Started on	Saturday, 10 May 2025, 10:33 AM
State	Finished
Completed on	Saturday, 10 May 2025, 11:13 AM
Time taken	40 mins 11 secs
Grade	**100.00** out of 100.00

Question **1**

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)

For example:

Test	Input	Result
minJumps(arr, 0, n-1)	10 1 3 6 3 2 3 6 8 9 5	Minimum number of jumps to reach end is 4

Answer: (penalty regime: 0 %)

Reset answer

```
def minJumps(arr, l, h):
    if(l==h):
        return 0
    elif(arr[l]==0):
        return 0
    min=float('inf')
    for i in range(l+1,h+1):
        if(i<l+arr[l]+1):
            jump=minJumps(arr,i,h)
        if(jump!=min and jump+1<min):
            min=jump+1
    return min


arr = []
n = int(input())
for i in range(n):
    arr.append(int(input()))
print('Minimum number of jumps to reach','end is', minJumps(arr, 0, n-1))

```

	Test	Input	Expected	Got	
✔	minJumps(arr, 0, n-1)	10 1 3 6 3 2 3 6 8 9 5	Minimum number of jumps to reach end is 4	Minimum number of jumps to reach end is 4	✔

	Test	Input	Expected	Got	
✔	minJumps(arr, 0, n-1)	7 3 2 5 9 4 1 6	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 **2**

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):
    dp=[[0 for x in range(R)]for x in range(C)]
    dp[0][0]=cost[0][0]
    for i in range(1,m+1):
        dp[i][0]=dp[i-1][0]+cost[i][0]
    for j in range(1,n+1):
        dp[0][j]=dp[0][j-1]+cost[0][j]
    for i in range(1,m+1):
        for j in range(1,n+1):
            dp[i][j]=min(dp[i-1][0],dp[j-1][0],dp[i-1][j-1])+cost[i][j]
    return dp[i][j]
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] ]
```

	Input	Expected	Got	
✔	3 3	8	8	✔

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 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


    ###################  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',
```

	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 **4**

Correct

Mark 20.00 out of 20.00

SUBSET SUM PROBLEM

Given a set of positive integers, and a value sum, determine that the sum of the subset of a given set is equal to the given sum.

Write the program for subset sum problem.

INPUT

1.no of elements

2.Input the given elements

3.Get the target sum

OUTPUT

True , if subset with required sum is found

False , if subset with required sum is not found

For example:

Input	Result
5 4 16 5 23 12 9	4 16 5 23 12 True,subset found

Answer: (penalty regime: 0 %)

Reset answer

```
def SubsetSum(a,i,sum,target,n):
    if(i==n):
        return sum==target
    elif(sum>target):
        return False
    elif(sum==target):
        return True
    else:
        return SubsetSum(a,i+1,sum,target,n) or SubsetSum(a,i+1,sum+a[i],target,n)

a=[]
size=int(input())
for i in range(size):
    x=int(input())
    a.append(x)

target=int(input())
n=len(a)
if(SubsetSum(a,0,0,target,n)==True):
    for i in range(size):
        print(a[i])
    print("True,subset found")
```

	Input	Expected	Got	
✔	5 4 16 5 23 12 9	4 16 5 23 12 True,subset found	4 16 5 23 12 True,subset found	✔
✔	4 1 2 3 4 11	1 2 3 4 False,subset not found	1 2 3 4 False,subset not found	✔
✔	7 10 7 5 18 12 20 15 35	10 7 5 18 12 20 15 True,subset found	10 7 5 18 12 20 15 True,subset found	✔

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 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(self,A):
        max=A[0]
        sum=0
        for i in range(0,n):
            sum=sum+A[i]
            if(sum<0):
                sum=0
            elif(sum>max):
                max=sum
        return max

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.