Started on	Saturday, 3 May 2025, 8:17 AM
State	Finished
Completed on	Saturday, 3 May 2025, 8:50 AM
Time taken	33 mins 11 secs
Grade	**80.00** out of 100.00

Question **1**

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):
        res=0
        mm= -10000
        for v in A:
            res+=v
            mm=max(mm,res)
            if res<0:
                res=0
        return mm
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	✓

	Test	Input	Expected	Got	
✓	`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 **2**

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

```python
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 + 1 < min):
                min = jumps + 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 **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

```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.

Question **4**

Incorrect

Mark 0.00 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):

# Write your code here








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])
```

Syntax Error(s)

```
Sorry: IndentationError: expected an indented block (__tester__.python3, line 12)
```

Incorrect

Marks for this submission: 0.00/20.00.

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

```python
minSum = 1000000000
def getMinPathSum(graph, visited, necessary,src, dest, currSum):
    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
    for node in graph[src]:
        if not visited[node[0]]:
            visited[node[0]]=True
            getMinPathSum(graph,visited,necessary,node[0],dest,currSum+node[1])
            visited[node[0]]=False
    visited[src]=False
if __name__=='__main__':
    graph=dict()
    graph[0] = [ [ 1, 2 ], [ 2, 3 ], [ 3, 2 ] ];
    graph[1] = [ [ 4, 4 ], [ 0, 1 ] ];
```

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

Passed all tests! ✓

Correct

Marks for this submission: 20.00/20.00.