Python

2018年10月21日 星期日 16:20

- 中文解决方法为只要在文件开头加入 # –*– coding: UTF–8 –*– 或者 #coding=utf–8 就行了

- 基本输入输出
  x = float(input("Enter the value of x: "))
  Python3 中没有raw_input()，只有input()
  Print 输出方式：
  EG:

- print ([[a,b,c] for a in range(x+1) for b in range(y+1) for c in range(z+1) if a+b+c != n])
- second_highest = sorted(list(set([marks for name, marks in marksheet])))[1]
  print('\n'.join([a for a,b in sorted(marksheet) if b == second_highest]))

==========

- enumerate() 和 zip() 函数:
  enumerate() returns a tuple containing a count (from *start* which defaults to 0)

```
>>> seasons = ['Spring', 'Summer', 'Fall', 'Winter']
>>> list(enumerate(seasons))
[(0, 'Spring'), (1, 'Summer'), (2, 'Fall'), (3, 'Winter')]
>>> list(enumerate(seasons, start=1))
[(1, 'Spring'), (2, 'Summer'), (3, 'Fall'), (4, 'Winter')]
animals = ['cat', 'dog', 'monkey']
for idx, animal in enumerate(animals):
    print('#%d: %s' % (idx + 1, animal))
# Prints "#1: cat", "#2: dog", "#3: monkey", each on its own line
```

- print("my name is %s.I am %d years old" % ('Shixiaolou',4))

- Xrange and range difference :
  Python3里面直接用range就好
  range creates a list, so if you do `range(1, 10000000)` it creates a list in memory with 9999999 elements.
  xrange is a sequence object that evaluates lazily.
  This is true, but in Python 3, `.range()` will be implemented by the Python 2 `.xrange()`. If you need to actually generate the list, you will need to do:
  list(range(1,100))

==========

- Map() 函数会根据提供的函数对指定的序列做映射

```
>>>def square(x) : # 计算平方数
... return x ** 2 ...
>>> map(square, [1,2,3,4,5]) # 计算列表各个元素的平方
[1, 4, 9, 16, 25]
>>> map(lambda x: x ** 2, [1, 2, 3, 4, 5]) # 使用 lambda 匿名函数
[1, 4, 9, 16, 25]
# 提供了两个列表，对相同位置的列表数据进行相加
>>> map(lambda x, y: x + y, [1, 3, 5, 7, 9], [2, 4, 6, 8, 10])
[3, 7, 11, 15, 19]
```

==========

- Python有5个标准数据类型：numbers(数字), string(字符串) , list(列表) , tuple(元组) , dictionary()
  总结：花括号是字典，中括号是列表，小括号是元组

Python支持四种不同的数字类型：

- int（有符号整型）
- long（长整型[也可以代表八进制和十六进制]）
- float（浮点型）
- complex（复数）

==========

Python列表 list

列表可以完成大多数集合类的数据结构实现。它支持字符，数字，字符串甚至可以包含列表（即嵌套）。

列表用[ ]标识，是 python 最通用的复合数据类型。可以用append()来添加列表项
列表中值的切割也可以用到变量**[头下标:尾下标]**，就可以截取相应的列表，从左到右索引默认 0 开始，从右到左索引默认 -1 开始，下标可以为空表示取到头或尾。
加号+是列表连接运算符，星号*是重复操作。

```
list = [ 'runoob', 786 , 2.23, 'john', 70.2 ]
tinylist = [123, 'john']
print list # 输出完整列表
print list[0] # 输出列表的第一个元素
print list[1:3] # 输出第二个至第三个元素
```

plus， list()函数正常是把元组转换成列表

```
Eg.: aTuple = (123, 'xyz', 'zara', 'abc');
aList = list(aTuple)
print "列表元素 : ", aList
```

==========

set:

Plus, set()函数穿件一个无序不重复元素集，可进行关系测试，删除重复数据，还可以计算交集、差集、并集等。
EG.

```
class set([iterable])
```

参数说明：iterable -- 可迭代对象对象；

```
>>>x = set('runoob')
>>> y = set('google')
>>> x, y (set(['b', 'r', 'u', 'o', 'n']), set(['e', 'o', 'g',
'l'])) # 重复的被删除
>>> x & y # 交集 set(['o'])
>>> x | y # 并集 set(['b', 'e', 'g', 'l', 'o', 'n', 'r', 'u'])
>>> x - y # 差集 set(['r', 'b', 'u', 'n'])
```

```
print(len(set([input() for x in range(int(input()))])))
```

set更多对list类型进行操作：

```
k,arr = int(input()),list(map(int, input().split()))
myset = set(arr)
```

==========

Python 元组

元组用()标识。内部元素用逗号隔开。但是元组不能二次赋值，相当于只读列表。

```
tuple = ( 'runoob', 786 , 2.23, 'john', 70.2 )
print tuple[0] # 输出元组的第一个元素
print tuple[1:3] # 输出第二个至第三个的元素
```

- Cmp(tuple1,tuple2)
- Len(tuple)
- Max(tuple)
- Min(tuple)
- Tuple(seq) 将列表转换为元组

==========

Python 字典

字典(dictionary)是除列表以外python之中最灵活的内置数据结构类型。列表是有序的对象集合，字典是无序的对象集合。
两者之间的区别在于：字典当中的元素是通过键来存取的，而不是通过偏移存取。
字典用"{ }"标识。字典由索引(key)和它对应的值value组成。

```
dict = {}
dict['one'] = "This is one"
dict[2] = "This is two"
tinydict = {'name': 'john','code':6734, 'dept': 'sales'}
print dict['one'] # 输出键为'one' 的值
print dict[2] # 输出键为 2 的值
```

Str(x) 将对象x转换为字符串； eval(str) 用来计算在字符串汇总的有效Python表达式，并返回一个对象

Lambda 表达式：匿名函数

Sorted()函数

sort 与 sorted 区别：
sort 是应用在 list 上的方法，sorted 可以对所有可迭代的对象进行排序操作。
list 的 sort 方法返回的是对已经存在的列表进行操作，无返回值，而内建函数 sorted 方法返回的是一个新的 list，而不是在原来的基础上进行的操作。

```
>>>a = [5,7,6,3,4,1,2]
>>> b = sorted(a) # 保留原列表
>>> a [5, 7, 6, 3, 4, 1, 2]
>>> b [1, 2, 3, 4, 5, 6, 7]
>>> L=[('b',2),('a',1),('c',3),('d',4)]
>>> sorted(L, cmp=lambda x,y:cmp(x[1],y[1])) # 利用cmp函数
[('a', 1), ('b', 2), ('c', 3), ('d', 4)]
>>> sorted(L, key=lambda x:x[1]) # 利用key
[('a', 1), ('b', 2), ('c', 3), ('d', 4)]
>>> students = [('john', 'A', 15), ('jane', 'B', 12), ('dave',
'B', 10)]
>>> sorted(students, key=lambda s: s[2]) # 按年龄排序
[('dave', 'B', 10), ('jane', 'B', 12), ('john', 'A', 15)]
>>> sorted(students, key=lambda s: s[2], reverse=True) # 按降序
[('john', 'A', 15), ('jane', 'B', 12), ('dave', 'B', 10)]

>>>print ('\n'.join(sorted(r,key = int)))
```

有的时候可能不对，加一个int 排序

==========

Python format()

用法：

它通过{}和:来代替传统%方式

1、使用位置参数

要点：从以下例子可以看出位置参数不受顺序约束，且可以为{},只要format里有相对应的参数值即可,参数索引从0开，传入位置参数列表可用*列表

```
>>> li = ['hoho',18]
>>> 'my name is {} ,age {}'.format('hoho',18)
'my name is hoho ,age 18'
>>> 'my name is {1} ,age {0}'.format(10,'hoho')
'my name is hoho ,age 10'
>>> 'my name is {1} ,age {0} {1}'.format(10,'hoho')
'my name is hoho ,age 10 hoho'
>>> 'my name is {} ,age {}'.format(*li)
'my name is hoho ,age 18'
```

2、使用关键字参数

要点：关键字参数值要对得上，可用字典当关键字参数传入值，字典前加**即可

```
>>> hash = {'name':'hoho','age':18}
>>> 'my name is {name},age is {age}'.format(name='hoho',age=19)
'my name is hoho,age is 19'
>>> 'my name is {name},age is {age}'.format(**hash)
'my name is hoho,age is 18'
```

