Play with numbers

- You are given an array of n numbers and q queries. For each query you have to print the floor of the expected value(mean) of the subarray from L to R.

- First line contains two integers N and Q denoting number of array elements and number of queries.
- Next line contains N space seperated integers denoting array elements.
- Next Q lines contain two integers L and R(indices of the array).

- print a single integer denoting the answer.

- 1<= N ,Q,L,R <= 10^6
- 1<= Array elements <= 10^9

In [4]:
```
# read no of elements and no of queries
n = input().split()
n[0],n[1] = int(n[0]),int(n[1])


#Read array elements
a = input().split()
sum = [] #initialise the cumulative sum array

#Cumulative sum
for i in range(0,n[0]):
    if i ==0:
        sum.append(int(a[i]))
    else:
        sum.append(int(sum[i-1])+int(a[i]))
del a
#Read each query and calculate the average
for k in range(0,n[1]):
    inputquery= input().split()
    i =int(inputquery[0])
    j = int(inputquery[1])
    if i >1:
        print((sum[j-1] - sum[i-2]) // (j-i+1) )
    else:
        print(sum[j-1] // (j-i+1))
```

```
5 3
1 2 3 4 5
1 3
2
2 4
3
2 5
3
```

In [ ]:
```

```

In [ ]:
```
1
```

Tuples

- t1 = ()
- li = []

- Difference between Lists and Tuples
  - Lists are mutable i.e they can be changed or modified
    - Lists are used to access,modify,add,delete data
  - Tuples are immutable i.e they can't be changed or updated once initialised
    - Used to access data only
    - All slicing operations work

In [12]:
```
1 t1 = (1,2,8,6,0)
2
3 t1[3] #Accessing fourth element
4
5 # Accessing all elememts from middle to last
6
7 t1[len(t1)//2:]
```

Out[12]:
```
(8, 6, 0)
```

In [13]:
```
1 type(t1)
```

Out[13]:
```
tuple
```

In [ ]:
```
1
```

Dictionaries

- It works on the concept of Set

- Unique Data
  - Keys,Values

Key is the unique identifier for a value Value is data that can be accessed with a key

In [25]:
```
d1 = {"k1":"value1","k2":"value2"}

d1["k2"]  #Accessing the value by using the unique key


d1.keys() #returns list of all keys

d1.values()  #returns list of all values

d1.items()  #returns list of tuples of keys and values

d1["k3"]= "value3" #adding an element to the dictionary

d1["k3"] = "value4" #Updating an element
d1.pop("k3")  #Removing an element

"k3" in d1  # check for key to know the value
```

Out[25]:
```
False
```

Contacts Application :

- Add Contact
- Search for contact
- List all contacts
  - name1 : phone1
  - name2 : phone 2
- Modify contact
- Remove contact
- Import contacts

In [29]:
```
contacts= {}

def addContact(name, phone):
    #Verify that the contact doesnot already exist in the dictionary
    if name not in contacts:
        contacts[name]= phone
        print("Contact %s added" % name)
    else:
        print("Contacts %s already exists" % name)
    return

addContact("abc", "8765432109")
```

```
Contact abc added
```

In [30]:
```
contacts

```

Out[30]:
```
{'abc': '8765432109'}
```

In [33]:
```
def searchContacts(name):
    if name in contacts:
        print(name,":",contacts[name])
    else:
        print("%s doesnot exist" % name)
    return


searchContacts("abc")
```

```
abc : 8765432109
```

In [35]:
```
addContact("def","1234567890")
```

```
Contact def added
```

In [36]:
```
contacts.items()
```

Out[36]:
```
dict_items([('abc', '8765432109'), ('def', '1234567890')])
```

In [40]:
```
# updating the value
contacts["def"]="7890123456"
contacts.items()
```

Out[40]:
```
dict_items([('abc', '8765432109'), ('def', '7890123456')])
```

In [41]:
```
#Adding new contact
addContact("fgh","5431289012")
```

```
Contact fgh added
```

In [42]:
```
#Displaying the contacts
contacts.items()
```

Out[42]:
```
dict_items([('abc', '8765432109'), ('def', '7890123456'), ('fgh', '543128901
2')])
```

