In [122]:
```
def isspecial(n,p):
    if primefactors(n)>=p:
        return True
    return False
def primeNo(n):
    fc = 0
    for k in range(1,n+1):
        n = k
        c = 0
        for i in range(1,n+1):
            if(n%i==0):
                c=c+1
        if(c==2):
            fc = fc+1
    if (fc==2):
        print("YES")
    else:
        print("NO")

primeNo(7)
def primefactors(n):
    if primeNo(n):
        return 1
    count=0
    for i in range(2,n//2+1):
        if primeNo(i) and n%i==0:
            count=count+1
    return count
isspecial(30,2)
```

```
NO
NO
NO
YES
YES
NO
NO
NO
NO
NO
NO
NO
NO
NO
NO
NO
```

Out[122]: True

In [3]:
```
#play with numbers
n=input().split()
n[0],n[1]=int(n[0]),int(n[1])
a=input().split()
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 query
for k in range(0,n[1]):
    quer=input().split()
    i=int(quer[0])
    j=int(quer[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 [31]:
```
## funtion to determine if no is special
def specialno(n,p):
    if numberprimefact(n2)>p:
        return True
    return False


## funtion to check if number is prime
def isprime(n):
    c=0
    for i in range(1,n):
        if n%i==0:
            c=c+1
    if c==2:
        return True
    else:
        return False
isprime(n)
#function to dertermine prime factors
def numberprimefact(n1):
    if isprime(n1):
        return 1
    count=0
    for j in range(2,n1+1):
        if isprime(j) and n1%j==0:
            count=count+1
    print(count)
numberprimefact()

def solution(p,t):
    p=int(input())
    t=int(input())
    for i in range(0,t):
        n=int(input())
        if specialno(n,p):
            print("YES")
        else:
            print("FALSE")
specialno(10,2)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-31-b94bdcfc9795> in <module>
     26             count=count+1
     27     print(count)
---> 28 numberprimefact()
     29
     30 def solution(p,t):

TypeError: numberprimefact() missing 1 required positional argument: 'n1'
```

In [29]:
```
## problem:highest remainder
#program to find natural no that is smaller then heighest remainder when divided
# N
# highest=0
#x<N and n%x==highest
def heighestrem(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)
heighestrem(5)
```

```
3
```

Tuples

t1=()
li=[]

difference b/w list and tuple

lists are mutable_can be changed/modified

- used to Acces,modify,add delete data Tuples are immutable-cannot be cahnged
- used to access data only
- all slicing we can perform

In [32]:
```
t1=(1,2,3,4)
t1[1:]
```

Out[32]:
```
(2, 3, 4)
```

In [33]:
```
len(t1)
```

Out[33]:
```
4
```

In [35]:
```
t1[len(t1)//2:]
```

Out[35]:
```
(3, 4)
```

In [36]:
```
type(t1)
```

Out[36]:
```
tuple
```

Dictionaries

its works on the concept of set
unique data
keys,values
key is unique identifier for a value value is data that can be accessed with data

In [37]:
```
d1={"s":"Satheesh",1:"one"}
```

In [40]:
```
d1[1]
```

Out[40]:
```
'one'
```

In [41]:
```
d1["s"]
```

Out[41]:
```
'Satheesh'
```

In [42]:
```
d1.keys()
```

Out[42]:
```
dict_keys(['s', 1])
```

In [43]:
```
d1.values()
```

Out[43]:
```
dict_values(['Satheesh', 'one'])
```

In [44]:
```
d1.items()
```

Out[44]:
```
dict_items([('s', 'Satheesh'), (1, 'one')])
```

In [45]:
```
d1[2]="two"
```

In [46]:
```
d1
```

Out[46]:
```
{'s': 'Satheesh', 1: 'one', 2: 'two'}
```

In [ ]:
```

```

In [48]:
```
d1.popitem()
```

Out[48]:
```
(2, 'two')
```

In [49]:
```
d1.pop(1)
```

Out[49]:
```
'one'
```

In [55]:
```
# contact application
# search for contacts
# list all contacts
# modify contacts
# remove contacts
contacts={}
def addcontacts(name,phone):
    if name not in contacts:
        contacts[name]=phone
        print("contact % added" % name)
    else:
        print("Contact %s is already exists" % name)
addcontacts("satheesh",9676047715)
```

```
contact 'satheesh'dded
```

In [51]:
```
contacts
```

Out[51]:
```
{'satheesh': 9676047715}
```

In [56]:
```
contacts
```

Out[56]:
```
{'satheesh': 9676047715}
```

In [57]:
```
contacts={}
def addcontacts(name,phone):
    if name not in contacts:
        contacts[name]=phone
        print("contact % added" % name)
    else:
        print("Contact %s is already exists" % name)
addcontacts("satheesh",9676047715)
```

```
contact 'satheesh'dded
```

In [58]:
```
contacts
```

Out[58]:
```
{'satheesh': 9676047715}
```

In [59]:
```
def addcontacts(name,phone):
    if name not in contacts:
        contacts[name]=phone
        print("contact % added" % name)
    else:
        print("Contact %s is already exists" % name)
addcontacts("satheesh",9676047715)
```

```
Contact satheesh is already exists
```

In [61]:
```
def searchcontacts(name):
    if name in contacts:
        print(name,":",contacts[name])
    else:
        print("%s doesnot exit" % name)
searchcontacts("sathees")
```