3、填充与格式化

:[填充字符][对齐方式 <^>][宽度]

```
>>> '{0:*>10}'.format(10)  ##右对齐
'********10'
>>> '{0:*<10}'.format(10)  ##左对齐
'10********'
>>> '{0:*^10}'.format(10)  ##居中对齐
'****10****'
```

4、精度与进制

```
>>> '{0:.2f}'.format(1/3)
'0.33'
>>> '{0:b}'.format(10)    #二进制
'1010'
>>> '{0:o}'.format(10)     #八进制
'12'
>>> '{0:x}'.format(10)     #16进制
'a'
>>> '{:,}'.format(12369132698)  #千分位格式化
'12,369,132,698'
```

5、使用索引

```
>>> li
['hoho', 18]
>>> 'name is {0[0]} age is {0[1]}'.format(li)
'name is hoho age is 18
```

```
print("{} H, {} M".format(h,m))
```

==========

eval() 函数用来执行一个字符串表达式，并返回表达式的值。

```
>>>x = 7
```

Python math 模块提供了许多对浮点数的数学运算函数。
Python cmath 模块包含了一些用于复数运算的函数。
cmath 模块的函数跟 math 模块函数基本一致，区别是 cmath 模块运算的是复数，math 模块运算的是数学运算。
要使用 math 或 cmath 函数必须先导入：

```
>>> eval( '3 * x' )
21
>>> eval('pow(2,2)')
4
>>> eval('2 + 2')
4
>>> n=81
>>> eval("n + 4")
85
```

==========

Python math 模块提供了许多对浮点数的数学运算函数。
Python cmath 模块包含了一些用于复数运算的函数。
cmath 模块的函数跟 math 模块函数基本一致，区别是 cmath 模块运算的是复数，math 模块运算的是数学运算。
要使用 math 或 cmath 函数必须先导入：

Input Format

A single line containing the complex number. Note: complex() function can be used in python to convert the input as a complex number.

Constraints

Given number is a valid complex number

Output Format

Output two lines:

The first line should contain the value of.

The second line should contain the value of .

Sample Input

1+2j

Sample Output

2.23606797749979
1.1071487177940904

Note: The output should be correct up to 3 decimal places.

```
import cmath
if    name    == '   main__':
r = complex(input())
print(cmath.polar(r)[0])
print(cmath.polar(r)[1])
```

==========

Python2：
raw_input 直接读取控制台的输入
input()读取一个合法的Python表达式

字符串的逆置：

```
print(s)
s=s[::-1]
print(s)
```

Python 有两种除法操作符，一种是单斜杠：用于传统除法，另一种双斜杠：用于浮点数除法，其结果进行四舍五入。

Python字符串格式化：
Python 支持格式化字符串的输出 。尽管这样可能会用到非常复杂的表达式，但最基本的用法是将一个值插入到一个有字符串格式符 %s 的字符串中。

```
print "My name is %s and weight is %d kg!" % ('Zara', 21)
```

Python单引号，双引号和三引号的区别：
表示字符串是不是单引号和双引号都可以

Python 多行输出：

```
print('''
 so this is an
 multi-line
 pirnt,pretty cool

''')
```

输入三个整数x,y,z，请把这三个数由小到大输出。

```
l=[]
for i in range(3):
    x= raw_input('Integer:\n')
    l.append(x)
l.sort()
print l
```

将一个列表中的数据复制到另一个列表中

```
import copy
a=[1,2,3]
b=copy.copy(a)
print(b)

l=[1,2,3,4]
```

```
p=[]
for i in range(len(l)):
   p.append(l[i])
print(p)
```

斐波那契数列：

```
def fib(n):
   a,b=1,1
   for i in range(n-1):
      a,b=b,a+b
   return a
print(fib(10))
```

read()每次读取整个文件
readlines()自动将文件内容分析成一个个行的列表
readline()每次只读取一行，速度比readlines()慢很多

python语言最常见的括号有三种，分别是：小括号( )、中括号[ ]和大括号也叫做花括号{ }。其作用也各不相同，分别用来代表不同的python基本内置数据类型。

1、python中的小括号( )：代表tuple元组数据类型，元组是一种不可变序列。创建方法很简单，大多时候都是用小括号括起来的。

```
1
2
3
4
5
6
7
8
9
>>> tup = (1,2,3)
>>> tup
(1, 2, 3)
>>>
>>> ()#空元组
()
>>>
>>> 55,#一个值的元组
(55,)
```

2、python中的中括号[ ]：代表list列表数据类型，列表是一种可变的序列。其创建方法即简单又特别，像下面一样：

```
1
2
>>> list('python')
['p', 'y', 't', 'h', 'o', 'n']
```

3、python大括号{ }花括号：代表dict字典数据类型，字典是由键对值组组成。冒号':'分开键和值，逗号','隔开组。用大括号创建的方法如下：

```
1
2
3
4
```

```
>>> dic={'jon':'boy','lili':'girl'}
>>> dic
{'lili': 'girl', 'jon': 'boy'}
```

总结：花括号是字典，中括号是列表，小括号是元组

Numpy

2018年10月23日 星期二 09:51

C = A * B 代替一个个数组里的元素相乘

```
import numpy as np
a = np.array([[1,2], [3, 4], [5, 6]])
# An example of integer array indexing.
# The returned array will have shape (3,) and
print(a[[0, 1, 2], [0, 1, 0]])  # Prints "[1 4 5]"
# The above example of integer array indexing is equivalent to this:
print(np.array([a[0, 0], a[1, 1], a[2, 0]]))  # Prints "[1 4 5]"
# When using integer array indexing, you can reuse the same
# element from the source array:
print(a[[0, 0], [1, 1]])  # Prints "[2 2]"
# Equivalent to the previous integer array indexing example
print(np.array([a[0, 1], a[0, 1]]))  # Prints "[2 2]"
```

python 的sum函数axis=0和axis=1
没有axis参数表示全部相加，axis = 0表示按列相加，axis = 1表示按照行的方向相加
加入axis=1就是将每一个矩阵的每一行向量相加

numpy的random模块：

```
>>> np.random.rand(3,2)
array([[ 0.14022471,  0.96360618],  #random
       [ 0.37601032,  0.25528411],  #random
       [ 0.49313049,  0.94909878]]) #random
```

Numpy.array() is used to convert a list into into a NumPy array.

```
import numpy as np
# Create a new array from which we will select elements
a = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])
print(a)  # prints "array([[ 1,  2,  3],
          #               [ 4,  5,  6],
          #               [ 7,  8,  9],
          #               [10, 11, 12]])"
# Create an array of indices
b = np.array([0, 2, 0, 1])
# Select one element from each row of a using the indices in b
print(a[np.arange(4), b])  # Prints "[ 1  6  7 11]"
# Mutate one element from each row of a using the indices in b
a[np.arange(4), b] += 10
print(a)  # prints "array([[11,  2,  3],
          #               [ 4,  5, 16],
          #               [17,  8,  9],
          #               [10, 21, 12]])
```

```
import numpy as np
x = np.array([[1,2],[3,4]])
print(np.sum(x))  # Compute sum of all elements; prints "10"
print(np.sum(x, axis=0))  # Compute sum of each column; prints "[4 6]"
print(np.sum(x, axis=1))  # Compute sum of each row; prints "[3 7]"
```

```
import numpy as np
np.sum(0,1,2],[2,1,3,axis = 1)
```

输出 array([3,6])

C = np.array([[0,2,1],[3,5,6],[0,1,1]])
print C.sum()
Print C.sum(axis = 0)
Print C.sum(axis = 1)
结果计算分别是：19，[3,8,8], [3,14,2]

Axis = 0 按照行求和
Axis = 1 按照列求和

Numpy.tile([0,0] , 5) # 在列方向上重复[0,0]5次，只有1行
Numpy.tile ([0,0],(1,3))

```
>>> a = np.array([0, 1, 2])
>>> np.tile(a, 2)
array([0, 1, 2, 0, 1, 2])
>>> np.tile(a, (2, 2))
array([[0, 1, 2, 0, 1, 2],
       [0, 1, 2, 0, 1, 2]])
>>> np.tile(a, (2, 1, 2))
array([[[0, 1, 2, 0, 1, 2]],
       [[0, 1, 2, 0, 1, 2]]])

>>>
>>> b = np.array([[1, 2], [3, 4]])
>>> np.tile(b, 2)
array([[1, 2, 1, 2],
       [3, 4, 3, 4]])
>>> np.tile(b, (2, 1)) #行数扩展为2倍，列数不变
array([[1, 2],
       [3, 4],
       [1, 2],
       [3, 4]])

>>>
>>> c = np.array([1,2,3,4])
>>> np.tile(c,(4,1))
array([[1, 2, 3, 4],
       [1, 2, 3, 4],
       [1, 2, 3, 4],
       [1, 2, 3, 4]])
```

