Started on	Wednesday, 16 April 2025, 11:51 AM
State	Finished
Completed on	Wednesday, 16 April 2025, 1:54 PM
Time taken	2 hours 3 mins
Overdue	3 mins 31 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

```
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 # Min. operations = j

            elif j == 0:
                dp[i][j] = i # Min. operations = 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

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(x,y,m,n,dp):
    if m==0 or n==0:
        return 0
    if dp[m][n]!=-1:
        return dp[m][n]
    if x[m-1]==y[n-1]:
        dp[m][n]=1+lcs(x,y,m-1,n-1,dp)
        return dp[m][n]
    dp[m][n]=max(lcs(x,y,m,n-1,dp),lcs(x,y,m-1,n,dp))
    return  dp[m][n]
x=input()
y=input()
dp=[[-1]*(len(y)+1) for i in range(len(x)+1)]
print("Length of LCS is",lcs(x,y,len(x),len(y),dp))
```

	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 **3**

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)
    max_len=0
    start=0
    for i in range(n):
        for j in range(1,n):
            s=str[i:j+1]
            if s==s[::-1]:
                cur=j-i+1
                if cur>max_len:
                    max_len=cur
                    start=i
    printSubStr(str, start, start + max_len - 1)

if __name__ == '__main__':

    str = input()
```

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

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **4**

Not answered

Mark 0.00 out of 20.00

Create a Python program to find longest common substring or subword (LCW) of two strings using dynamic programming with bottom-up approach.

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)	bisect trisect	Longest Common Subword: isect

Answer: (penalty regime: 0 %)

Reset answer

```
def lcw(u, v):
    ############# Add your code here ##############

    return
u = input()
v = input()
length_lcw, lcw_i, lcw_j = lcw(u, v)
print('Longest Common Subword: ', end='')
if length_lcw > 0:
    print(u[lcw_i:lcw_i + length_lcw])
```

Question **5**

Correct

Mark 20.00 out of 20.00

Write a Python program for Bad Character Heuristic of Boyer Moore String Matching Algorithm

For example:

Input	Result
ABAAAABCD ABC	Pattern occur at shift = 5

Answer: (penalty regime: 0 %)

Reset answer

```
NO_OF_CHARS = 256
def badCharHeuristic(string, size):
    ########### Add your Code Here ###############
    badChar = [-1] * NO_OF_CHARS
    for i in range(size):
        badChar[ord(string[i])] = i
    return badChar
def search(txt, pat):
    m = len(pat)
    n = len(txt)
    badChar = badCharHeuristic(pat, m)
    s = 0
    while(s <= n-m):
        j = m-1
        while j>=0 and pat[j] == txt[s+j]:
            j -= 1
        if j<0:
            print("Pattern occur at shift = {}".format(s))
            s += (m-badChar[ord(txt[s+m])] if s+m<n else 1)
        else:
            s += max(1, j-badChar[ord(txt[s+j])])
def main():
```

	Input	Expected	Got	
✔	ABAAAABCD ABC	Pattern occur at shift = 5	Pattern occur at shift = 5	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.