Started on	Saturday, 17 May 2025, 8:14 AM
State	Finished
Completed on	Saturday, 17 May 2025, 8:34 AM
Time taken	19 mins 6 secs
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())
import sys
def minCost(cost, m, n):
    if (n < 0 or m < 0):
        return sys.maxsize
    elif (m == 0 and n == 0):
        return cost[m][n]
    else:
        return cost[m][n] + min( minCost(cost, m-1, n-1),
                                minCost(cost, m-1, n),
                                minCost(cost, m, n-1) )
def min(x, y, z):
    if (x < y):
        return x if (x < z) else z
    else:
        return y if (y < z) else z
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

LONGEST PALINDROMIC SUBSEQUENCE

Given a sequence, find the length of the longest palindromic subsequence in it.

For example:

Input	Result
ABBDCACB	The length of the LPS is 5

Answer: (penalty regime: 0 %)

```
def Lps(X):
    n=len(X)
    dp=[[0 for _ in range(n)] for _ in range(n)]
    for x in range(n):
        dp[x][x]=1

    for l in range(2,n+1):
        for i in range(n-l+1):
            j=i+l-1
            if X[i]==X[j]:
                dp[i][j]=dp[i+1][j-1]+2
            else:
                dp[i][j]=max(dp[i+1][j],dp[i][j-1])
    return dp[0][n-1]


X=input()
print("The length of the LPS is",Lps(X))
```

	Input	Expected	Got	
✓	ABBDCACB	The length of the LPS is 5	The length of the LPS is 5	✓
✓	BBABCBCAB	The length of the LPS is 7	The length of the LPS is 7	✓
✓	cbbd	The length of the LPS is 2	The length of the LPS is 2	✓
✓	abbab	The length of the LPS is 4	The length of the LPS is 4	✓

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 to find Minimum number of jumps to reach end of the array using naive method(recursion) using float values

For example:

Test	Input	Result
`minJumps(arr, 0, n-1)`	6 2.3 7.4 6.3 1.5 8.2 0.1	`Minimum number of jumps to reach end is 2`

Answer: (penalty regime: 0 %)

Reset answer

```
def minJumps(arr, l, h):
    if (h == l):
        return 0
    if (arr[l] == 0):
        return float('inf')
    min = float('inf')
    for i in range(l + 1, h + 1):
        if (i < l + arr[l] + 1):
            jumps = minJumps(arr, i, h)
            if (jumps != float('inf') and
                        jumps < min):
                min = jumps + 1
    return min
arr = []
n = int(input())
for i in range(n):
    arr.append(float(input()))
print('Minimum number of jumps to reach','end is', minJumps(arr, 0, n-1))
```

	Test	Input	Expected	Got	
✓	`minJumps(arr, 0, n-1)`	6 2.3 7.4 6.3 1.5 8.2 0.1	`Minimum number of jumps to reach end is 2`	`Minimum number of jumps to reach end is 2`	✓

	Test	Input	Expected	Got	
✓	`minJumps(arr, 0, n-1)`	10 3.2 3.2 5 6.2 4.9 1.2 5.0 7.3 4.6 6.2	Minimum number of jumps to reach end is 2	Minimum number of jumps to reach end is 2	✓

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 maximum contiguous sub array using Dynamic Programming.

For example:

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

Answer: (penalty regime: 0 %)

```python
def maxSubArraySum(a,size):
    max_till_now = a[0]
    max_ending = 0

    for i in range(0, size):
        max_ending = max_ending + a[i]
        if max_ending < 0:
            max_ending = 0


        elif (max_till_now < max_ending):
            max_till_now = max_ending

    return max_till_now
n=int(input())
a =[]
for i in range(n):
    a.append(int(input()))

print("Maximum contiguous sum is", maxSubArraySum(a,n))
```

	Test	Input	Expected	Got	
✓	`maxSubArraySum(a,len(a))`	8 -2 -3 4 -1 -2 1 5 -3	Maximum contiguous sum is 7	Maximum contiguous sum is 7	✓
✓	`maxSubArraySum(a,len(a))`	5 1 2 3 -4 -6	Maximum contiguous sum is 6	Maximum contiguous sum is 6	✓

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 Function to find the total number of distinct ways to get a change of 'target' from an unlimited supply of coins in set 'S'.

For example:

Test	Input	Result
`count(S, len(S) - 1, target)`	3 4 1 2 3	`The total number of ways to get the desired change is 4`

Answer: (penalty regime: 0 %)

Reset answer

```python
def count(S, n, target):
    if target == 0:
        return 1
    if target < 0 or n < 0:
        return 0
    incl = count(S, n, target - S[n])
    excl = count(S, n - 1, target)
    return incl + excl


if __name__ == '__main__':
    S = []
    n=int(input())
    target = int(input())
    for i in range(n):
        S.append(int(input()))
    print('The total number of ways to get the desired change is',
        count(S, len(S) - 1, target))



```

	Test	Input	Expected	Got	
✓	`count(S, len(S) - 1, target)`	3 4 1 2 3	The total number of ways to get the desired change is 4	The total number of ways to get the desired change is 4	✓
✓	`count(S, len(S) - 1, target)`	3 11 1 2 5	The total number of ways to get the desired change is 11	The total number of ways to get the desired change is 11	✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.