**#1 Subarray with given sum**

Given an unsorted array **A**of size **N** of non-negative integers, find a continuous sub-array which adds to a given number **S**.

**Input:**  
The first line of input contains an integer **T** denoting the number of test cases. Then **T** test cases follow. Each test case consists of two lines. The first line of each test case is **N**and**S**, where N is the size of array and S is the sum. The second line of each test case contains **N** space separated integers denoting the array elements.

**Output:**  
For each testcase, in a new line, print the **starting and ending positions**(**1** indexing) of **first such occuring subarray from the left** if sum equals to subarray, else print**-1**.

**Constraints:**  
1 <= T <= 100  
1 <= N <= 107  
1 <= Ai <= 1010

**Example:**  
**Input:**  
2  
5 12  
1 2 3 7 5  
10 15  
1 2 3 4 5 6 7 8 9 10  
**Output:**  
2 4  
1 5

**Explanation :**  
**Testcase1:** sum of elements from 2nd position to 4th position is 12  
**Testcase2:** sum of elements from 1st position to 5th position is 15

#code

def subarray(A,N,S):

su=0

for i in range(T):

for i in range(N):

su=A[i]

j=i+1

while(j<=N):

if su==S:

print(i+1, j)

return 1

if su>S or j==N:

break

su+=A[j]

j+=1

return -1

if \_\_name\_\_=="\_\_main\_\_":

T=int(input())

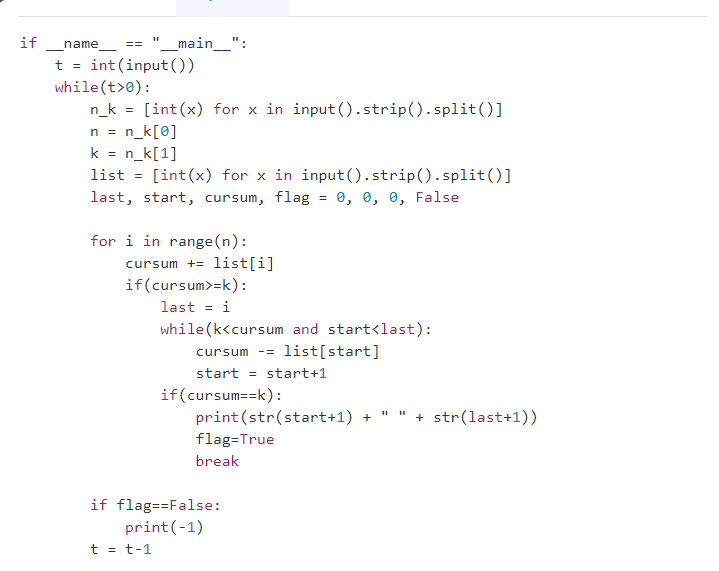
for i in range(T):

N,S=list(map(int,input().split(" ")))

#S=int(input())

A=list(map(int,input().split(" ")))[:N]

subarray(A,N,S)



**#problem2:**

Given an array of distinct integers. The task is to count all the triplets such that sum of two elements equals the third element.

Example 1:

Input:N = 4

arr[] = {1, 5, 3, 2}

Output: 2

Explanation: There are 2 triplets:

1 + 2 = 3 and 3 +2 = 5

arr=[2,3,4]

n=len(arr)

count=0

for i in range(n-1):

    for j in range(i+1,n):

        su=arr[i]+arr[j]

        if su in arr:

            count+=1

            print(arr[i],"+",arr[j]," =",su)

if count>=1:

    print("There are ",count," triplets: ")

else:

    print("No such triplet exits: ")

Given an array **arr**of **N** integers. Find the contiguous sub-array with maximum sum.

**Example 1:**

**Input:**

N = 5

arr[] = {1,2,3,-2,5}

**Output:** 9

**Explanation:** Max subarray sum is 9

of elements (1, 2, 3, -2, 5) which

is a contiguous subarray.

**Example 2:**

**Input:**

N = 4

arr[] = {-1,-2,-3,-4}

**Output:** -1

**Explanation:** Max subarray sum is -1

of element (-1)

**Your Task:**  
You don't need to read input or print anything. The task is to complete the function **maxSubarraySum**() which takes arr and N as input parameters and returns the sum of subarray with maximum sum

def maxSubArraySum(a,size):

max\_so\_far = -9999999 - 1

max\_ending\_here = 0

for i in range(0, size):

max\_ending\_here = max\_ending\_here + a[i]

# if max\_so\_far is less than max\_ending\_here

# then update max\_so\_far

if (max\_so\_far < max\_ending\_here):

max\_so\_far = max\_ending\_here

# check if max\_ending\_here becomes negative at any point

# then make it 0

if max\_ending\_here < 0:

max\_ending\_here = 0

return max\_so\_far

Close