Numpy 中数字反过来，有两种方式：
一种是 np.flipud(A)
另一种是

```
def arrays(arr):
  #revrser array first, convert to float array with numpy
  return (numpy.array(arr[::-1], float))
```

Sample Input

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

Sample Output

```
[[1 3]
[2 4]]
[1 2 3 4]
```

```
import numpy
n, m = map(int, input().split())
array = numpy.array([input().strip().split() for    in range(n)], int)
print(numpy.transpose(array)) # transpose()得到矩阵的倒置
print(array.flatten())  # 把这个矩阵铺平下来
```

用numpy连接数组

Two or more arrays can be concatenated together using the *concatenate* function with a tuple of the arrays to be joined:

```
import numpy

array_1 = numpy.array([1,2,3])
array_2 = numpy.array([4,5,6])
array_3 = numpy.array([7,8,9])

print numpy.concatenate((array_1, array_2, array_3))


#Output
[1 2 3 4 5 6 7 8 9]
```

If an array has more than one dimension, it is possible to specify the axis along which multiple arrays are concatenated. By default, it is along the first dimension.

```
import numpy


array_1 = numpy.array([[1,2,3],[0,0,0]])
array_2 = numpy.array([[0,0,0],[7,8,9]])


print numpy.concatenate((array_1, array_2), axis = 1)   # axis =1 代表行方向


#Output
[[1 2 3 0 0 0]
[0 0 0 7 8 9]]
```

用numpy表示全0或者全1

Sample Input 0

```
3 3 3
```

Sample Output 0

```
[[[0 0 0]
  [0 0 0]
  [0 0 0]]


[[0 0 0]
  [0 0 0]
  [0 0 0]]


[[0 0 0]
  [0 0 0]
  [0 0 0]]]
[[[1 1 1]
  [1 1 1]
  [1 1 1]]


[[1 1 1]
  [1 1 1]
  [1 1 1]]


[[1 1 1]
  [1 1 1]
```

[1 1 1]]]

Explanation 0

Print the array built using numpy.zeros and numpy.ones tools and you get the result as shown.

转化为tuple因为numpy.zeros( )只能处理元组类型

```
import numpy
nums = tuple(map(int, input().split())) #输入的个数不一定的时候使用
print (numpy.zeros(nums, dtype = numpy.int))
print (numpy.ones(nums, dtype = numpy.int))
```

dot

The *dot* tool returns the dot product of two arrays.

```
import numpy

A = numpy.array([ 1, 2 ])
B = numpy.array([ 3, 4 ])

print numpy.dot(A, B)       #Output : 11
```

cross

The *cross* tool returns the cross product of two arrays.

```
import numpy

A = numpy.array([ 1, 2 ])
B = numpy.array([ 3, 4 ])

print numpy.cross(A, B)     #Output : -2
```

1.Python Find a string

In this challenge, the user enters a string and a substring. You have to print the number of times that the substring occurs in the given string. String traversal will take place from left to right, not from right to left.

NOTE: String letters are case-sensitive.

Input Format

The first line of input contains the original string. The next line contains the substring.

Constraints

Each character in the string is an ascii character.

Output Format

Output the integer number indicating the total number of occurrences of the substring in the original string.

Sample Input

```
ABCDCDC
CDC
```

Sample Output

2

Concept

Some string processing examples, such as these, might be useful.
There are a couple of new concepts:
In Python, the length of a string is found by the function `len(s)`, where is the string.
To traverse through the length of a string, use a for loop:

```
for i in range(0, len(s)):
    print (s[i])
```

A range function is used to loop over some length:

```
range (0, 5)
```

Here, the range loops over to . is excluded.

Python strip() 函数用于移除字符串头尾指定的字符（默认为空格或换行符）或字符序列。正常strip()指的是移除空格。

```python
def count_substring(string, sub_string):
    count=0
    for i in range(0, len(string)-len(sub_string)+1):
        if string[i] == sub_string[0]:
            flag=1
            for j in range (0, len(sub_string)):
                if string[i+j] != sub_string[j]:
                    flag=0
                    break
            if flag==1:
                count += 1
    return count


if __name__ == '__main__':
    string = input().strip()
    sub_string = input().strip()

    count = count_substring(string, sub_string)
    print(count)
```

2.Python Merge the Tools

Consider the following:

- A string, s, of length n where $s = c_0c_1 \ldots c_{n-1}$.
- An integer, k, where k is a factor of n.

We can split s into $\frac{n}{k}$ subsegments where each subsegment, t_i, consists of a contiguous block of k characters in s. Then, use each t_i to create string u_i such that:

- The characters in u_i are a subsequence of the characters in t_i.
- Any repeat occurrence of a character is removed from the string such that each character in u_i occurs exactly once. In other words, if the character at some index j in t_i occurs at a previous index $< j$ in t_i, then do not include the character in string u_i.

Given s and k, print $\frac{n}{k}$ lines where each line i denotes string u_i.

Input Format

The first line contains a single string denoting s.

The second line contains an integer, k, denoting the length of each subsegment.

Constraints

- $1 \leq n \leq 10^4$, where n is the length of s
- $1 \leq k \leq n$
- It is guaranteed that n is a multiple of k.

Output Format

Print $\frac{n}{k}$ lines where each line i contains string u_i.

zip() 函数用于将可迭代的对象作为参数，将对象中对应的元素打包成一个个元组，然后返回由这些元组组成的对象，这样做的好处是节约了不少的内存。

```
1
2 >>>a = [1,2,3]
3 >>> b = [4,5,6]
4 >>> c = [4,5,6,7,8]
5 >>> zipped = zip(a,b)     # 返 回 一 个 对 象
6 >>> zipped
7 <zip object at 0x103abc288>
```

```
>>> list(zipped)  # list() 转换为列表
[(1, 4), (2, 5), (3, 6)]
>>> list(zip(a,c))              # 元素个数与最短的列表一致
[(1, 4), (2, 5), (3, 6)]

>>> a1, a2 = zip(*zip(a,b))          # 与 zip 相反，zip(*) 可理解为解压，返回二维矩阵式
>>> list(a1)
[1, 2, 3]
>>> list(a2)
[4, 5, 6]
>>>

```

```
# 以下实例展示了setdefault()方法的使用方法
```

```
#!/usr/bin/python3

dict = {'Name': 'Runoob', 'Age': 7}

print ("Age 键的值为 : %s" %  dict.setdefault('Age', None))
print ("Sex 键的值为 : %s" %  dict.setdefault('Sex', None))
print ("新字典为：", dict)
```

```
Age 键的值为 : 7
Sex 键的值为 : None
新字典为：  {'Age': 7, 'Name': 'Runoob', 'Sex': None}
```

Sample Input

```
AABCAAADA
3
```

Sample Output

```
AB
CA
AD
```

Explanation

String s is split into $\frac{n}{k} = \frac{9}{3} = 3$ equal parts of length $k = 3$. We convert each t_i to u_i by removing any subsequent occurrences non-distinct characters in t_i:

1. $t_0 = \texttt{"AAB"} \rightarrow u_0 = \texttt{"AB"}$
2. $t_1 = \texttt{"CAA"} \rightarrow u_1 = \texttt{"CA"}$
3. $t_2 = \texttt{"ADA"} \rightarrow u_2 = \texttt{"AD"}$

We then print each u_i on a new line.

```
# dict()使 用 方 法
>>>dict()                          # 创 建 空 字 典
{}
>>> dict(a='a', b='b', t='t')      # 传 入 关 键 字
{'a': 'a', 'b': 'b', 't': 't'}
>>> dict(zip(['one', 'two', 'three'], [1, 2, 3]))   # 映 射 函 数 方 式 来 构 造 字 典
{'three': 3, 'two': 2, 'one': 1}
>>> dict([('one', 1), ('two', 2), ('three', 3)])    # 可 迭 代 对 象 方 式 来 构 造 字 典
{'three': 3, 'two': 2, 'one': 1}
>>>
```

```
def merge_the_tools(string, k):
    #S = input()
    #N = int(input())
    for part in zip(*[iter(string)] * k):
        d = dict()
        print(''.join([ d.setdefault(c, c) for c in part if c not in d ]))


if __name__ == '__main__':
    string, k = input(), int(input())
    merge_the_tools(string, k)
```

3.Python Nested Lists

Given the names and grades for each student in a Physics class of students, store them in a nested list and print the name(s) of any student(s) having the second lowest grade.

