Started on	Thursday, 15 May 2025, 2:25 PM
State	Finished
Completed on	Sunday, 18 May 2025, 6:54 PM
Time taken	3 days 4 hours
Overdue	3 days 2 hours
Grade	**100.00** out of 100.00

Question **1**

Correct

Mark 20.00 out of 20.00

Flag question

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 #Import this package
def match(str1,str2):

    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 **2**

Correct

Mark 20.00 out of 20.00

Flag question

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):
    #start
    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())
```

```
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
	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

Passed all tests!

Correct

Marks for this submission: 20.00/20.00.

Question **3**

Correct

Mark 20.00 out of 20.00

Flag question

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(object):
    def cherryPickup(self, grid):
        dp=[[0 for j in range(len(grid))]for i in range(len(grid))]
        for i in range(len(grid)):
            for j in range(len(grid)-1):
                dp[i][j]=grid[i-1][j-1]
        res=len(grid)+1


        ROW_NUM = len(grid)
        COL_NUM = len(grid[0])
        return dp[0][COL_NUM - 1]*res

grid=[[3,1,1],
      [2,5,1],
      [1,5,5],
      [2,1,1]]
obj=Solution()
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 4

Correct

Mark 20.00 out of 20.00

Flag question

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 5

Correct

Mark 20.00 out of 20.00

Flag question

Given a 2D matrix **tsp[][]**, where each row has the array of distances from that indexed city to all the other cities and **-1** denotes that there doesn't exist a path between those two indexed cities. The task is to print minimum cost in TSP cycle.

tsp[][] = {{-1, 30, 25, 10},
{15, -1, 20, 40},
{10, 20, -1, 25},
{30, 10, 20, -1}};

Answer: (penalty regime: 0 %)

Reset answer

```
from typing import DefaultDict
INT_MAX = 2147483647
def findMinRoute(tsp):
    sum = 0
    counter = 0
    j = 0
    i = 0
    min = INT_MAX
    visitedRouteList = DefaultDict(int)
    visitedRouteList[0] = 1
    route = [0] * len(tsp)
    while i < len(tsp) and j < len(tsp[i]):
        #Write your code here
        #Start here
```

```
        if counter >= len(tsp[i]) - 1:
            break
        if j != i and (visitedRouteList[j] == 0):
            if tsp[i][j] < min:
                min = tsp[i][j]
                route[counter] = j + 1
        j += 1
        if j == len(tsp[i]):
```

	Expected	Got	
	`Minimum Cost is : 50`	`Minimum Cost is : 50`	

Passed all tests!

Correct

Marks for this submission: 20.00/20.00.