Problem Statements

Special Number

- A special number is defined as a number which has at least P disctinct prime factors
- Write a program to determine whether a number N is a special number
- Input Format
  - First line: P
  - Second line: T(number of test cases)
  - Next T lines: N
- output Format
  - for each test case ,print YES or NO depending on the result
- Constraints
  - 1<=T<=20
  - 1<=P<=10**7
  - 1<=P<=N
- Sample Input - Sample Output
  - 2
  - 6
  - 1 - NO
  - 6 - YES
  - 7 - NO
  - 8 - NO
  - 9 - NO
  - 10 - YES

In [1]:
```
def isPrime(n):
    c=0
    for i in range(1,n+1):
        if n%i==0:
            c+=1
    if(c==2):
        return True
    else:
        return False

def SpecialNum(num):
    count = 0

    for i in range(1,T+1):
        #count = 0
        if num%i==0:

            if isPrime(i):
                count=count+1
    if(count>=pcount):
        return "YES"
    else:
        return "NO"




pcount=int(input())
T=int(input())
for i in range(1,T+1):
    num = int(input())
    print(SpecialNum(num))

```

```
2
6
34
NO
6
YES
28
NO
45
YES
67
NO
10
YES
```

In [3]:
```
def PrimeorNot(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

n = int(input())
PrimeorNot(n)
```

```
5
```

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

Problem Statement

Play with Numbers

In [ ]:
```
# Read no of array 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 = [] # initialize cummulative sum array

# Cummulative 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 i in range(0,):
    inq = input.split()     # input query
    i = int(inq[0])
    j = int(inq[1])
    print()

```

Problem Statememt

- Consider All lowercase Alphabets of the English language. Here we consider each alphabet from a to z to have a certain weight. The weight of the alphabet a is considered to be 1, b to be 2, c to be 3 and so on until z has a weight of 26. In short, the weight of the alphabet a is 1, and the weight of all other alphabets is the weight of its previous alphabet + 1.
- Now, you have been given a String S consisting of lowercase English characters. You need to find the summation of weight of each character in this String.
- For example, Consider the String aba
- Here, the first character a has a weight of 1, the second character b has 2 and the third character a again has a weight of - 1. So the summation here is equal to : 1+2+1=4
- Input Format:
  - The first and only line of input contains the String S.
- Output Format:
  - Print the required answer on a single line
- Constraints:
  - 1≤|S|≤100
- Sample Input - Sample Output
  - aba -4

In [1]:

```python
# Char Sum
def charsum(s):
    sum = 0
    for i in range(0,len(s)):
        e = ord(s[i]) - 96
        sum = sum + e
    return sum



s=input()
print(charsum(s))
```

```
abcd
10
```

Problem : Highest Remainder

- Write a program to find a natural number that is smaller than n such that N gives the highest remainder when divided by that number, If there is more than one such number,print the smallest one.
- input format
  - First line: T (number of test cases)
  - Next T line : N
- Output farmat
  - for each test case,print a natural number that is smaller than N such that N gives the hightest remainder when divided by that number
- Constraints
  - 1<=T<=10**5
  - 2<=N<= 10**9
- Sample Input - Sample Output
  - 2

- 5 - 3
- 4 - 3

- Explanation
    - 4 % 3 = 1 & 5 % 3 = 2
    - These are the maximum possible remainders for 4 and 5

In [7]:
```
def highestRemainder(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
highestRemainder(30)
```

```
16
```

In [12]:
```
#li[]
s = input()
li = [s.split()]
#li
```

```
22 3 4 5 6 77 6
```

Tuples

Difference between Lists and Tuples

- t1 = ( ) --> Tuple
- li = [ ] --> List

lists are mutable - can be changed / modified-

- Used to Access, Modify,Add,Delete data

Tuples are immutable - Cannot be changed once initialised

- Used to access data only
- All Slicing operations work

In [19]:
```
t1 = (1, 2, 8, 6, 0)

t1[3]    # Accessing the fourth element

# Accessing all elements from middle to last
t1[len(t1)//2:]
```

Out[19]:
```
(8, 6, 0)
```

In [20]:
```
type(t1)
```
Out[20]:
```
tuple
```

In [ ]:
```

```

Dictionaries

It works on the concept of Set

- Dictionaries has Unique Data

It has two parameters

Keys, Values

- Key is the unique identifier for a value
- Value is data that can be accessed with a key
- Dictionaries are like list we can add delete an element

In [21]:
```
d1 = {"k1":"Value1", "k2":"value2" }
d1["k2"]   # Accessing the value with key "k2"

```
Out[21]:
```
'value2'
```

In [22]:
```
d1.keys() # return list of all keys
```
Out[22]:
```
dict_keys(['k1', 'k2'])
```

In [23]:
```
d1.values() # returns list of all values
```
Out[23]:
```
dict_values(['Value1', 'value2'])
```

