You have given number N

Write a program to find natural number that is samller than N such that N gives the highest remainder when divided by the number

if there is more than one such num,print the smallest one

In [101]:
```
def number(N):
    s=[]
    j=[]
    for i in range(1,N):
        l=N%i
        s.append(l)
        j.append(i)

    g=s.index(max(s))
    return (j[g])








T=int(input())
for i in range(T):
    N=int(input())
    print(number(N))
```

```
1
5
3
```

In [104]:
```
#Another method
def highestnumber(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
highestnumber(30)
```

```
16
```

problem : special number

a special number is defined a number which has atleast p distinct prime factors

write a program to determine wheather a number N is a special number

input format

- first line:p
- second line :t(test case)
- next T lines:N
- output:
- for each test case ,print "YES" or "NO" depending on range

In [82]:
```
#function to determine if a number is special number or not
#function to check if number is prime
#function to determine number of prime factors for a given number
def isspecialnumber(n,p):
    if primefactors(n)>=p:
        return True
    return False
def isprime(n):
    count=0
    for i in range(1,n+1):
        if n%i==0:
            count=count+1
    if count==2:
        return True
    return False
def primefactors(n):
    count1=0
    if isprime(n):
        return 1

    for i in range(1,n+1):
        if isprime(i) or n%i==0:
            count1+=1
    return count1
#primefactors(30)
isspecialnumber(7,2)
```

Out[82]:
```
False
```

```
In [88]: dir(list)
```

```
Out[88]: ['__add__',
          '__class__',
          '__contains__',
          '__delattr__',
          '__delitem__',
          '__dir__',
          '__doc__',
          '__eq__',
          '__format__',
          '__ge__',
          '__getattribute__',
          '__getitem__',
          '__gt__',
          '__hash__',
          '__iadd__',
          '__imul__',
          '__init__',
          '__init_subclass__',
          '__iter__',
          '__le__',
          '__len__',
          '__lt__',
          '__mul__',
          '__ne__',
          '__new__',
          '__reduce__',
          '__reduce_ex__',
          '__repr__',
          '__reversed__',
          '__rmul__',
          '__setattr__',
          '__setitem__',
          '__sizeof__',
          '__str__',
          '__subclasshook__',
          'append',
          'clear',
          'copy',
          'count',
          'extend',
          'index',
          'insert',
          'pop',
          'remove',
          'reverse',
          'sort']
```

```
In [105]:
```

```
Out[105]: 15
```

Tuples

- ti=()
- li-[]
- differnce b/w list and tupple

list are mutable-can be changed/modified

- Used to Access,Modify,Add,Delete Data

Tuples are immutable--cannot ne changed once initialised

- Used to Acess the data only
- Slicing operation works

In [124]:
```
t1=(1,2,3,4)

t1[3]#Accessing the data

t1[2::] #slicing the data

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

Out[124]:
```
(3, 4)
```

```
In [109]: dir(tuple)
```

```
Out[109]: ['__add__',
           '__class__',
           '__contains__',
           '__delattr__',
           '__dir__',
           '__doc__',
           '__eq__',
           '__format__',
           '__ge__',
           '__getattribute__',
           '__getitem__',
           '__getnewargs__',
           '__gt__',
           '__hash__',
           '__init__',
           '__init_subclass__',
           '__iter__',
           '__le__',
           '__len__',
           '__lt__',
           '__mul__',
           '__ne__',
           '__new__',
           '__reduce__',
           '__reduce_ex__',
           '__repr__',
           '__rmul__',
           '__setattr__',
           '__sizeof__',
           '__str__',
           '__subclasshook__',
           'count',
           'index']
```

Dictionaries

- It works on the concept of set
- Unique Data
- Keys,value
- key is unique identifier for a value
- value is a data that can be accessed with a key
- Dictionary can be mutable

In [147]:
```
d1={"k1":"value1","k2":"value2"}

d1["k1"]#Accessing the value with k1

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["j1"]="value3"#adding the new key and value

d1["k1"]="value3"# replacing the value

d1.pop("k1")#removing an element

d1.pop("k2")#removes the element and returns the value which is removed
d1
"j1" is d1
```

Out[147]:
```
False
```

Contact Application

- Add Contact
- Search for contact
- List all contacts
  - name1:phone1
  - nmae2:phone2
- Modify Contact
- Remove Contact

In [153]:
```
contacts={}
def addcontact(name,phonenumber):
    #if name
    contacts[name]=phonenumber
    return contacts
addcontact("alekhya","928999777")
```

Out[153]:
```
{'alekhya': '928999777'}
```

In [154]:
```
contacts={}
def addcontact(name,phonenumber):
    if name not in contacts:
        contacts[name]=phonenumber
    else:
        print("contact %s is already exits" % name)
    return
addcontact("alekhya","928999777")
```

In [155]:
```
contacts
```

Out[155]:
```
{'alekhya': '928999777'}
```

In [161]:
```
def searchcontact(name):
    if name  in contacts:
        print(name,":",contacts[name])
    else:
        print("%s does not exist"% name)
    return
searchcontact("alekhya")
```

```
alekhya : 928999777
```

In [163]:
```
def modify(name,phonenumber):
    if name in contacts:
        contacts[name]=phonenumber
        return contacts
    else:
        print("%s does not exist"% name)
modify("alekhya","832833444442")
```

Out[163]:
```
{'alekhya': '832833444442'}
```

In [164]:
```
def remove(name):
    if name in contacts:
        contacts.pop(name)
    return contacts
remove("alekhya")
```

Out[164]:
```
{}
```

In [181]:
```
contacts={"a":"123144","b":"7518"}
def listofcontacts():
    for key,value in contacts.items():

        print(key,":",value)
listofcontacts()
```

```
a : 123144
b : 7518
```

In [173]:
```
#New contacts in a given dictionary
#Merge two data
def importcontact(newcontacts):
    contacts.update(newcontacts)
    print(len(newcontacts.keys()),"contacts added successfully")
    return
newcontacts={"nam1":1344444,"na2":13444}
importcontact(newcontacts)
```

```
2 contact added successfully
```

In [ ]:
```

```