Note: If there are multiple students with the same grade, order their names alphabetically and print each name on a new line.

Input Format

The first line contains an integer, , the number of students.
The subsequent lines describe each student over lines; the first line contains a student's name, and the second line contains their grade.

Constraints

There will always be one or more students having the second lowest grade.
Output Format

Print the name(s) of any student(s) having the second lowest grade in Physics; if there are multiple students, order their names alphabetically and print each one on a new line.

Sample Input 0

5
Harry
37.21
Berry
37.21
Tina
37.2
Akriti
41
Harsh
39

Sample Output 0

Berry
Harry

Explanation 0

There are students in this class whose names and grades are assembled to build the following list:

python students = [['Harry', 37.21], ['Berry', 37.21], ['Tina', 37.2], ['Akriti', 41], ['Harsh', 39]]

The lowest grade of belongs to Tina. The second lowest grade of belongs to both Harry and Berry, so we

order their names alphabetically and print each name on a new line.

```
if __name__ == '__main__':
    marksheet =[]
    for _ in range(int(input())):
        name = input()
        score = float(input())
        #marksheet = []
        marksheet.append([name,score])
    second_highest = sorted(list(set([marks for name, marks in marksheet])))
[1] # set very important because we want to find the second largest
    print('\n'.join([a for a,b in sorted(marksheet) if b == second_highest]))


```

4.Python String Formatting

Given an integer, n, print the following values for each integer i from 1 to n:

1. Decimal
2. Octal
3. Hexadecimal (capitalized)
4. Binary

The four values must be printed on a single line in the order specified above for each i from 1 to n. Each value should be space-padded to match the width of the binary value of n.

Input Format

A single integer denoting n.

Constraints

- $1 \leq n \leq 99$

Output Format

Print n lines where each line i (in the range $1 \leq i \leq n$) contains the respective decimal, octal, capitalized hexadecimal, and binary values of i. Each printed value must be formatted to the width of the binary value of n

Sample Input

```
17
```

Sample Output

```
 1     1     1     1
 2     2     2    10
 3     3     3    11
 4     4     4   100
 5     5     5   101
 6     6     6   110
 7     7     7   111
 8    10     8  1000
 9    11     9  1001
10    12     A  1010
11    13     B  1011
12    14     C  1100
```

```
13    13    15     D  1101
14    14    16     E  1110
15    15    17     F  1111
16    16    20    10 10000
17    17    21    11 10001
```

format() 函数的使用，str.format()，是一种格式化字符串的函数，增强了字符串格式化的功能 基本语法是通过 {} 和：来代替以前的 % 。

```
>>>"{} {}".format("hello", "world")    # 不设置指定位置，按默认顺序
'hello world'

>>> "{0} {1}".format("hello", "world")  # 设置指定位置
'hello world'

>>> "{1} {0} {1}".format("hello", "world")  # 设置指定位置
'world hello world'
```

```
>>> print("{:.2f}".format(3.1415926)); # str.format() 格式化数字的多种方法：
3.14
>>> "The sum of 1 + 2 is {0}".format(1+2)
'The sum of 1 + 2 is 3'
```

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-

print("网站名：{name}, 地址 {url}".format(name="菜鸟教程
", url="www.runoob.com"))

# 通过字典设置参数
site = {"name": "菜鸟教程", "url": "www.runoob.com"}
print("网站名：{name}, 地址 {url}".format(**site))

# 通过列表索引设置参数
my_list = ['菜鸟教程', 'www.runoob.com']
print("网站名：{0[0]}, 地址 {0[1]}".format(my_list))  # "0" 是必须的
```

```
def print_formatted(number):
    width = len("{0:b}".format(number))
    for i in range(1,number+1):
        print ("{0:{width}d} {0:{width}o} {0:{width}X} {0:
{width}b}".format(i,width=width))

  # here "0" represents that this string starts from the beginning for each line

if __name__ == '__main__':
    n = int(input())
    print_formatted(n)
```

5.Python Set.discard(), . remove()&.pop()

.remove(x)

This operation removes element x from the set.

If element x does not exist, it raises a KeyError.

The .remove(x) operation returns None.

Example

```
>>> s = set([1, 2, 3, 4, 5, 6, 7, 8, 9])
>>> s.remove(5)
>>> print s
set([1, 2, 3, 4, 6, 7, 8, 9])
>>> print s.remove(4)
None
>>> print s
set([1, 2, 3, 6, 7, 8, 9])
>>> s.remove(0)
KeyError: 0
```

.discard(x)

This operation also removes element x from the set.

If element x does not exist, it **does not** raise a KeyError.

The .discard(x) operation returns None.

Example

```
>>> s = set([1, 2, 3, 4, 5, 6, 7, 8, 9])
>>> s.discard(5)
>>> print s
set([1, 2, 3, 4, 6, 7, 8, 9])
>>> print s.discard(4)
None
>>> print s
set([1, 2, 3, 6, 7, 8, 9])
>>> s.discard(0)
>>> print s
set([1, 2, 3, 6, 7, 8, 9])
```

.pop()

This operation removes and return an arbitrary element from the set.

If there are no elements to remove, it raises a KeyError.

Example

```
>>> s = set([1])
>>> print s.pop()
1
>>> print s
set([])
>>> print s.pop()
KeyError: pop from an empty set
```

Task

You have a non-empty set s, and you have to execute N commands given in N lines.

The commands will be pop, remove and discard.

Input Format

The first line contains integer n, the number of elements in the set s.

The second line contains n space separated elements of set s. All of the elements are non-negative integers, less than or equal to 9.

The third line contains integer N, the number of commands.

The next N lines contains either pop, remove and/or discard commands followed by their associated value.

Constraints

$0 < n < 20$

$0 < N < 20$

Output Format

Print the sum of the elements of set s on a single line.

Sample Input

```
9
1 2 3 4 5 6 7 8 9
10
pop
remove 9
discard 9
```

```
discard 8
remove 7
pop
discard 6
remove 5
pop
discard 5
```

Sample Output

```
4
```

Explanation

After completing these 10 operations on the set, we get set$([4])$. Hence, the sum is 4.

Note: Convert the elements of set s to integers while you are assigning them. To ensure the proper input of the set, we have added the first two lines of code to the editor.

解法一

```
n = int(input())

s = set(map(int, input().split()))

for i in range(int(input())):

 eval('s.{0}({1})'.format(*input().split()+['']))# it means like s.remove(9)

print(sum(s))
```

解释：eval() 函数用来执行一个字符串表达式，并返回表达式的值。

```
>>>x = 7
>>> eval( '3 * x' )
21
>>> eval('pow(2,2)')
4
>>> eval('2 + 2')
4
>>> n=81
>>> eval("n + 4")
85
```

解法二

```python
n = int(raw_input())
s = set([int(x) for x in raw_input().strip().split()]) # strip() can be added or not added, both OK
for _ in range(int(raw_input())):

    a = list(raw_input().strip().split())

    if a[0] == 'pop':
        s.pop()
    elif a[0] == 'discard':
        s.discard(int(a[1]))
    else:
        s.remove(int(a[1]))

print sum(s)
```

6.Python Text Wrap

You are given a string and width .
Your task is to wrap the string into a paragraph of width .

Input Format

The first line contains a string, .
The second line contains the width, .

Sample Input 0

```
ABCDEFGHIJKLIMNOQRSTUVWXYZ
4
```

Sample Output 0

```
ABCD
EFGH
IJKL
IMNO
QRST
UVWX
YZ
```

值得注意的还是 join()函数的使用，挺方便的，用这种形式来实现换行\n

还有留意这里range() 的使用，使用了3个参数

```python
import textwrap

def wrap(string, max_width):
    return "\n".join(string[i:i+max_width] for i in range(0, len(string), max_width))

if __name__ == '__main__':
    string, max_width = input(), int(input())
    result = wrap(string, max_width)
    print(result)
```

7.Python Set Mutations

We have seen the applications of union, intersection, difference and symmetric difference operations, but these operations do not make any changes or mutations to the set.

We can use the following operations to create mutations to a set:

.update() or `|=`

Update the set by adding elements from an iterable/another set.

