Started on	Monday, 19 May 2025, 1:58 PM
State	Finished
Completed on	Monday, 19 May 2025, 9:57 PM
Time taken	7 hours 59 mins
Overdue	5 hours 59 mins
Grade	**80.00** 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):
    ############   Add your code here ############
    #Start here
    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 + 1 < min):
                min = jumps + 1
    return min
    #End here
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

Create a python function to compute the fewest number of coins that we need to make up the amount given.

For example:

Test	Input	Result
ob1.coinChange(s,amt)	3 11 1 2 5	3

Answer: (penalty regime: 0 %)

Reset answer

```
class Solution(object):
    def coinChange(self, coins, amount):
        ####################      Add your Code Here ############
        #End here
        if amount == 0 :
            return 0
        if min(coins) > amount:
            return -1
        dp = [-1 for i in range(0, amount + 1)]
        for i in coins:
            if i > len(dp) - 1:
                continue
            dp[i] = 1
            for j in range(i + 1, amount + 1):
                if dp[j - i] == -1:
                    continue
                elif dp[j] == -1:
                    dp[j] = dp[j - i] + 1
                else:
                    dp[j] = min(dp[j], dp[j - i] + 1)
        return dp[amount]
    #End here
```

	Test	Input	Expected	Got	
✔	ob1.coinChange(s,amt)	3 11 1 2 5	3	3	✔
✔	ob1.coinChange(s,amt)	3 12 1 2 5	3	3	✔
✔	ob1.coinChange(s,amt)	3 22 1 2 5	5	5	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.

Question **3**

Not answered

Mark 0.00 out of 20.00

SUBSET SUM PROBLEM

COUNT OF SUBSETS WITH SUM EQUAL TO X

Given an array arr[] of length N and an integer X, the task is to find the number of subsets with a sum equal to X.

Examples:

Input: *arr[] = {1, 2, 3, 3}, X = 6*
Output: *3*
All the possible subsets are {1, 2, 3},
{1, 2, 3} and {3, 3}

Input: *arr[] = {1, 1, 1, 1}, X = 1*
Output: *4*

THE INPUT

1.No of numbers

2.Get the numbers

3.Sum Value

For example:

Input	Result
4 2 4 5 9 15	1
6 3 34 4 12 3 2 7	2

Answer: (penalty regime: 0 %)

Reset answer

```
def subsetSum(arr, n, i,sum, count):
#Write your code here









arr=[]
size=int(input())
for j in range(size):
```

```
    value=int(input())
    arr.append(value)
sum = int(input())
n = len(arr)

print(subsetSum(arr, n, 0, sum, 0))
```

Question **4**

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.

Question **5**

Correct

Mark 20.00 out of 20.00

Write a Python program to Implement Minimum cost path in a Directed Graph

For example:

Test	Result
`getMinPathSum(graph, visited, necessary, source, dest, 0);`	12

Answer: (penalty regime: 0 %)

Reset answer

```
minSum = 1000000000
def getMinPathSum(graph, visited, necessary,
                  src, dest, currSum):
    ################## Add your Code here ################
    #Start here
    global minSum
    if (src == dest):
        flag = True;
        for i in necessary:
            if (not visited[i]):
                flag = False;
                break;
        if (flag):
            minSum = min(minSum, currSum);
        return;

    else:
        visited[src] = True;
        for node in graph[src]:
            if not visited[node[0]]:
                visited[node[0]] = True;
                getMinPathSum(graph, visited,
```

	Test	Expected	Got	
✔	`getMinPathSum(graph, visited, necessary, source, dest, 0);`	12	12	✔

Passed all tests! ✔

Correct

Marks for this submission: 20.00/20.00.