Started on	Saturday, 24 May 2025, 9:31 AM
State	Finished
Completed on	Saturday, 24 May 2025, 9:53 AM
Time taken	21 mins 48 secs
Grade	**100.00** out of 100.00

Question 1

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

```
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.

Question **2**

Correct

Mark 20.00 out of 20.00

Write a python program to find the maximum contiguous subarray on the given float array using kadane's algorithm.

For example:

Test	Input	Result
s.maxSubArray(A)	5 -9.6 -3.5 6.3 8.31 9.2	The sum of contiguous sublist with the largest sum is 23.8

Answer: (penalty regime: 0 %)

Reset answer

```
class Solution:
    def maxSubArray(a,size):
        max_sum = A[0]
        current_sum = A[0]
        for i in range(1, len(A)):
            current_sum = max(A[i], current_sum + A[i])
            max_sum = max(max_sum, current_sum)
        return max_sum

A =[]
n=int(input())
for i in range(n):
    A.append(float(input()))
s=Solution()
print("The sum of contiguous sublist with the largest sum is {:.1f}".format(s.maxSubArray(A)))
```

	Test	Input	Expected	Got	
✔	s.maxSubArray(A)	5 -9.6 -3.5 6.3 8.31 9.2	The sum of contiguous sublist with the largest sum is 23.8	The sum of contiguous sublist with the largest sum is 23.8	✔
✔	s.maxSubArray(A)	7 2.3 6.5 4.6 -7.8 -2.8 -1.6 9.8	The sum of contiguous sublist with the largest sum is 13.4	The sum of contiguous sublist with the largest sum is 13.4	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **3**

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):
    dp = [sys.maxsize] * len(arr)
    dp[-1] = 0
    for i in range(len(arr) - 2, -1, -1):
        steps = arr[i]
        min_jump = sys.maxsize
        for j in range(1, steps + 1):
            if i + j < len(arr):
                min_jump = min(min_jump, dp[i + j])
        if min_jump != sys.maxsize:
            dp[i]= min jump + 1
```

	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 **4**

Correct

Mark 20.00 out of 20.00

Write a python program to Implement Minimum cost path using Dynamic Programming.

For example:

Input	Result
3 3	8

Answer: (penalty regime: 0 %)

```
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, 2, 2))
```

	Input	Expected	Got	
✔	3 3	8	8	✔

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 to implement Quick sort using the first element as pivot value

For example:

Input	Result
5 61 24 3 50 8	Pivot: 61 Pivot: 8 Pivot: 24 Sorted array: [3, 8, 24, 50, 61]
6 2 3 54 10 28 94	Pivot: 2 Pivot: 3 Pivot: 54 Pivot: 28 Sorted array: [2, 3, 10, 28, 54, 94]

Answer: (penalty regime: 0 %)

```
def quick_sort(alist,start,end):
    if end-start>1:
        p=partition(alist,start,end)
        quick_sort(alist,start,p)
        quick_sort(alist,p+1,end)
def partition(alist,start,end):
    pivot=alist[start]
    i=start+1
    j=end-1
    print("Pivot: ",pivot)
    while True:
        while(i<=j and alist[i]<=pivot):
            i=i+1
        while(i<=j and alist[j]>=pivot):
            j=j-1
        if i<=j:
            alist[i],alist[j]=alist[j],alist[i]
        else:
            alist[start],alist[j]=alist[j],alist[start]
            return j
alist=[]
n=int(input())
```

	Input	Expected	Got	
✔	5 61 24 3 50 8	Pivot: 61 Pivot: 8 Pivot: 24 Sorted array: [3, 8, 24, 50, 61]	Pivot: 61 Pivot: 8 Pivot: 24 Sorted array: [3, 8, 24, 50, 61]	✔
✔	6 2 3 54 10 28 94	Pivot: 2 Pivot: 3 Pivot: 54 Pivot: 28 Sorted array: [2, 3, 10, 28, 54, 94]	Pivot: 2 Pivot: 3 Pivot: 54 Pivot: 28 Sorted array: [2, 3, 10, 28, 54, 94]	✔

	Input	Expected	Got	
✔	4 21 3 56 8	Pivot: 21 Pivot: 8 Sorted array: [3, 8, 21, 56]	Pivot: 21 Pivot: 8 Sorted array: [3, 8, 21, 56]	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.