```
>>> H = set("Hacker")
>>> R = set("Rank")
>>> H.update(R)
>>> print H
set(['a', 'c', 'e', 'H', 'k', 'n', 'r', 'R'])
```

.intersection_update() or `&=`

Update the set by keeping only the elements found in it and an iterable/another set.

```
>>> H = set("Hacker")
>>> R = set("Rank")
>>> H.intersection_update(R)
>>> print H
set(['a', 'k'])
```

.difference_update() or `-=`

Update the set by removing elements found in an iterable/another set.

```
>>> H = set("Hacker")
>>> R = set("Rank")
>>> H.difference_update(R)
>>> print H
set(['c', 'e', 'H', 'r'])
```

.symmetric_difference_update() or `^=`

Update the set by only keeping the elements found in either set, but not in both.

```
>>> H = set("Hacker")
>>> R = set("Rank")
```

```
>>> H.symmetric_difference_update(R)
>>> print H
set(['c', 'e', 'H', 'n', 'r', 'R'])
```

TASK

You are given a set A and N number of other sets. These N number of sets have to perform some specific mutation operations on set A.

Your task is to execute those operations and print the sum of elements from set A.

Input Format

The first line contains the number of elements in set A.

The second line contains the space separated list of elements in set A.

The third line contains integer N, the number of other sets.

The next $2 * N$ lines are divided into N parts containing two lines each.

The first line of each part contains the space separated entries of the operation name and the length of the other set.

The second line of each part contains space separated list of elements in the other set.

$0 <$ len(set(**A**)) < 1000

$0 <$ len(otherSets) < 100

$0 < N < 100$

Output Format

Output the sum of elements in set A.

Sample Input

```
16
1 2 3 4 5 6 7 8 9 10 11 12 13 14 24 52
4
intersection_update 10
2 3 5 6 8 9 1 4 7 11
update 2
55 66
symmetric_difference_update 5
22 7 35 62 58
difference_update 7
11 22 35 55 58 62 66
```

Sample Output

```
38
```

Explanation

After the first operation, (intersection_update operation), we get:

set $A = set([1, 2, 3, 4, 5, 6, 7, 8, 9, 11])$

After the second operation, (update operation), we get:

set $A = set([1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 55, 66])$

After the third operation, (symmetric_difference_update operation), we get:

set $A = set([1, 2, 3, 4, 5, 6, 8, 9, 11, 22, 35, 55, 58, 62, 66])$

After the fourth operation, (difference_update operation), we get:

set $A = set([1, 2, 3, 4, 5, 6, 8, 9])$

The sum of elements in set A after these operations is 38.

eval()函数用来执行一个字符串表达式，并返回表达式的值。

```
>>>x = 7
>>> eval( '3 * x' )
21
>>> eval('pow(2,2)')
4
>>> eval('2 + 2')
4
>>> n=81
>>> eval("n + 4")
85
```

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
if __name__ == '__main__':
    m = int(input())
    A = set(map(int, input().split()))# After map operation, it turns out to be a vector
    n = int(input())

    for i in range(n):
        cmd, args = input().split() #args not need in this code
        B = set(map(int, input().split()))
        eval('A.'+cmd+'(B)')

    print (sum(A))
```

8. Python Triangle Quest 2

You are given a positive integer N.

Your task is to print a palindromic triangle of size N.

For example, a palindromic triangle of size 5 is:

```
1
121
12321
1234321
123454321
```

You can't take more than two lines. The first line (a for-statement) is already written for you.

You have to complete the code using exactly one print statement.

Note:

Using anything related to strings will give a score of 0.

Using more than one for-statement will give a score of 0.

Input Format

A single line of input containing the integer N.

Constraints

- $0 < N < 10$

Output Format

Print the palindromic triangle of size N as explained above.

Sample Input

```
5
```

Sample Output

```
1
121
12321
1234321
123454321
```

list 直接输出来是矩阵，所以这里加一个 * 号，代表输出里面的数字

```python
for i in range(0,int(input())):
    #print ([1, 121, 12321, 1234321, 123454321, 12345654321, 1234567654321, 123456787654321, 12345678987654321, 1234567891
0987654321][i])
    print(*(list(range(1, i)) + list(range(i, 0, -1))), sep="")
```

9.Python Integers Come In All Sizes

Integers in Python can be as big as the bytes in your machine's memory. There is no limit in size as there is: $2^{31} - 1$ (c++ int) or $2^{63} - 1$ (C++ long long int).

As we know, the result of a^b grows really fast with increasing b.

Let's do some calculations on very large integers.

Task

Read four numbers, a, b, c, and d, and print the result of $a^b + c^d$.

Input Format

Integers a, b, c, and d are given on four separate lines, respectively.

Constraints

$1 \leq a \leq 1000$

$1 \leq b \leq 1000$

$1 \leq c \leq 1000$

$1 \leq d \leq 1000$

Output Format

Print the result of $a^b + c^d$ on one line.

Sample Input

```
9
29
7
27
```

Sample Output

```
4710194409608608369201743232
```

Note: This result is bigger than $2^{63} - 1$. Hence, it won't fit in the long long int of C++ or a 64-bit integer.

Python中用pow()函数，不受数位的限制，可以研究下相关的源码

```
1 # Enter your code here. Read input from STDIN. Print output to STDOUT
```

```
if __name__== '__main__':
    a=int(input())
    b=int(input())
    c=int(input())
    d=int(input())
    e=pow(a,b)+pow(c,d)
    print(e)
```

10. Python Numpy Polynomials

poly

The poly tool returns the coefficients of a polynomial with the given sequence of roots.

```
print numpy.poly([-1, 1, 1, 10])    #Output : [ 1 -11  9 11 -10]
```

roots

The roots tool returns the roots of a polynomial with the given coefficients.

```
print numpy.roots([1, 0, -1])     #Output : [-1. 1.]
```

polyint

The polyint tool returns an antiderivative (indefinite integral) of a polynomial.

```
print numpy.polyint([1, 1, 1])     #Output : [0.33333333 0.5     1.     0.    ]
```

polyder

The polyder tool returns the derivative of the specified order of a polynomial.

```
print numpy.polyder([1, 1, 1, 1])    #Output : [3 2 1]
```

polyval

The polyval tool evaluates the polynomial at specific value.

```
print numpy.polyval([1, -2, 0, 2], 4)  #Output : 34
```

polyfit

The polyfit tool fits a polynomial of a specified order to a set of data using a least-squares approach.

```
print numpy.polyfit([0,1,-1, 2, -2], [0,1,1, 4, 4], 2)
#Output : [ 1.00000000e+00  0.00000000e+00 -3.97205465e-16]
```

The functions polyadd, polysub, polymul, and polydiv also handle proper addition, subtraction, multiplication, and division of polynomial coefficients, respectively.

Task

You are given the coefficients of a polynomial P.

Your task is to find the value of P at point x.

Input Format

The first line contains the space separated value of the coefficients in P.

The second line contains the value of x.

Output Format

Print the desired value.

Sample Input

```
1.1 2 3
0
```

Sample Output

```
3.0
```

```
1 import numpy as np
2 print(np.polyval(list(map(float,input().split())), int(input())))
3
```

11. Python Numpy Linear Algebra

The NumPy module also comes with a number of built-in routines for linear algebra calculations. These can be found in the sub-module linalg.

linalg.det

The linalg.det tool computes the determinant of an array.

```
print numpy.linalg.det([[1 , 2], [2, 1]])   #Output : -3.0
```

linalg.eig

The linalg.eig computes the eigenvalues and right eigenvectors of a square array.

```
vals, vecs = numpy.linalg.eig([[1 , 2], [2, 1]])
print vals                  #Output : [ 3. -1.]
print vecs                   #Output : [[ 0.70710678 -0.70710678]
                    #     [ 0.70710678 0.70710678]]
```

linalg.inv

The linalg.inv tool computes the (multiplicative) inverse of a matrix.

```
print numpy.linalg.inv([[1 , 2], [2, 1]])   #Output : [[-0.33333333 0.66666667]
                    #     [ 0.66666667 -0.33333333]]
```

Other routines can be found here

Task

You are given a square matrix A with dimensions NXN. Your task is to find the determinant.

Input Format

The first line contains the integer N.

The next N lines contains the N space separated elements of array A.

Output Format

Print the determinant of A.

Sample Input

```
2
1.1 1.1
1.1 1.1
```

Sample Output

```
0.0
```

```
import numpy
m = int(input())
array = numpy.array([input().strip().split() for _ in range(m)],float)
numpy.set_printoptions(legacy='1.13') #important
print(numpy.linalg.det(array))

```

12.Python Built-ins Zipped!

zip([iterable, ...])

