Started on	Tuesday, 29 April 2025, 8:19 AM
State	Finished
Completed on	Tuesday, 29 April 2025, 9:06 AM
Time taken	47 mins 18 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 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
```

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

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **2**

Incorrect

Mark 0.00 out of 20.00

Write a python program to implement knight tour problem

For example:

Input	Result
5 5	[1, 12, 25, 18, 3] [22, 17, 2, 13, 24] [11, 8, 23, 4, 19] [16, 21, 6, 9, 14] [7, 10, 15, 20, 5] [(0, 0), (1, 2), (0, 4), (2, 3), (4, 4), (3, 2), (4, 0), (2, 1), (3, 3), (4, 1), (2, 0), (0, 1), (1, 3), (3, 4), (4, 2), (3, 0), (1, 1), (0, 3), (2, 4), (4, 3), (3, 1), (1, 0), (2, 2), (1, 4), (0, 2)] Done!

Answer: (penalty regime: 0 %)

Reset answer

```
import sys
class KnightsTour:
    def __init__(self, width, height):
        self.w = width
        self.h = height
        self.board = []
        self.generate_board()

    def generate_board(self):
        for i in range(self.h):
            self.board.append([0]*self.w)

    def print_board(self):

        for elem in self.board:
            print (elem)

    def generate_legal_moves(self, cur_pos):
        possible_pos = []
        move_offsets = [(1, 2), (1, -2), (-1, 2), (-1, -2),
                        (2, 1), (2, -1), (-2, 1), (-2, -1)]

```

	Input	Expected	Got	
✗	5 5	[1, 12, 25, 18, 3] [22, 17, 2, 13, 24] [11, 8, 23, 4, 19] [16, 21, 6, 9, 14] [7, 10, 15, 20, 5] [(0, 0), (1, 2), (0, 4), (2, 3), (4, 4), (3, 2), (4, 0), (2, 1), (3, 3), (4, 1), (2, 0), (0, 1), (1, 3), (3, 4), (4, 2), (3, 0), (1, 1), (0, 3), (2, 4), (4, 3), (3, 1), (1, 0), (2, 2), (1, 4), (0, 2)] Done!	[0, 0, 0, 0, 0] [0, 0, 0, 0, 0] [0, 0, 0, 0, 0] [0, 0, 0, 0, 0] [0, 0, 0, 0, 0]	✗

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 **3**

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 lcw(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",lcw(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.

Question **4**

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

```
def edit_distance(str1, str2, a, b):
    string_matrix = [[0 for i in range(b+1)] for i in range(a+1)]
    for i in range(a+1):
        for j in range(b+1):
            if i == 0:
                string_matrix[i][j] = j
            elif j == 0:
                string_matrix[i][j] = i
            elif str1[i-1] == str2[j-1]:
                string_matrix[i][j] = string_matrix[i-1][j-1]
            else:
                string_matrix[i][j] = 1 + min(string_matrix[i][j-1],strin
    return string_matrix[a][b]
if __name__ == '__main__':
    str1 = input()
    str2 = input()
    print('No. of Operations required :',edit_distance(str1, str2, len(st
```

	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 **5**

Correct

Mark 20.00 out of 20.00

Create a python program to find the length of longest common subsequence using naive recursive method

For example:

Input	Result
AGGTAB GXTXAYB	Length of LCS is 4

Answer: (penalty regime: 0 %)

```
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()#"AGGTAB"
Y = input()#"GXTXAYB"
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	✔
✔	saveetha engineering	Length of LCS is 2	Length of LCS is 2	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.