Started on	Saturday, 10 May 2025, 10:24 AM
State	Finished
Completed on	Saturday, 10 May 2025, 2:37 PM
Time taken	4 hours 13 mins
Overdue	2 hours 13 mins
Grade	**100.00** out of 100.00

Question **1**

Correct

Mark 20.00 out of 20.00

To Write a Python Program to find longest common subsequence using Dynamic Programming

For example:

Input	Result
abcbdab bdcaba	bdab

Answer: (penalty regime: 0 %)

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

    dp = [[0] * (n + 1) for _ 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]:
                dp[i][j] = dp[i - 1][j - 1] + 1
            else:
                dp[i][j] = max(dp[i - 1][j], dp[i][j - 1])

    lcs_length = dp[m][n]
    lcs = [''] * lcs_length
    i, j = m, n

    while i > 0 and j > 0:
        if X[i - 1] == Y[j - 1]:
            lcs[lcs_length - 1] = X[i - 1]
            i -= 1
            j -= 1
```

	Input	Expected	Got	
✔	abcbdab bdcaba	bdab	bdab	✔
✔	treehouse elephant	eeh	eeh	✔

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 for the following problem statement.

You are given an `n x n grid` representing a field of cherries, each cell is one of three possible integers.

- `0` means the cell is empty, so you can pass through,
- `1` means the cell contains a cherry that you can pick up and pass through, or
- `-1` means the cell contains a thorn that blocks your way.

Return *the maximum number of cherries you can collect by following the rules below*:

- Starting at the position `(0, 0)` and reaching `(n - 1, n - 1)` by moving right or down through valid path cells (cells with value `0` or `1`).
- After reaching `(n - 1, n - 1)`, returning to `(0, 0)` by moving left or up through valid path cells.
- When passing through a path cell containing a cherry, you pick it up, and the cell becomes an empty cell `0`.
- If there is no valid path between `(0, 0)` and `(n - 1, n - 1)`, then no cherries can be collected.

For example:

Test	Result
obj.cherryPickup(grid)	5

Answer: (penalty regime: 0 %)

Reset answer

```
class Solution:
    def cherryPickup(self, grid):
        n = len(grid)
        dp = [[0]*n for _ in range(n)]
        for i in range(n-1,-1,-1):
            for j in range(n-1, -1, -1):
                if i==n-1 and j==n-1:
                    dp[i][j] = grid[i][j]
                elif i==n-1:
                    dp[i][j] = grid[i][j]+dp[i][j+1]
                elif j==n-1:
                    dp[i][j] = grid[i][j]+dp[i+1][j]
                else:
                    dp[i][j] = grid[i][j]+max(dp[i][j+1], dp[i+1][j])

        return max(0,dp[0][0])+1
obj=Solution()
grid=[[0,1,-1],[1,0,-1],[1,1,1]]
print(obj.cherryPickup(grid))
```

	Test	Expected	Got	
✔	obj.cherryPickup(grid)	5	5	✔

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 using brute force method of searching for the given substring in the main string.

For example:

Test	Input	Result
match(str1,str2)	AABAACAADAABAABA AABA	Found at index 0 Found at index 9 Found at index 12

Answer: (penalty regime: 0 %)

Reset answer

```
import re
def match(string,sub):
    pattern=re.compile(str2)
    r=pattern.search(str1)
    while r:
        print("Found at index {}".format(r.start()))
        r=pattern.search(str1,r.start()+1)
str1=input()
str2=input()
```

	Test	Input	Expected	Got	
✔	match(str1,str2)	AABAACAADAABAABA AABA	Found at index 0 Found at index 9 Found at index 12	Found at index 0 Found at index 9 Found at index 12	✔
✔	match(str1,str2)	saveetha savee	Found at index 0	Found at index 0	✔

Passed all tests! ✔

Correct

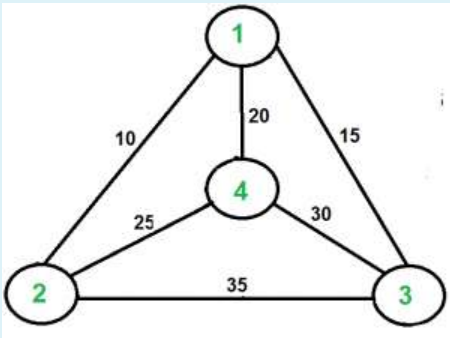
Marks for this submission: 20.00/20.00.

Question **4**

Correct

Mark 20.00 out of 20.00

Solve Travelling Sales man Problem for the following graph



Answer: (penalty regime: 0 %)

Reset answer

```
from sys import maxsize
from itertools import permutations
V = 4
def travellingSalesmanProblem(graph, s):
    vertex =[]
    for i in range(V):
        if i !=s:
            vertex.append(i)
    min_path = maxsize
    next_permutation = permutations(vertex)
    for i in next_permutation:
        current_pathweight = 0
        k = s
        for j in i:
            current_pathweight += graph[k][j]
            k = j
        current_pathweight += graph[k][s]
        min_path = min(min_path, current_pathweight)

    return min_path


```

	Expected	Got	
✔	80	80	✔

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 for 0/1 knapsack problem using naive recursion method

For example:

Test	Input	Result
knapSack(W, wt, val, n)	3 3 50 60 100 120 10 20 30	The maximum value that can be put in a knapsack of capacity W is: 220

Answer: (penalty regime: 0 %)

Reset answer

```
def knapSack(W, wt, val, n):
    if n==0 or W==0:
        return 0
    if(wt[n-1] > W):
        return knapSack(W, wt, val, n-1)
    else:
        return max(val[n-1]+knapSack(W-wt[n-1], wt, val, n-1), knapSack(W, wt, val, n-1))

x=int(input())
y=int(input())
W=int(input())
val=[]
wt=[]
for i in range(x):
    val.append(int(input()))
for y in range(y):
    wt.append(int(input()))
n = len(val)
print('The maximum value that can be put in a knapsack of capacity W is: ',knapSack(W, wt, val, n))
```

	Test	Input	Expected	Got	
✔	knapSack(W, wt, val, n)	3 3 50 60 100 120 10 20 30	The maximum value that can be put in a knapsack of capacity W is: 220	The maximum value that can be put in a knapsack of capacity W is: 220	✔

	Test	Input	Expected	Got	
✔	knapSack(W, wt, val, n)	3 3 55 65 115 125 15 25 35	The maximum value that can be put in a knapsack of capacity W is: 190	The maximum value that can be put in a knapsack of capacity W is: 190	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.