In [43]:
```
# Poping or deleting a contact
contacts.pop("fgh")
```

Out[43]:
```
'5431289012'
```

In [44]:
```
contacts.items()
```

Out[44]:
```
dict_items([('abc', '8765432109'), ('def', '7890123456')])
```

In [49]:
```
#Importing a contact
# newContacts is goven as a dictionary
# Merge new contacts with existing contacts


def importContacts(newContacts):
    contacts.update(newContacts)
    print(len(newContacts.keys()), "Contacts added Successfully")
    return


newContacts={"name":"1234568809","name2" : "2345678091"}
importContacts(newContacts)
```

```
2 Contacts added Successfully
```

In [48]:
```
contacts
```

Out[48]:
```
{'abc': '8765432109',
 'def': '7890123456',
 'name': '1234568809',
 'name2': '2345678091'}
```

In [61]:
```
#Deletion of a contact

def deleteContact(name):
    if name in contacts:
        contacts.pop("abc")
        print("Contact %s deleted" % name)
    else:
        print("%s dosenot deleted" % name)
    return


deleteContact("abc")

```

```
Contact abc deleted
```

In [62]:
```
#Modify the existing contact

def updateContacts(name):
    if name in contacts:
        contacts["def"] = "1234567890"
        print("Contact %s got updated" % name)
    else:
        print("%s doesnot exist" % name)
    return


updateContacts("def")

```

```
Contact def got updated
```

In [63]:
```
1 contacts
```

Out[63]:
```
{'def': '1234567890', 'name2': '2345678091'}
```

In [64]:
```
1 contacts.items()
```

Out[64]:
```
dict_items([('def', '1234567890'), ('name2', '2345678091')])
```

In [65]:
```
1 addContact("hello", 9848032145)
```

```
Contact hello added
```

In [66]:
```
1 contacts
2
```

Out[66]:
```
{'def': '1234567890', 'name2': '2345678091', 'hello': 9848032145}
```

In [68]:
```
1 searchContacts("hello")
```

```
hello : 9848032145
```

In [ ]:
```
1
```

In [ ]:
```
1
```

Packages and Modules

- **Package** --> Collection of Modules(Python File. py) and subpackages
- **Sub package** -->
- **Module** --> A single python file containing functions
- Package --> Subpackage --> Modules --> Functions

In [ ]:
```
1
```

In [70]:
```
1 import math
2
3 math.floor(123.456)
4
5 math.pi
```

Out[70]:
```
3.141592653589793
```

In [3]:
```
1 #only importing floor from a module
2 from math import floor
3 floor(123.456)
4
```

Out[3]:
```
123
```

In [6]:
```
# importing two functions
from math import floor,pi
floor(123.456)

pi
```

Out[6]:
```
3.141592653589793
```

In [5]:
```
# Renaming the function in current scope

from math import floor as fl
fl(123.456)


```

Out[5]:
```
123
```

In [9]:
```


import random

#To generate a number in the given range
random.randint(0,100)


```

Out[9]:
```
6
```

In [11]:
```
# Function to generate N random numbers in a given range
def generateNRandomNums(n,lb,ub):
    for i in range(0,n):
        print(random.randint(lb,ub), end=" ") #inclusive range

generateNRandomNums(10,0,100)

```

```
77 99 85 59 67 60 9 44 86 37
```

In [ ]:
```

```

In [ ]:
```
# Buidling a Custom package or module

from Packages import numerical

numerical.factorial(28)

```

```
1 2 4 7 14
```

In [ ]:
```
from Packages import numerical

numerical.factorial(28)

```

In [ ]:
```
1 from Packages import numerical
2
3 numerical.is_prime(50)
4
```

```
1 2 4 7 14
```

In [ ]:
```
1 from Packages.numerical import is_prime
2 is_prime(10)
```

```
1 2 4 7 14
```

In [ ]:
```
1 from Packages.numerical import isCheckprime
2 isCheckprime(10)
```

```
1 2 4 7 14
```

In [1]:
```
1 from Packages.numerical import isCheckprime
2 isCheckprime(10)
```

Out[1]:
```
False
```

In [3]:
```
1 from Packages.numerical import isCheckprime
2 isCheckprime(31)
```

Out[3]:
```
True
```

In [ ]:
```
1 from Packages.numerical import is_prime
2 is_prime(10)
```

```
10
20
11
11
6
11
6
```

In [ ]:
```
1
```