In [ ]:
```
# Play with numbers

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

a = input().split
sum = []

#Cimilative Sum
for i in range(0,n):
    if i == 0:
        sum.append(int(a[i]))
    else:
        sum.append(int(sum[i-1])+int(a[i]))

print(sum[n[0]-1])
```

In [ ]:
```
# Specail Number

def isPrime(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

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

def isSpecialNumber(n,p):
    if numberPrimeFactors(n) >= p:
        return True
    return False


numberPrimeFactors(30)
isSpecialNumber(7,2)
```

In [ ]:
```
def solution():
    p = int(input())
    t = int(input())
    for i in range(0,t):
        n = int(input())
        if isSpecialNumber(n,p):
            print("YES")
        else:
            print("NO")

solution()
```

In [ ]:
```
# Highest Remainder

def highestRemainder(n):
    hr = 0
    v = n
    for i in range(n-2,n//2,-1):
        r = n%i
        if r>hr:
            hr = r
            v =i
    print(v)
    return

highestRemainder(30)
```

Tuples

- t1 = ()
- li = []

Difference between Lists and Tuples

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 work

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

t1[3] # Accessing the Fourth element

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

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

```
In [ ]:
```

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

        d1["k2"] #Accessing the value with key "k2"

        d1.keys() #returns the list of all keys

        d1.values() #returns the list of all values

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

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

        d1

        d1["k3"] = "values3" # updating an element in the dictionary

        d1.pop("k3") #Removing an element

        d1
```

```
In [ ]: dir(d1)
```

Contacts Application

- Add Contact
- Search for Contact
- List all Contacts
- Modify Contact
- Removing Contact

In [7]:
```
contacts = {}

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

addContact("chaitu","8106410134")
```

```
Contact chaitu added
```

In [8]:
```
addContact("chaitu","8106410134")
```

```
Contact chaitu already exists
```

In [11]:
```
def searchContacts(name):
    if name in contacts:
        print (name,":",contacts[name])
    else:
        print("%s doesnot exists"%name)
    return

searchContacts("chaitu")
```

```
chaitu : 8106410134
```

In [ ]:
```
searchContacts("patty")
```

In [10]:
```
def getAll(contacts):
    print(contacts)

getAll(contacts)
```

```
{'chaitu': '8106410134'}
```

In [12]:
```
def importContacts(newContacts):
    contacts.update(newContacts)
    print(len(newContacts.keys()),"got added successfully")
    return

newContacts = {"patty":9876543210,"charly":1234567890}

importContacts(newContacts)
```

```
2 got added successfully
```

In [17]:
```
def updatingContact(name,phone):
    if name in contacts:
        contacts[name] = phone
        print("Updated Successfully")
    else:
        print("contact not available")
    return

updatingContact("chaitu",8330990517)
```

```
Updated Successfully
```

In [21]:
```
contacts
```

Out[21]:
```
{'chaitu': 8330990517, 'patty': 9876543210}
```

In [19]:
```
def removingContact(name):
    if name in contacts:
        contacts.pop(name)
        print("Contact removed")
    else:
        print("contact Does not Exists")
    return

removingContact("charly")
```

```
Contact removed
```

In [20]:
```
contacts
```

Out[20]:
```
{'chaitu': 8330990517, 'patty': 9876543210}
```

In [ ]:
```

```

```
### Packages and Modules

Package -> Collection of Modules(Python File)

Sub-Package ->

Module -> A single python file containing functions

Package -> Subpackages -> Modules -> Functions
```

In [22]:
```
import  math

math.floor(123.456)
```

Out[22]:
```
123
```

In [26]:
```
help
```

Out[26]:
```
Type help() for interactive help, or help(object) for help about object.
```

In [ ]:
```
help()
```

```

Welcome to Python 3.7's help utility!

If this is your first time using Python, you should definitely check out
the tutorial on the Internet at https://docs.python.org/3.7/tutorial/. (http
s://docs.python.org/3.7/tutorial/.)

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules.  To quit this help utility and
return to the interpreter, just type "quit".

To get a list of available modules, keywords, symbols, or topics, type
"modules", "keywords", "symbols", or "topics".  Each module also comes
with a one-line summary of what it does; to list the modules whose name
or summary contain a given string such as "spam", type "modules spam".

help> modules

Please wait a moment while I gather a list of all available modules...


C:\Users\CHAITANYA\Anaconda3\lib\site-packages\IPython\kernel\__init__.py:13: S
himWarning: The `IPython.kernel` package has been deprecated since IPython 4.0.
You should import from ipykernel or jupyter_client instead.
  "You should import from ipykernel or jupyter_client instead.", ShimWarning)
```

In [3]:
```
import math

math.pi
```

Out[3]:
```
3.141592653589793
```

In [ ]:
```
from math import floor

math.floor(123.456)
```

In [11]:
```
# 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)
```

```
88 2 74 23 49 65 21 9 4 74
```

In [ ]:

In [14]:
```
from Packages import numerical
numerical.isPrime(4)
```

Out[14]:
```
False
```

In [ ]: