SEC

TT DS PYTHON MODULE-23

SEC

State	Finished
Completed on	Saturday, 10 May 2025, 4:04 PM
Time taken	46 mins 29 secs
Grade	**80.00** out of 100.00

Question **1**

Correct

Mark 20.00 out of 20.00

Flag question

Write a python program to find the maximum contiguous subarray.

For example:

Test	Input	Result
maxSubArraySum(a,n)	8 -2 -3 4 -1 -2 1 5 -3	Maximum contiguous sum is 7

Answer: (penalty regime: 0 %)

Reset answer

```
def maxSubArraySum(a,size):

    ####################  Add your Code here #############
    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,n)	8 -2 -3 4 -1 -2 1 5 -3	Maximum contiguous sum is 7	Maximum contiguous sum is 7	
	maxSubArraySum(a,n)	5 1 -2 -3 4 5	Maximum contiguous sum is 9	Maximum contiguous sum is 9	

Passed all tests!

Correct

Marks for this submission: 20.00/20.00.

Question **2**

Not answered

Mark 0.00 out of 20.00

Flag question

Write a Python program to sort unsorted numbers using Random Pivot Quick Sort. Picks the random index as the pivot

For example:

Test	Input	Result
quick_sort_random(nums, 0, len(nums))	5 1	Original list: [1, 2, 65, 4, 9]

SEC

quick_sort_random(nums, 0, len(nums))	6 32 10 5 6 4 8	Original list: [32, 10, 5, 6, 4, 8] After applying Random Pivot Quick Sort the said list becomes: [4, 5, 6, 8, 10, 32]

Answer: (penalty regime: 0 %)

```
1
```

Question **3**

Correct

Mark 20.00 out of 20.00

Flag question

Create a Dynamic Programming python Implementation of Coin Change Problem.

For example:

Test	Input	Result
count(arr, m, n)	3 4 1 2 3	4

Answer: (penalty regime: 0 %)

Reset answer

```
def count(S, m, n):
    table = [[0 for x in range(m)] for x in range(n+1)]
    for i in range(m):
        table[0][i] = 1
    for i in range(1, n+1):
        for j in range(m):

            x = table[i - S[j]][j] if i-S[j] >= 0 else 0

            # Count of solutions excluding S[j]
            y = table[i][j-1] if j >= 1 else 0

            # total count
            table[i][j] = x + y

    return table[n][m-1]


arr = []
m = int(input())
n = int(input())
for i in range(m):
```

	Test	Input	Expected	Got	
	count(arr, m, n)	3 4 1	4	4	

SEC

		16 1 2 5			

Passed all tests!

Correct

Marks for this submission: 20.00/20.00.

Question **4**

Correct

Mark 20.00 out of 20.00

Flag question

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 **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

SEC

Answer: (penalty regime: 0 %)

Reset answer

```
minSum = 1000000000
def getMinPathSum(graph, visited, necessary,
                  src, dest, currSum):

    ################# Add your Code 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;
```

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

Passed all tests!

Correct

Marks for this submission: 20.00/20.00.