Play with numbers

In [11]:
```python
n=input().split()
N=int(n[0])
Q=int(n[1])
#Read array elements
a=input().split()
sum=[]   #Initialize cumulative sum array
# Cumulative sum
for i in range(0,N):
    if i==0:
        sum.append(int(a[0]))
    else:
        sum.append(int(sum[i-1]+int(a[i])))

del a

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

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

In [12]:
```python
### Play with numbers another way
n,q=map(int,input().split())
l=list(map(int,input().split()))
z=[]
e=0
for i in range(0,n):
    e+=l[i]
    z.append(e)
for i in range(0,q):
    s,e=map(int,input().split())
    if(s==1):
        print((z[e-1])//(e-s+1))
    else:
        print((z[e-1]-z[s-2])//(e-s+1))


```

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

Special Number

In [3]:
```python
###Special number
def isspecial(n,p):
    if numberprimefactors(n) >=p:
        return True
    return False


#Functio to check if no is prime
def ischeck(n):
    flag=1
    if n == 2:
        return True
    for i in range(2,n//2 + 1):
        if n % i== 0:
            flag= 0
            return False
    if flag == 1:
        return True
#ischeck(5)


#Function to determine number of prime factors for a given number

def numberprimefactors(n):
    if ischeck(n):
        return 1
    count=0
    for i in range(2,n // 2+1):
        if ischeck(i) and n % i==0:
            count=count+1
    return count

#numberprimefactors(40)
isspecial(6,2)

def solution2():
    p=int(input())
    t=int(input())
    for i in range(0,t):
        n=int(input())
        if isspecial(n,p):
            print("YES")
        else:
            print("NO")
solution2()

```

```
2
3
6
YES
3
NO
1
NO
```

Highest reminder

write a program to find a natural number that is smaller than N that gives the highest reminder when divided by that number if there is more than such number ,print the smallest one x < N and n % x == highest

10 9 9 1 8 1 8 2 7 2 7 3 6 3 6 4 5 4 5 0 4 1 4 2 3 0 3 1 2 1 2 0 1 0 1 0

In [2]:
```
def highestrem(n):
    hr=0
    v=n
    for i in range(n-1,n // 2,-1):
        r= n% i
        if r>hr:

            hr = r
            v = i
    print(v)
    return

highestrem(30)


```
```
16
```

Tuples

differece between tuples and lists

lists are mutable-can be changed/modified

- used to access,modify,add,delete data

tuples are immutable- cannot be changed once initialised

- used to acess the data only
- All Slicing Operations will be work

In [8]:
```
t1=(1,2,8,6,0)
t1[3]          #Acessing the fourth element of the tuple
t1[len(t1)//2:]      #Accessing all elements from middle to last
```
Out[8]:
```
(8, 6, 0)
```

In [9]:
```
type(t1)
```
Out[9]:
```
tuple
```

Dictionaries

- It works on the concepts of set

- unique Data
- Keys,Values
- Key is the unique idntifier for a value
- Vlaue is the data that can be accessed with a key

In [24]:
```
d1={"k1":"value1","k2":"value2"}
d1["k2"]     # Accessing the value with key "k2"
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"]="values3"  # Adding an element to the dictionary

d1["k3"]="value4"  # Updating an element
d1.pop("k2")       # removing an element
d1
"value1" in d1

```

Out[24]:
```
False
```

```
### Contacts Applications
- Add contacts
- Search for contacts
- List all contacts
    - name1 : phone1
    - name2 : phone2
- Modify contacts
- Remove contacts
```

In [3]:
```
contacts={}
def addcontact(name,phone):
    if name not in contacts:
        contacts[name]=phone
        print("contact %s added" %name)
    else:
        print("contact %s already exists" % name)
    return
addcontact("name1","8790700295")


```

```
contact name1 added
```

In [4]:
```
def searchcontact(name):
    if name in contacts:
        print(name,":",contacts[name])
    else:
        print("%s does not exist" %name)
    return
searchcontact("name1")
```

```
name1 : 8790700295
```

In [7]:
```python
def modifycontact(name,phone):
    if name not in contacts:
        contacts[name]=phone
        print("contacts %s added" %name)
    else:
        print("%s does not exit" %name)
    return
modifycontact("name","9492363502")
```

```
contacts name added
```

In [41]:
```python
contacts
```

Out[41]:
```
{'name1': '8790700295', 'vanitha': '9492363502', 'name': '9492363502'}
```

In [5]:
```python
def importcontact(newcontacts):
    contacts.update(newcontacts)
    print(len(newcontacts.keys())), "added successfully"
    return
newcontacts={"name2":9505820607,"name3":6543221}
importcontact(newcontacts)
```

```
2
```

In [6]:
```python
contacts
```

Out[6]:
```
{'name1': '8790700295', 'name2': 9505820607, 'name3': 6543221}
```

In [10]:
```python
def removecontact(name):
    if name in contacts:
        contacts.pop(name)
        print("%s contact is deleted" %name)
    else:
        print("%s contact is not deleted" %name)
    return

removecontact("vanitha")


```

```
vanitha contact is not deleted
```

In [9]:
```python
contacts
```

Out[9]:
```
{'name1': '8790700295',
 'name2': 9505820607,
 'name3': 6543221,
 'name': '9492363502'}
```

In [15]:
```
def listofcontacts(n):
    for i in n.keys():
        print(i,":",contacts[i])
    return
listofcontacts(contacts)
```

```
name1 : 8790700295
name2 : 9505820607
name3 : 6543221
name : 9492363502
```

Packages and Modules

- Packages -> collection of modules(python file .py) and sub packages
- Modules -> A single python file containing functions
- package -> Subpackage -> Modules -> Functions

In [33]:
```
from  math import floor as fl

floor(123.456)

#pi

#math.floor(123.456)
#math.ceil(123.456)
#math.factorial(6)
#math.pi
#math.gcd(12,14)
```

Out[33]:
```
123
```

In [52]:
```
#Function to generate a N random numbers

import random
def generateRandom(n,lb,ub):
    for i in range(0,n):
        print(random.randint(lb,ub),end="  ")
generateRandom(10,0,100)





#random.randint(0,100)
```

```
4  12  41  28  84  31  13  48  39  70
```

In [8]:
```
from Packages.numerical import ischeck

ischeck(5)
```

Out[8]:
```
True
```

In [9]:
```
from Packages import numerical
numerical.ischeck(5)
```

Out[9]:
```
True
```

In [ ]:
```


```