Started on	Wednesday, 14 May 2025, 11:40 AM
State	Finished
Completed on	Wednesday, 21 May 2025, 2:41 PM
Time taken	7 days 3 hours
Overdue	7 days 1 hour
Grade	**100.00** out of 100.00

Question **1**

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 **2**

Correct

Mark 20.00 out of 20.00

Create a python program to find the minimum number of jumps needed to reach end of the array using Dynamic Programming.

For example:

Test	Input	Result
minJumps(arr,n)	6 1 3 6 1 0 9	Minimum number of jumps to reach end is 3

Answer: (penalty regime: 0 %)

Reset answer

```
def minJumps(arr, n):
    ##########  Add your code here ###############
    #Start here
    jumps = [0 for i in range(n)]
    if (n == 0) or (arr[0] == 0):
        return float('inf')
    jumps[0] = 0
    for i in range(1, n):
        jumps[i] = float('inf')
        for j in range(i):
            if (i <= j + arr[j]) and (jumps[j] != float('inf')):
                jumps[i] = min(jumps[i], jumps[j] + 1)
                break
    return jumps[n-1]
    #End here
arr = []
n = int(input()) #len(arr)
for i in range(n):
    arr.append(int(input()))
print('Minimum number of jumps to reach','end is', minJumps(arr,n))
```

	Test	Input	Expected	Got	
✔	minJumps(arr,n)	6 1 3 6 1 0 9	Minimum number of jumps to reach end is 3	Minimum number of jumps to reach end is 3	✔
✔	minJumps(arr,n)	7 2 3 -8 9 5 6 4	Minimum number of jumps to reach end is 3	Minimum number of jumps to reach end is 3	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **3**

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

```
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 **4**

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)
    maxLength = 1
    start = 0
    for i in range(n):
        for j in range(i, n):
            flag = 1
            for k in range(0, ((j - i) // 2) + 1):
                if (str[i + k] != str[j - k]):
                    flag = 0
            if (flag != 0 and (j - i + 1) > maxLength):
                start = i
                maxLength = j - i + 1
    printSubStr(str, start, start + maxLength - 1)

str = input()
longestPalindrome(str)

```

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

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **5**

Correct

Mark 20.00 out of 20.00

Given an integer array `nums`, find the contiguous subarray (containing at least one number) which has the largest sum and return *its sum*.

A **subarray** is a **contiguous** part of an array.

Example 1:

```
Input: nums = [-2,1,-3,4,-1,2,1,-5,4]
Output: 6
Explanation: [4,-1,2,1] has the largest sum = 6.
```

For example:

Test	Input	Result
s.maxSubArray(A)	9 -2 1 -3 4 -1 2 1 -5 4	The sum of contiguous sublist with the largest sum is 6

Answer: (penalty regime: 0 %)

Reset answer

```
class Solution:
    def maxSubArray(self,A):
        ############ Add your Code here
        #Start here
        res=0
        mm= -10000
        for v in A:
            res+=v
            mm=max(mm,res)
            if res<0:
                res=0
        return mm
        #End here
A =[]
n=int(input())
for i in range(n):
    A.append(int(input()))
s=Solution()
print("The sum of contiguous sublist with the largest sum is",s.maxSubArray(A))
```

	Test	Input	Expected	Got	
✔	s.maxSubArray(A)	9 -2 1 -3 4 -1 2 1 -5 4	The sum of contiguous sublist with the largest sum is 6	The sum of contiguous sublist with the largest sum is 6	✔
✔	s.maxSubArray(A)	5 5 4 -1 7 8	The sum of contiguous sublist with the largest sum is 23	The sum of contiguous sublist with the largest sum is 23	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.