In [24]:
```
d1.items() # returns list of tuples of keys and values
```
Out[24]:
```
dict_items([('k1', 'Value1'), ('k2', 'value2')])
```

In [26]:
```
d1["k3"] = "value3"    # Adding a key and value to d1
d1
```
Out[26]:
```
{'k1': 'Value1', 'k2': 'value2', 'k3': 'value3'}
```

In [39]:
```
# Updating an element
d1["k3"] = "value4"   # Value of key 3 i.e k3 has updated,But we cannot obta
d1
d1.pop("k3")  # Removing an element
"k3" in d1    # --> False
"k1" in d1   # --> True
"value1" in d1  # --> False   since it only searching for keys



```

Out[39]:
```
False
```

Contacts Application

- Add Contact
- Search for contact
- List all contacts
  - name1 : phone1
  - name2 : phone2
- Modify contact
- Remove contact
- Import contacts

In [64]:
```
contacts = {}

def addContact(name,phone):
    # verify that the caontact already exit in contacts
    if name not in contacts:
        contacts[name] = phone
        print("Contact %s added"  % name)
    else:
        print("Contact %s already exits"  % name)
    return

addContact("name1","1234567890")
#addContact()
```

```
Contact name1 added
```

In [65]:
```
def searchContacts(name):
    if name in contacts:
        print(name, ":", contacts[name])
    else:
        print("%s does not exists" % name)
    return
searchContacts("name1")
```

```
name1 : 1234567890
```

In [62]:
```
# New contacts is given as a dictionary
# Merge new contacts with existing contacts
def importContacts(newContacts):
    contacts.update(newContacts)
    print(len(newContacts.keys())," contacts added successfully")
    return
newContacts = {"name2":9876543210,"name3":6537837637}

importContacts(newContacts)

```

```
2  contacts added successfully
```

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

Out[63]:
```
{'name1': '1234567890', 'name2': 9876543210, 'name3': 6537837637}
```

In [ ]:
```
def modifyContacts():

```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```

```

Packages and Modules

- **Packages** --> Collection of Modules(Python File.py) and subpackage
- **Module** --> A single python file containing functions
- Package --> Subpackages --> Modules --> Function

In [69]:
```
import math

math.floor(123.456)

math.pi
```

Out[69]:
```
3.141592653589793
```

In [74]:
```
from math import floor,pi
floor(12888.980993993)    # output -->1288
pi

```

Out[74]:
```
3.141592653589793
```

In [75]:
```python
from math import floor as fl
fl(23444.99494949)
```
Out[75]:
```
23444
```

In [85]:
```python
# Function to generate N random numbers

import random

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


generateNRandomNumbers(10, 0, 100)

```
```
90 48 25 51 51 8 47 4 46 70
```

In [4]:
```python
from Packages import numerical

numerical.isPrime(5)
```
Out[4]:
```
True
```

In [5]:
```python
from Packages.numerical import isPrime
```

Problem Statement

Goki and his breakup

- Goki recently had a breakup, so he wants to have some more friends in his life. Goki has N people who he can be friends with, so he decides to choose among them according to their skills set Yi(1<=i<=n). He wants atleast X skills in his friends.
- Help Goki find his friends.
- INPUT
  - First line of the input contains an integer N denoting the number of people.
  - Next line contains a single integer X - denoting the minimum skill required to be Goki's friend.
  - Next n lines contain one integer Y - denoting the skill of ith person.
- OUTPUT
  - For each person print if he can be friend with Goki. 'YES' (without quotes) if he can be friends with Goki else 'NO' (without quotes).
- CONSTRAINTS
  - 1<=N<=1000000
  - 1<=X,Y<=1000000
- SAMPLE INPUT - SAMPLE OUTPUT
  - 5
  - 100
  - 110 --> YES
  - 130 ---> YES

- 90 ---> NO
- 100 ---> YES
- 45 ---> NO

In [6]:
```python
# GOKI AND HIS BREAKUP
def GokiFrnd(Y):
    if Y>=X:
        print("YES")
    else:
        print("NO")

N = int(input())
X = int(input())
for i in range(1,N+1):
    Y = int(input())
    GokiFrnd(Y)
```

```
5
100
110
YES
23
NO
555
YES
456
YES
23
NO
```

In [1]:
```
# Play with numbers

n = input().split()
n[0],n[1] = int(n[0]),int(n[1])

a = input().split()
sum = []

# Cummualative 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
   # sum[0] = # first element
    #sum[1] = #first + second
    #sum[2] = # sum[1]+ third element
for k in range(0,n[1]):
    l,r = map(int,input().split())
    if l>1:
        print((sum[r-1] - sum[l-2]) // (r -l +1))
    else:
        print(sum[r-1] // (r - l+1))
```

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

In [ ]:
```

```