        if vertex == self.start and len(self.cycle) == N+1:
            self.hasCycle = True
            self.displayCycle()
            return
        for i in range(len(vertices)):
            if adjacencyM[vertex][i] == 1 and visited[i] == 0:
                nbr = i
                visited[nbr] = 1
                self.cycle.append(nbr)
```

	Test	Expected	Got	
✔	`hamiltonian.findCycle()`	`['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'A']` `['A', 'H', 'G', 'F', 'E', 'D', 'C', 'B', 'A']`	`['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'A']` `['A', 'H', 'G', 'F', 'E', 'D', 'C', 'B', 'A']`	✔

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

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 %)

```
def lcs(str1 , str2):
    m = len(str1)
    n = len(str2)
    matrix = [[0]*(n+1) for i in range(m+1)]
    for i in range(m+1):
        for j in range(n+1):
            if i==0 or j==0:
                matrix[i][j] = 0
            elif str1[i-1] == str2[j-1]:
                matrix[i][j] = 1 + matrix[i-1][j-1]
            else:
                matrix[i][j] = max(matrix[i-1][j] , matrix[i][j-1])
    return matrix[-1][-1]
str1 = input()
str2 = input()
lcs_length = lcs(str1, str2)
print("Length of LCS is {}".format(lcs_length))
```

	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 **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

```
def lcw(X,Y):
    m = len(X)
    n = len(Y)
    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]

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 **5**

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

```
def printSubStr(str, low, high):
    for i in range(low, high + 1):
        print(str[i], end = "")
def longestPalindrome(str):
    n = len(str)
    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 (str[i + k] != str[j - k]):
                    flag = 0
            if (flag != 0 and (j - i + 1) > maxLength):
                start = i
                maxLength = j - i + 1
    printSubStr(str, start, start + maxLength - 1)

str = input() #"mojologiccigolmojo"
longestPalindrome(str)
```

	Input	Expected	Got	
✔	mojologiccigolmojo	logiccigol	logiccigol	✔
✔	sampleelpams	pleelp	pleelp	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.