```
sathees doesnot exit
```

In [ ]:
```

```

In [65]:
```
contacts
```

Out[65]:
```
{'satheesh': 9676047715}
```

In [73]:
```
#modify
def modify(name,phone):
    if name in contacts:
        contacts[name]=phone
    else:
        print("wrong")
modify("satheesh",9676047719)
```

In [74]:
```
contacts
```

Out[74]:
```
{'satheesh': 9676047719, 9676047719: 9676047719}
```

In [75]:
```
contacts
```

Out[75]:
```
{'satheesh': 9676047719, 9676047719: 9676047719}
```

In [76]:
```
def modify(name,phone):
    if name in contacts:
        contacts[name]=phone
    else:
        print("wrong")
modify("satheesh",9676047710)
```

In [77]:
```
contacts
```

Out[77]:
```
{'satheesh': 9676047710, 9676047719: 9676047719}
```

In [97]:
```
#list all contacts
def listcontacts():
    if contacts:
        print(contacts,"\n")
    else:
        print("contacts are empty")
listcontacts()
```

```
{'satheesh': 1234567890, 'name2': 'ravi'}
```

In [80]:
```
#import contacts
def importcontacts(newcontacts):
    contacts.update(newcontacts)
    print(len(newcontacts.keys()),"contacts added")
newcontacts={"name1":"Rajesh","name2":"ravi"}
importcontacts(newcontacts)
```

```
2 contacts added
```

In [81]:
```
newcontacts
```

Out[81]:
```
{'name1': 'Rajesh', 'name2': 'ravi'}
```

In [82]:
```
#update contacts
def updatename(name):
    if name in contacts:
        phone=int(input("enter number"))
        contacts[name]=phone
        print("successfully added")
    else:
        print(" not exit")
updatename("satheesh")
```

```
enter number1234567890
successfully added
```

In [83]:
```
contacts
```

Out[83]:
```
{'satheesh': 1234567890,
 9676047719: 9676047719,
 'name1': 'Rajesh',
 'name2': 'ravi'}
```

In [87]:
```
#remove contacts
def removecontact(name):
    if name in contacts:
        contacts.pop(name)
        print("% removed" % name)
    else:
        print("no data")
removecontact(9676047719)
```

```
9676047719emoved
```

In [86]:
```
contacts
```

Out[86]:
```
{'satheesh': 1234567890, 9676047719: 9676047719, 'name2': 'ravi'}
```

In [88]:
```
contacts
```

Out[88]:
```
{'satheesh': 1234567890, 'name2': 'ravi'}
```

In [2]:
```
#list table
# def listtablecontacts():
#     #print(contacts)
#     l=[]
#     if name in contacts:
#         l.append(contacts[name])


# listtablecontacts()
```

packages and modules

- package->collection of modules and sub packages
  **modules ** -->collection of methods(single python file which contains diff functions

In [102]:
```
import keyword
dir(keyword)
```

Out[102]:
```
['__all__',
 '__builtins__',
 '__cached__',
 '__doc__',
 '__file__',
 '__loader__',
 '__name__',
 '__package__',
 '__spec__',
 'iskeyword',
 'kwlist',
 'main']
```

In [104]:
```
import math
dir(math)
```

Out[104]:
```
['__doc__',
 '__loader__',
 '__name__',
 '__package__',
 '__spec__',
 'acos',
 'acosh',
 'asin',
 'asinh',
 'atan',
 'atan2',
 'atanh',
 'ceil',
 'copysign',
 'cos',
 'cosh',
 'degrees',
 'e',
 'erf',
 'erfc',
 'exp',
 'expm1',
 'fabs',
 'factorial',
 'floor',
 'fmod',
 'frexp',
 'fsum',
 'gamma',
 'gcd',
 'hypot',
 'inf',
 'isclose',
 'isfinite',
 'isinf',
 'isnan',
 'ldexp',
 'lgamma',
 'log',
 'log10',
 'log1p',
 'log2',
 'modf',
 'nan',
 'pi',
 'pow',
 'radians',
 'remainder',
 'sin',
 'sinh',
 'sqrt',
 'tan',
 'tanh',
```

```
 'tau',
 'trunc']
```

In [110]:
```
from math import floor as f,pi
f(193.456)
#pi
```

Out[110]:
```
193
```

In [120]:
```
import random
def generaterandomno(n,lb,ub):
    for i in range(0,n):
        print(random.randint(lb,ub),end=" ")
generaterandomno(4,10,30)
```

```
16 19 12 11
```

In [118]:
```
dir(random)
```

Out[118]:
```
['BPF',
 'LOG4',
 'NV_MAGICCONST',
 'RECIP_BPF',
 'Random',
 'SG_MAGICCONST',
 'SystemRandom',
 'TWOPI',
 '_BuiltinMethodType',
 '_MethodType',
 '_Sequence',
 '_Set',
 '__all__',
 '__builtins__',
 '__cached__',
 '__doc__',
 '__file__',
 '__loader__',
 '__name__',
 '__package__',
 '__spec__',
 '_acos',
 '_bisect',
 '_ceil',
 '_cos',
 '_e',
 '_exp',
 '_inst',
 '_itertools',
 '_log',
 '_os',
 '_pi',
 '_random',
 '_sha512',
 '_sin',
 '_sqrt',
 '_test',
 '_test_generator',
 '_urandom',
 '_warn',
 'betavariate',
 'choice',
 'choices',
 'expovariate',
 'gammavariate',
 'gauss',
 'getrandbits',
 'getstate',
 'lognormvariate',
 'normalvariate',
 'paretovariate',
 'randint',
 'random',
 'randrange',
 'sample',
```

```
 'seed',
 'setstate',
 'shuffle',
 'triangular',
 'uniform',
 'vonmisesvariate',
 'weibullvariate']
```

In [124]:
```
from packages import numerical
numerical.primeNo(5)
```

```
NO
```

In [ ]:
```

```