Started on	Thursday, 15 May 2025, 9:19 AM
State	Finished
Completed on	Thursday, 15 May 2025, 11:22 AM
Time taken	2 hours 3 mins
Overdue	3 mins 26 secs
Grade	**80.00** out of 100.00

Question **1**

Correct

Mark 20.00 out of 20.00

Create a Dynamic Programming python Implementation of Coin Change Problem.

For example:

Test	Input	Result
`count(arr, m, n)`	3 4 1 2 3	4

Answer: (penalty regime: 0 %)

Reset answer

```python
def count(S, m, n):
    table = [[0 for x in range(m)] for x in range(n+1)]
    for i in range(m):
        table[0][i] = 1
    for i in range(1, n+1):
        for j in range(m):
            # Count of solutions including S[j]
            #Start here
            x = table[i - S[j]][j] if i-S[j] >= 0 else 0
            # Count of solutions excluding S[j]
            y = table[i][j-1] if j >= 1 else 0
            # total count
            table[i][j] = x + y
    return table[n][m-1]
    #End here
arr = []
m = int(input())
n = int(input())
for i in range(m):
    arr.append(int(input()))
print(count(arr, m, n))

```

	Test	Input	Expected	Got	
✓	`count(arr, m, n)`	3 4 1 2 3	4	4	✓
✓	`count(arr, m, n)`	3 16 1 2 5	20	20	✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.

Question **2**

Correct

Mark 20.00 out of 20.00

Print All Paths With Minimum Jumps

```
1. You are given a number N representing number of elements.
2. You are given N space separated numbers (ELE : elements).
3. Your task is to find & print
    3.1) "MINIMUM JUMPS" need from 0th step to (n-1)th step.
    3.2) all configurations of "MINIMUM JUMPS".
NOTE: Checkout sample question/solution video inorder to have more insight.
```

For example:

Test	Input	Result
minJumps(arr)	10 3 3 0 2 1 2 4 2 0 0	0 -> 3 -> 5 -> 6 -> 9 0 -> 3 -> 5 -> 7 -> 9

Answer: (penalty regime: 0 %)

Reset answer

```
from queue import Queue
import sys
class Pair(object):
    idx = 0
    psf = ""
    jmps = 0
    def __init__(self, idx, psf, jmps):
        
        self.idx = idx
        self.psf = psf
        self.jmps = jmps
def minJumps(arr):
    MAX_VALUE = sys.maxsize
    dp = [MAX_VALUE for i in range(len(arr))]
    n = len(dp)
    dp[n - 1] = 0
    for i in range(n - 2, -1, -1):
        steps = arr[i]
        minimum = MAX_VALUE
        for j in range(1, steps + 1, 1):
            if i + j >= n:
                break
```

	Test	Input	Expected	Got	
✓	minJumps(arr)	10 3 3 0 2 1 2 4 2 0 0	0 -> 3 -> 5 -> 6 -> 9 0 -> 3 -> 5 -> 7 -> 9	0 -> 3 -> 5 -> 6 -> 9 0 -> 3 -> 5 -> 7 -> 9	✓

	Test	Input	Expected	Got	
✓	`minJumps(arr)`	7 5 5 0 3 2 3 6	0 -> 1 -> 6 0 -> 3 -> 6 0 -> 4 -> 6 0 -> 5 -> 6	0 -> 1 -> 6 0 -> 3 -> 6 0 -> 4 -> 6 0 -> 5 -> 6	✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.

Question **3**

Correct

Mark 20.00 out of 20.00

Write a python program to find the maximum contiguous subarray.

For example:

Test	Input	Result
`maxSubArraySum(a,n)`	8 -2 -3 4 -1 -2 1 5 -3	`Maximum contiguous sum is 7`

Answer: (penalty regime: 0 %)

Reset answer

```python
def maxSubArraySum(a,size):
    max_so_far = a[0]
    max_ending_here = 0
    for i in range(0, size):
        max_ending_here = max_ending_here + a[i]
        if max_ending_here < 0:
            max_ending_here = 0
        elif (max_so_far < max_ending_here):
            max_so_far = max_ending_here

    return max_so_far
    #End here
n=int(input())
a =[] #[-2, -3, 4, -1, -2, 1, 5, -3]
for i in range(n):
    a.append(int(input()))
print("Maximum contiguous sum is", maxSubArraySum(a,n))
```

	Test	Input	Expected	Got	
✓	`maxSubArraySum(a,n)`	8 -2 -3 4 -1 -2 1 5 -3	`Maximum contiguous sum is 7`	`Maximum contiguous sum is 7`	✓
✓	`maxSubArraySum(a,n)`	5 1 -2 -3 4 5	`Maximum contiguous sum is 9`	`Maximum contiguous sum is 9`	✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.

Question **4**

Correct

Mark 20.00 out of 20.00

Write a Python program using A Naive recursive implementation of Minimum Cost Path Problem.

For example:

Input	Result
3 3	8

Answer: (penalty regime: 0 %)

Reset answer

```
R = int(input())
C = int(input())
def minCost(cost, m, n):
    tc = [[0 for x in range(C)] for x in range(R)]
    tc[0][0] = cost[0][0]
    for i in range(1, m+1):
        tc[i][0] = tc[i-1][0] + cost[i][0]
    for j in range(1, n+1):
        tc[0][j] = tc[0][j-1] + cost[0][j]
    for i in range(1, m+1):
        for j in range(1, n+1):
            tc[i][j] = min(tc[i-1][j-1], tc[i-1][j], tc[i][j-1]) + cost[i][j]

    return tc[m][n]

cost = [[1, 2, 3],
        [4, 8, 2],
        [1, 5, 3]]
print(minCost(cost, R-1, C-1))
```

	Input	Expected	Got	
✓	3 3	8	8	✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.

Question **5**

Not answered

Mark 0.00 out of 20.00

Write a Python Program to calculate the GCD of the given two numbers using Recursive function

For example:

Input	Result
49 35	7
25 90	5

Answer: (penalty regime: 0 %)