This function returns a list of tuples. The i^{th} tuple contains the i^{th} element from each of the argument sequences or iterables.

If the argument sequences are of unequal lengths, then the returned list is truncated to the length of the shortest argument sequence.

Sample Code

```
>>> print zip([1,2,3,4,5,6],'Hacker')
[(1, 'H'), (2, 'a'), (3, 'c'), (4, 'k'), (5, 'e'), (6, 'r')]
>>>
>>> print zip([1,2,3,4,5,6],[0,9,8,7,6,5,4,3,2,1])
[(1, 0), (2, 9), (3, 8), (4, 7), (5, 6), (6, 5)]
>>>
>>> A = [1,2,3]
>>> B = [6,5,4]
>>> C = [7,8,9]
>>> X = [A] + [B] + [C]
>>>
>>> print zip(*X)
[(1, 6, 7), (2, 5, 8), (3, 4, 9)]
```

Task

The National University conducts an examination of N students in X subjects.

Your task is to compute the average scores of each student.

$$Average\ score = \frac{Sum\ of\ scores\ obtained\ in\ all\ subjects\ by\ a\ student}{Total\ number\ of\ subjects}$$

The format for the general mark sheet is:

```
Student ID →   1     2     3     4     5
Subject 1  | 89  90  78  93  80
Subject 2  | 90  91  85  88  86
Subject 3  | 91  92  83  89  90.5
           |_____________________________
Average      90  91  82  90  85.5
```

Input Format

The first line contains N and X separated by a space.

The next X lines contains the space separated marks obtained by students in a particular subject.

Constraints

$0 < N \leq 100$

$0 < X \leq 100$

Output Format

Print the averages of all students on separate lines.

The averages must be correct up to 1 decimal place.

Sample Input

```
5 3
89 90 78 93 80
90 91 85 88 86
91 92 83 89 90.5
```

Sample Output

```
90.0
91.0
82.0
90.0
85.5
```

Explanation

Marks obtained by student 1: $89, 90, 91$

Average marks of student 1:

$270/3 = 90$

Marks obtained by student 2: $90, 91, 92$

Average marks of student 2:

$273/3 = 91$

Marks obtained by student 3: $78, 85, 83$

Average marks of student 3:

$246/3 = 82$

Marks obtained by student 4: $93, 88, 89$

Average marks of student 4:

$270/3 = 90$

Marks obtained by student 5: $80, 86, 90.5$

Average marks of student 5:

$256.5/3 = 85.5$

```python
# Enter your code here. Read input from STDIN. Print output to STDOUT
if __name__ == '__main__':
    n,m = map(int, input().split())
    sheet = []
    for _ in range(m):
        sheet.append(map(float, input().split()) )

    for i in zip(*sheet):
        print(sum(i)/len(i) )
```

13.Python Built-Ins Athelete Sort

You are given a spreadsheet that contains a list of N athletes and their details (such as age, height, weight and so on). You are required to sort the data based on the K^{th} attribute and print the final resulting table. Follow the example given below for better understanding.

Rank	Age	Height (in cm)
1	32	190
2	35	175
3	41	188
4	26	195
5	24	176

sort based on k=1 → i.e (age)

Rank	Age	Height (in cm)
5	24	176
4	26	195
1	32	190
2	35	175
3	41	188

Note that K is indexed from 0 to $M-1$, where M is the number of attributes.

Note: If two attributes are the same for different rows, for example, if two atheletes are of the same age, print the row that appeared first in the input.

Input Format

The first line contains N and M separated by a space.
The next N lines each contain M elements.
The last line contains K.

Constraints

$1 \leq N, M \leq 1000$
$0 \leq K < M$
Each element ≤ 1000

Output Format

Print the N lines of the sorted table. Each line should contain the space separated elements. Check the sample below for clarity.

Sample Input 0

```
5 3
10 2 5
7 1 0
9 9 9
1 23 12
```

```
6 5 9
1
```

Sample Output 0

```
7 1 0
10 2 5
6 5 9
9 9 9
1 23 12
```

Explanation 0

The details are sorted based on the second attribute, since K is zero-indexed.

Python lamda()函数

1、lambda是什么?

看个例子:

```
1 g = lambda x:x+1
```

看一下执行的结果:

g(1)

>>>2

g(2)

>>>3

当然，你也可以这样使用:

lambda x:x+1(1)

>>>2

可以这样认为,lambda作为一个表达式，定义了一个匿名函数，上例的代码x为入口参数，x+1为函数体，用函数来表示为:

```
1 def g(x):
2     return x+1
```

非常容易理解，在这里lambda简化了函数定义的书写形式。是代码更为简洁，但是使用函数的定义方式更为

直观，易理解。

Python中，也有几个定义好的全局函数方便使用的，filter, map, reduce

```
>>> foo = [2, 18, 9, 22, 17, 24, 8, 12, 27]
>>>
>>> print filter(lambda x: x % 3 == 0, foo)
[18, 9, 24, 12, 27]
>>>
>>> print map(lambda x: x * 2 + 10, foo)
[14, 46, 28, 54, 44, 58, 26, 34, 64]
>>>
>>> print reduce(lambda x, y: x + y, foo)
139
```

上面例子中的map的作用，非常简单清晰。但是，Python是否非要使用lambda才能做到这样的简洁程度呢？在对象遍历处理方面，其实Python的for..in..if语法已经很强大，并且在易读上胜过了lambda。

比如上面map的例子，可以写成：

print [x * 2 + 10 for x in foo]

非常的简洁，易懂。

filter的例子可以写成：

print [x for x in foo if x % 3 == 0]

同样也是比lambda的方式更容易理解。

上面简要介绍了什么是lambda,下面介绍为什么使用lambda,看一个例子（来自apihelper.py)：

```
processFunc = collapse and (lambda s: " ".join(s.split())) or (lambda s: s
)
```

在Visual Basic，你很有可能要创建一个函数，接受一个字符串参数和一个 *collapse* 参数，并使用 if 语句确定是否压缩空白，然后再返回相应的值。这种方式是低效的，因为函数可能需要处理每一种可能的情况。每次你调用它，它将不得不在给出你所想要的东西之前，判断是否要压缩空白。在 Python 中，你可以将决策逻辑拿到函数外面，而定义一个裁减过的 lambda 函数提供确切的 (唯一的) 你想要的。这种方式更为高效、更为优雅，而且很少引起那些令人讨厌 (哦，想到那些参数就头昏) 的错误。

通过此例子，我们发现，lambda的使用大量简化了代码，使代码简练清晰。但是值得注意的是，这会在一定程度上降低代码的可读性。如果不是非常熟悉python的人或许会对此感到不可理解。

---

lambda 定义了一个匿名函数

lambda 并不会带来程序运行效率的提高，只会使代码更简洁。

如果可以使用for...in...if来完成的，坚决不用lambda。

如果使用lambda，lambda内不要包含循环，如果有，我宁愿定义函数来完成，使代码获得可重用性和更好的可读性。

总结：lambda 是为了减少单行函数的定义而存在的。

```python
#!/bin/python
# Very excellent solution! Try more questions using lamda()
import math
import os
import random
import re
import sys

if __name__ == '__main__':
    N, M = map(int, input().split())
    rows = [input() for _ in range(N)]
    K = int(input())

    for row in sorted(rows, key=lambda row: int(row.split()[K])):
        print(row)
```

14.Python Built-Ins Any or All

---

any()

This expression returns `True` if **any** element of the iterable is true.

If the iterable is empty, it will return `False`.

Code

```
>>> any([1>0,1==0,1<0])
True
>>> any([1<0,2<1,3<2])
False
```

---

all()

This expression returns `True` if **all** of the elements of the iterable are true. If the iterable is empty, it will return `True`.

Code

```
>>> all(['a'<'b','b'<'c'])
True
>>> all(['a'<'b','c'<'b'])
False
```

Task

You are given a space separated list of integers. If all the integers are positive, then you need to check if any integer is a palindromic integer.

Input Format

The first line contains an integer N. N is the total number of integers in the list.

The second line contains the space separated list of N integers.

Constraints

$0 < N < 100$

Output Format

Print `True` if all the conditions of the problem statement are satisfied. Otherwise, print `False`.

Sample Input

```
5
12961514
```

Sample Output

True

Explanation

Condition 1: All the integers in the list are positive.

