| | |
|---|---|
| **Started on** | Tuesday, 29 April 2025, 1:47 PM |
| **State** | Finished |
| **Completed on** | Tuesday, 29 April 2025, 5:57 PM |
| **Time taken** | 4 hours 10 mins |
| **Overdue** | 2 hours 10 mins |
| **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<br>GXTXAYB | Length of LCS is 4 |

**Answer:** (penalty regime: 0 %)

```
def lcs(X, Y, m, n):
    if (m == 0 or n == 0):
        return 0
    if (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<br>GXTXAYB | Length of LCS is 4 | Length of LCS is 4 | ✔ |
| ✔ | SAMPLE<br>SAEMSUNG | Length of LCS is 3 | Length of LCS is 3 | ✔ |
| ✔ | saveetha<br>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 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<br>trisect | Longest Common Subword: isect |

**Answer:** (penalty regime: 0 %)

Reset answer

```
def lcw(u, v):
    m = len(u)
    n = len(v)
    dp = [[0] * (n + 1) for _ in range(m + 1)]
    length_lcw = 0
    lcw_i = 0
    for i in range(1, m + 1):
        for j in range(1, n + 1):
            if u[i - 1] == v[j - 1]:
                dp[i][j] = dp[i - 1][j - 1] + 1
                if dp[i][j] > length_lcw:
                    length_lcw = dp[i][j]
                    lcw_i = i - length_lcw
    return length_lcw, lcw_i

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

| | Test | Input | Expected | Got | |
|---|---|---|---|---|---|
| ✔ | lcw(u, v) | bisect<br>trisect | Longest Common Subword: isect | Longest Common Subword: isect | ✔ |
| ✔ | lcw(u, v) | director<br>conductor | Longest Common Subword: ctor | Longest Common Subword: ctor | ✔ |

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **3**

Incorrect

Mark 0.00 out of 20.00

**SUBSET SUM PROBLEM**

**We are given a list of n numbers and a number x, the task is to write a python program to find out all possible subsets of the list such that their sum is x.**

**Examples:**

***Input:*** *arr = [2, 4, 5, 9], x = 15*

***Output:*** *[2, 4, 9]*

*15 can be obtained by adding 2, 4 and 9 from the given list.*

***Input :*** *arr = [10, 20, 25, 50, 70, 90], x = 80*

***Output :*** *[10, 70]*

*[10, 20, 50]*

*80 can be obtained by adding 10 and 70 or by adding 10, 20 and 50 from the given list.*

**THE INPUT**

**1.No of numbers**

**2.Get the numbers**

**3.Sum Value**

**For example:**

| Input | Result |
|---|---|
| 4<br>2<br>4<br>5<br>9<br>15 | [2, 4, 9] |
| 5<br>4<br>16<br>5<br>23<br>12<br>9 | [4, 5] |

**Answer:** (penalty regime: 0 %)

Reset answer

```
# Write your code here
n=int(input())
arr=[]
```

```
arr = []
for i in range(0,n):
    a=int(input())
    arr.append(a)
x = int(input())

subsetSum(n, arr, x)
```

| | Input | Expected | Got | |
|---|---|---|---|---|
| ✖ | 4<br>2<br>4<br>5<br>9<br>15 | [2, 4, 9] | ***Run error***<br>Traceback (most recent call last):<br>  File "__tester__.python3", line 22, in <module><br>  subsetSum(n, arr, x)<br>NameError: name 'subsetSum' is not defined | ✖ |

Testing was aborted due to error.
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 Naive recursive python program to find the minimum number of operations to convert str1 to str2

**For example:**

| Input | Result |
|---|---|
| Python<br>Peithen | Edit Distance 3 |

**Answer:** (penalty regime: 0 %)

Reset answer

```
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('Edit Distance',LD(str1,str2))
```

| | Input | Expected | Got | |
|---|---|---|---|---|
| ✔ | Python<br>Peithen | Edit Distance 3 | Edit Distance 3 | ✔ |
| ✔ | food<br>money | Edit Distance 4 | Edit Distance 4 | ✔ |

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():
    def longestPalindrome(self, s):
        n = len(s)
        dp = [[0]*n]*n
        for i in range(n):
            dp[i][i] = 1
        max_length = 1
        start = 0
        for length in range(2, n + 1):  # length of the substring
            for i in range(n - length + 1):
                end = i + length - 1  # end index of the substring
                if length == 2:
                    if s[i] == s[end]:
                        dp[i][end] = 1
                        max_length = length
                        start = i
                else:
                    if s[i] == s[end] and dp[i + 1][end - 1]:
                        dp[i][end] = 1
                        max_length = length
                        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.