Rahul is a very busy person he dont wan't to waste his time . He keeps account of duration of each and every work. Now he don't even get time to calculate duration of works, So your job is to count the durations for each work and give it to rahul.

- Input:
  - First line will be given by N number of works
    - Next N line will be given SH,SM,EH and EM each separated by space(SH=starting hr, SM=starting min, EH=ending hr, EM=ending min)
- Output:
  - N lines with duration HH MM(hours and minutes separated by space)

- **Input** start time , end time (HH,MM) HH = {00, 01, 02, 03 .. 23} MM = {00,01, 02,03, .. 59}

  `HH MM = { 00 00. 23 59 }`

- **Output** time difference in HH MM

In [4]:
```python
#Calculate the  time differnce as total number of minutes
# Convert the total minutes into HH MM

s= "2 42 8 23"

def durationDifference(s):
    s = s.split()
    sh = int(s[0])
    sm = int(s[1])
    eh = int(s[2])
    em = int(s[3])
    startminutes = (sh * 60) + sm
    endminutes = (eh * 60) + em
    return endminutes - startminutes

def outputTimeFormat(minutes):
    # Convert minutes to HH MM format

    hh = minutes // 60
    mm = minutes % 60
    print(hh,mm)
    return

minutes=durationDifference(s)
outputTimeFormat(minutes)
```

```
5 41
```

In [1]:
```python
def durationDifference(s):
    s = s.split()
    sh = int(s[0])
    sm = int(s[1])
    eh = int(s[2])
    em = int(s[3])
    startminutes = (sh * 60) + sm
    endminutes = (eh * 60) + em
    return endminutes - startminutes

def outputTimeFormat(minutes):
    # Convert minutes to HH MM format

    hh = minutes // 60
    mm = minutes % 60
    print(hh,mm)
    return

n= int(input())
for i in range(0,n):
    s = input()
    minutes=durationDifference(s)
    outputTimeFormat(minutes)
```

```
2
1 44 2 14
0 30
2 42 8 23
5 41
```

Problem : Play with Numbers

- line 1 : array size(n), no of queries
- line 2 :

In [ ]:
```

```

In [ ]:
```

```

character and number count in a string

- Problem no 2 in Assesment test

In [1]:
```
def digitCountforAlphaNum(s):
    charcount=0
    digitcount=0
    for i in range(0,len(s)):
        if((s[i]>='a' and s[i]<='z') or (s[i]>='A' and s[i]<='Z')):
            charcount +=1
        elif(s[i]>='0' and s[i]<='9' ):
            digitcount += 1
    print(charcount)
    print(digitcount)

s= input()
digitCountforAlphaNum(s)
```

```
srav!234
4
3
```

In [3]:
```
#Another way using ASCII Values

def digitCountforAlphaNum(s):
    charcount=0
    digitcount=0
    for i in range(0,len(s)):
        if((ord(s[i])>=97 and ord(s[i])<=122) or (ord(s[i])>=65 and ord(s[i]
            charcount +=1
        elif(ord(s[i])>=48 and ord(s[i])<=57 ):
            digitcount += 1
    print(charcount)
    print(digitcount)

s= input()
digitCountforAlphaNum(s)
```

```
hii@1235
3
4
```

In [4]:
```
ord('Z')
```

Out[4]:
```
90
```

In [8]:
```
dir(str)   #To know the string methods
```

Out[8]:
```
['__add__',
 '__class__',
 '__contains__',
 '__delattr__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__getitem__',
 '__getnewargs__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__iter__',
 '__le__',
 '__len__',
 '__lt__',
 '__mod__',
 '__mul__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__rmod__',
 '__rmul__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 'capitalize',
 'casefold',
 'center',
 'count',
 'encode',
 'endswith',
 'expandtabs',
 'find',
 'format',
 'format_map',
 'index',
 'isalnum',
 'isalpha',
 'isascii',
 'isdecimal',
 'isdigit',
 'isidentifier',
 'islower',
 'isnumeric',
 'isprintable',
 'isspace',
 'istitle',
 'isupper',
 'join',
 'ljust',
 'lower',
 'lstrip',
 'maketrans',
 'partition',
 'replace',
 'rfind',
 'rindex',
 'rjust',
 'rpartition',
 'rsplit',
 'rstrip',
 'split',
 'splitlines',
 'startswith',
 'strip',
 'swapcase',
 'title',
 'translate',
 'upper',
 'zfill']
```

Using upper() and lower()

In [12]:

```python
def digitCountforAlphaNum(s):
    charcount=0
    digitcount=0
    for i in range(0,len(s)):
        if(s[i].islower() or s[i].isupper()):   #s[i].isalpha()
            charcount +=1
        elif(s[i].isnumeric() ):       #s[i].isnumeric()
            digitcount += 1
    print(charcount)
    print(digitcount)

s= input()
digitCountforAlphaNum(s)
```

```
aps123
3
3
```

Factors of a number to find perfect number

In [13]:

```python
def factors(n):
    for i in range(1,n):
        if(n%i == 0):
            print(i,end= " ")
factors(28)
```

```
1 2 4 7 14
```

In [4]:
```
def factorsSum(n):
    sum1= 0
    for i in range(1,n):
        if n%i ==0:
            sum1=sum1+i
    if (n==sum1):
           return "YES"
    else:
           return "NO"
testcases=int(input())
for i in range(testcases):
    n=int(input())
    print(factorsSum(n))
```

```
3
6
YES
5
NO
28
YES
```

Highest remainder

In [5]:
```
def highestRem(n):
    rem=0
    for i in range(1,n):
        r=n%i
        if r>rem:
            rem=r
            j=i
    return j

t=int(input())
for i in range(t):
    n=int(input())
    print(highestRem(n))
```

```
2
4
3
5
3
```

In [12]:
```
### Prime numbers
```

In [13]:
```
n=int(input())
def prime_number(n):
    c=0
    for i in range(1,n+1):
        if(n%i==0):
            c +=1
    if(c==2):
        print(n)
prime_number(n)
```

```
10
```

In [19]:
```
def prime_number(n1):
    fc=0
    for k in range(2,n1+1):
        n=k
        c=0
        for i in range(1,n+1):
            if(n%i==0):
                c +=1
        if(c==2):
            fc=fc+1
        if(fc==2):
            print("YES")
        else:
            print("NO")

prime_number(7)
```

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

In [ ]:
```
#Using is prime to find the prime factors  (Special number)

def is_prime(n1):
    fc=0
    for k in range(2,n1+1):
        n=k
        c=0
        for i in range(1,n+1):
            if(n%i==0):
                c +=1
        if(c==2):
            fc=fc+1
        if(fc==2):
            print("YES")
        else:
            print("NO")

prime_number(7)

```