Condition 2: 5 is a palindromic integer.

Hence, the output is `True`.

Can you solve this challenge in `3 lines of code or less`?

There is `no penalty` for solutions that are correct but have more than 3 lines.

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
if __name__ == '__main__':
    #i = int(input())
    #lis = map(int, input().split())
    #print ( all( v > 0 for v in lis) and any( str(v) == str(v)[::-1] for v in lis))
    n = int(input())
    arr = input().split(" ")
    print(all(int(i)>=0 for i in arr) and any(i == i[::-1]for i in arr))

```

15. Python itertools.product()

Python的内建模块 itertools 提供了非常有用的用于操作迭代对象的函数。

首先，我们看看`itertools`提供的几个“无限”迭代器：

```
>>> import itertools
>>> natuals = itertools.count(1)
>>> for n in natuals:
...     print n
...
1
2
3
...
```

因为`count()`会创建一个无限的迭代器，所以上述代码会打印出自然数序列，根本停不下来，只能按`Ctrl+C`退出。

`cycle()`会把传入的一个序列无限重复下去：

```
>>> import itertools
>>> cs = itertools.cycle('ABC') # 注意字符串也是序列的一种
>>> for c in cs:
...     print c
...
'A'
'B'
'C'
'A'
'B'
'C'
...
```

同样停不下来。

`repeat()`负责把一个元素无限重复下去，不过如果提供第二个参数就可以限定重复次数：

```
>>> ns = itertools.repeat('A', 10)
>>> for n in ns:
...     print n
...
打印10次'A'
```

无限序列只有在`for`迭代时才会无限地迭代下去，如果只是创建了一个迭代对象，它不会事先把无限个元素生成出来，事实上也不可能在内存中创建无限多个元素。

无限序列虽然可以无限迭代下去，但是通常我们会通过`takewhile()`等函数根据条件判断来截取出一个有限的序列：

```
>>> natuals = itertools.count(1)
>>> ns = itertools.takewhile(lambda x: x <= 10, natuals)
>>> for n in ns:
...     print n
...
打印出1到10
```

`itertools`提供的几个迭代器操作函数更加有用：

chain()

`chain()`可以把一组迭代对象串联起来，形成一个更大的迭代器：

```
for c in itertools.chain('ABC', 'XYZ'):
    print c
# 迭代效果：'A' 'B' 'C' 'X' 'Y' 'Z'
```

groupby()

`groupby()`把迭代器中相邻的重复元素挑出来放在一起：

```
>>> for key, group in itertools.groupby('AAABBBCCAAA'):
...     print key, list(group) # 为什么这里要用list()函数呢?
...
A ['A', 'A', 'A']
B ['B', 'B', 'B']
C ['C', 'C']
A ['A', 'A', 'A']
```

实际上挑选规则是通过函数完成的，只要作用于函数的两个元素返回的值相等，这两个元素就被认为是在一组的，而函数返回值作为组的key。如果我们要忽略大小写分组，就可以让元素`'A'`和`'a'`都返回相同的key：

```
>>> for key, group in itertools.groupby('AaaBBbcCAAa', lambda c: c.upper()):
...     print key, list(group)
...
A ['A', 'a', 'a']
B ['B', 'B', 'b']
C ['c', 'C']
A ['A', 'A', 'a']
```

imap()

`imap()`和`map()`的区别在于，`imap()`可以作用于无穷序列，并且，如果两个序列的长度不一致，以短的那个为准。

```
>>> for x in itertools.imap(lambda x, y: x * y, [10, 20, 30], itertools.count(1)):
...     print x
...
10
40
90
```

注意`imap()`返回一个迭代对象，而`map()`返回list。当你调用`map()`时，已经计算完毕：

```
>>> r = map(lambda x: x*x, [1, 2, 3])
>>> r # r已经计算出来了
[1, 4, 9]
```

当你调用`imap()`时，并没有进行任何计算：

```
>>> r = itertools.imap(lambda x: x*x, [1, 2, 3])
>>> r
<itertools.imap object at 0x103d3ff90>
# r只是一个迭代对象
```

必须用`for`循环对`r`进行迭代，才会在每次循环过程中计算出下一个元素：

```
>>> for x in r:
...     print x
...
1
4
9
```

这说明`imap()`实现了“惰性计算”，也就是在需要获得结果的时候才计算。类似`imap()`这样能够实现惰性计算的函数就可以处理无限序列：

```
>>> r = itertools.imap(lambda x: x*x, itertools.count(1))
>>> for n in itertools.takewhile(lambda x: x<100, r):
...     print n
...
结果是什么?
```

如果把`imap()`换成`map()`去处理无限序列会有什么结果?

```
>>> r = map(lambda x: x*x, itertools.count(1))
结果是什么?
```

ifilter()

不用多说了，`ifilter()`就是`filter()`的惰性实现。

小结

`itertools`模块提供的全部是处理迭代功能的函数，它们的返回值不是list，而是迭代对象，只有用`for`循环迭代的时候才真正计算。

itertools.product()

This tool computes the cartesian product of input iterables.

It is equivalent to nested for-loops.

For example, product(A, B) returns the same as ((x,y) for x in A for y in B).

Sample Code

```
>>> from itertools import product
>>>
>>> print list(product([1,2,3],repeat = 2))
[(1, 1), (1, 2), (1, 3), (2, 1), (2, 2), (2, 3), (3, 1), (3, 2), (3, 3)]
>>>
>>> print list(product([1,2,3],[3,4]))
[(1, 3), (1, 4), (2, 3), (2, 4), (3, 3), (3, 4)]
>>>
>>> A = [[1,2,3],[3,4,5]]
>>> print list(product(*A))
[(1, 3), (1, 4), (1, 5), (2, 3), (2, 4), (2, 5), (3, 3), (3, 4), (3, 5)]
>>>
>>> B = [[1,2,3],[3,4,5],[7,8]]
>>> print list(product(*B))
[(1, 3, 7), (1, 3, 8), (1, 4, 7), (1, 4, 8), (1, 5, 7), (1, 5, 8), (2, 3, 7), (2, 3, 8), (2, 4, 7), (2, 4, 8), (2, 5, 7), (2, 5, 8), (3,
3, 7), (3, 3, 8), (3, 4, 7), (3, 4, 8), (3, 5, 7), (3, 5, 8)]
```

Task

You are given a two lists A and B. Your task is to compute their cartesian product AXB.

Example

```
A = [1, 2]
B = [3, 4]

AxB = [(1, 3), (1, 4), (2, 3), (2, 4)]
```

Note: A and B are sorted lists, and the cartesian product's tuples should be output in sorted order.

Input Format

The first line contains the space separated elements of list A.

The second line contains the space separated elements of list B.

Both lists have no duplicate integer elements.

Constraints

$0 < A < 30$

$0 < B < 30$

Output Format

Output the space separated tuples of the cartesian product.

Sample Input

```
12
34
```

Sample Output

```
(1,3)(1,4)(2,3)(2,4)
```

```python
import itertools
if __name__ =='__main__':
    A = list(map(int,input().split()))
    B = list(map(int,input().split()))
    #print(A*B) # can't multiply sequence by non-int of type 'list'
    print(*itertools.product(A, B)) #(1, 3) (1, 4) (2, 3) (2, 4)

```

16. Python itertools.permutations

---

itertools.permutations(iterable[, r])

就是一共有多少种排列组合，都显示出来

This tool returns successive r length permutations of elements in an iterable.

If r is not specified or is None, then r defaults to the length of the iterable, and all possible full length permutations are generated.

Permutations are printed in a lexicographic sorted order. So, if the input iterable is sorted, the permutation tuples will be produced in a sorted order.

Sample Code

```
>>> from itertools import permutations
>>> print permutations(['1','2','3'])
<itertools.permutations object at 0x02A45210>
>>>
>>> print list(permutations(['1','2','3']))
[('1', '2', '3'), ('1', '3', '2'), ('2', '1', '3'), ('2', '3', '1'), ('3', '1', '2'), ('3', '2', '1')]
>>>
>>> print list(permutations(['1','2','3'],2))
[('1', '2'), ('1', '3'), ('2', '1'), ('2', '3'), ('3', '1'), ('3', '2')]
>>>
>>> print list(permutations('abc',3))
[('a', 'b', 'c'), ('a', 'c', 'b'), ('b', 'a', 'c'), ('b', 'c', 'a'), ('c', 'a', 'b'), ('c', 'b', 'a')]
```

Task

You are given a string S.

Your task is to print all possible permutations of size k of the string in lexicographic sorted order.

Input Format

A single line containing the space separated string S and the integer value k.

Constraints

$0 < k \leq len(S)$

The string contains only UPPERCASE characters.

Output Format

Print the permutations of the string S on separate lines.

Sample Input

```
HACK 2
```

Sample Output

```
AC
AH
AK
CA
CH
CK
HA
HC
HK
KA
KC
KH
```

Explanation

All possible size 2 permutations of the string "HACK" are printed in lexicographic sorted order.

