Started on	Tuesday, 8 July 2025, 1:41 PM
State	Finished
Completed on	Tuesday, 8 July 2025, 2:05 PM
Time taken	24 mins 2 secs
Grade	**80.00** out of 100.00

Question **1**

Correct

Mark 20.00 out of 20.00

Write a python program to calculate the length of the given string using recursion

For example:

Test	Input	Result
`length(str)`	`saveetha`	`length of saveetha is 8`
`length(str)`	`engineering`	`length of engineering is 11`

Answer: (penalty regime: 0 %)

```python
def length(str):
    if str=="":
        return 0
    return 1+length(str[1:])
str = input()
leng = length(str)
print("length of",str,"is",leng)
```

	Test	Input	Expected	Got	
✓	`length(str)`	`saveetha`	`length of saveetha is 8`	`length of saveetha is 8`	✓
✓	`length(str)`	`engineering`	`length of engineering is 11`	`length of engineering is 11`	✓
✓	`length(str)`	`Welcome`	`length of Welcome is 7`	`length of Welcome is 7`	✓

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.

Question **2**

Incorrect

Mark 0.00 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

```
def count(S, n, target):


    ###################  Add Your Code Here #################




if __name__ == '__main__':
    S = []#[1, 2, 3]
    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))
```

Syntax Error(s)

```
Sorry: IndentationError: expected an indented block (__tester__.python3, line 9)
```

Incorrect

Marks for this submission: 0.00/20.00.

Question **3**

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):
    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]

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 **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 =[] #[-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,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

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.