```python
# Enter your code here. Read input from STDIN. Print output to STDOUT
from itertools import permutations
if __name__ == '__main__':
    #name,number = input(),int(input())
    #print (list(itertools.permutations(name,number)))
    s,n = input().split()
    print(*[''.join(i) for i in permutations(sorted(s),int(n))],sep='\n')
    # 这个表示这两个字母之间没有间隔
    #使用join默认数字和数字之间是有间隔的， 这样就没有间隔了

```

```python
# 错误解法
print(*['\n'.join(i) for i in permutations(sorted(s),int(n))])

# 输出的错误结果
'''
A
C A
H A
K C
A C
H C
K H
A H
C H
```

```
K K
A K
C K
H
'''
```

17. Python itertools.combinations

itertools.combinations(iterable, r)

排列组合，是组合不是排列

This tool returns the r length subsequences of elements from the input iterable.

Combinations are emitted in lexicographic sorted order. So, if the input iterable is sorted, the combination tuples will be produced in sorted order.

Sample Code

```
>>> from itertools import combinations
>>>
>>> print list(combinations('12345',2))
[('1', '2'), ('1', '3'), ('1', '4'), ('1', '5'), ('2', '3'), ('2', '4'), ('2', '5'), ('3', '4'), ('3', '5'), ('4', '5')]
>>>
>>> A = [1,1,3,3,3]
>>> print list(combinations(A,4))
[(1, 1, 3, 3), (1, 1, 3, 3), (1, 1, 3, 3), (1, 3, 3, 3), (1, 3, 3, 3)]
```

Task

You are given a string S.

Your task is to print all possible combinations, up to size k, of the string in lexicographic sorted order.

Input Format

A single line containing the string S and integer value k separated by a space.

Constraints

$0 < k \leq len(S)$

The string contains only UPPERCASE characters.

Output Format

Print the different combinations of string S on separate lines.

Sample Input

```
HACK 2
```

Sample Output

```
A
C
H
K
AC
AH
AK
CH
CK
HK
```

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
from itertools import combinations
if __name__ == '__main__':
    s,k = input().split()
    #for i in range()
    for i in range(1, int(k)+1):
        for j in combinations(sorted(s), i):
            print (''.join(j),sep = '\n')

```

18. Python itertools.groupby()

—

In this task, we would like for you to appreciate the usefulness of the groupby() function of itertools . To read more about this function, Check this out .

就是数每个数字出现了几次，然后用这种形式表达出来

You are given a string S. Suppose a character 'c' occurs consecutively X times in the string. Replace these consecutive occurrences of the character 'c' with (X, c) in the string.

For a better understanding of the problem, check the explanation.

Input Format

A single line of input consisting of the string S.

Output Format

A single line of output consisting of the modified string.

Constraints

All the characters of S denote integers between 0 and 9.

$1 \leq | S | \leq 10^4$

Sample Input

```
1222311
```

Sample Output

```
(1, 1) (3, 2) (1, 3) (2, 1)
```

Explanation

First, the character 1 occurs only once. It is replaced by $(1, 1)$. Then the character 2 occurs three times, and it is replaced by $(3, 2)$ and so on.

Also, note the single space within each compression and between the compressions.

Python 中的分组函数 groupby( )

groupby()

`groupby()`把迭代器中相邻的重复元素挑出来放在一起：

```
>>> for key, group in itertools.groupby('AAABBBCCAAA'):
...     print key, list(group) # 为什么这里要用list()函数呢?
...
A ['A', 'A', 'A']
B ['B', 'B', 'B']
C ['C', 'C']
A ['A', 'A', 'A']
```

实际上挑选规则是通过函数完成的，只要作用于函数的两个元素返回的值相等，这两个元素就被认为是在一组的，而函数返回值作为组的key。如果我们要忽略大小写分组，就可以让元素`'A'`和`'a'`都返回相同的key：

```
>>> for key, group in itertools.groupby('AaaBBbcCAAa', lambda c: c.upper()):
...     print key, list(group)
...
A ['A', 'a', 'a']
B ['B', 'B', 'b']
C ['c', 'C']
A ['A', 'A', 'a']
```

print(**objects*, *sep=' '*, *end='\n'*, *file=sys.stdout*, *flush=False*)
print函数默认是 \n ， 注意使用。这道题是每个结果的最后用空格分割。

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
from itertools import groupby
if __name__ == '__main__':
    #nums = list(input())
    for i,j in groupby(input()):
        print((len(list(j)),int(i)),end=" ")
'''
Sample Input
1222311
Sample Output
(1, 1) (3, 2) (1, 3) (2, 1)
'''
```

19. Python Iterables and Iterators

The itertools module standardizes a core set of fast, memory efficient tools that are useful by themselves or in combination. Together, they form an iterator algebra making it possible to construct specialized tools succinctly and efficiently in pure Python.

To read more about the functions in this module, check out their documentation here.

You are given a list of N lowercase English letters. For a given integer K, you can select any K indices (assume 1-based indexing) with a uniform probability from the list.

Find the probability that at least one of the K indices selected will contain the letter: 'a'.

Input Format

The input consists of three lines. The first line contains the integer N, denoting the length of the list. The next line consists of N space-separated lowercase English letters, denoting the elements of the list.

The third and the last line of input contains the integer K, denoting the number of indices to be selected.

Output Format

Output a single line consisting of the probability that at least one of the K indices selected contains the letter:'a'.

Note: The answer must be correct up to 3 decimal places.

Constraints

$1 \leq N \leq 10$

$1 \leq K \leq N$

All the letters in the list are lowercase English letters.

Sample Input

```
4
aacd
2
```

Sample Output

```
0.8333
```

Explanation：

All possible unordered tuples of length 2 comprising of indices from 1 to 4 are:

$(1,2),(1,3),(1,4),(2,3),(2,4),(3,4)$

Out of these 6 combinations, 5 of them contain either index 1 or index 2 which are the indices that contain the letter 'a'.

Hence, the answer is $\frac{5}{6}$.

把 a a c d 用 1 2 3 4 指代去思考

```python
# Enter your code here. Read input from STDIN. Print output to STDOUT
from itertools import combinations
if __name__ == '__main__':
```

```
    m = int(input())
    s = list(input().split())
    n = int(input())
    t = list(combinations(s,n))
    f = [i for i in t if 'a' in i]
    print(len(f)/len(t))
    # len() is mainly used in 字 符 串 或 者 数 组

```

Python 对len ()的使用

以下实例展示了 len() 的使用方法：

```
>>>str = "runoob"
>>> len(str)             # 字符串长度
6
>>> l = [1,2,3,4,5]
>>> len(l)               # 列表元素个数
5
```

20. Python Collections.Counter()

collections.Counter()

A counter is a container that stores elements as dictionary keys, and their counts are stored as dictionary values.

这是Python中的字典类型的计数方式

Sample Code

```
>>> from collections import Counter
>>>
>>> myList = [1,1,2,3,4,5,3,2,3,4,2,1,2,3]
>>> print Counter(myList)
Counter({2: 4, 3: 4, 1: 3, 4: 2, 5: 1})
>>>
```

>>> print Counter(myList).items()

```
[(1, 3), (2, 4), (3, 4), (4, 2), (5, 1)]
>>>
```

>>> print Counter(myList).keys()

```
[1, 2, 3, 4, 5]
>>>
```

>>> print Counter(myList).values()

```
[3, 4, 4, 2, 1]
```

Task

$Raghu$ is a shoe shop owner. His shop has X number of shoes.

He has a list containing the size of each shoe he has in his shop.

There are N number of customers who are willing to pay x_i amount of money only if they get the shoe of their desired size.

Your task is to compute how much money $Raghu$ earned.

Input Format

The first line contains X, the number of shoes.

The second line contains the space separated list of all the shoe sizes in the shop.

The third line contains N, the number of customers.

The next N lines contain the space separated values of the $shoe\ size$ desired by the customer and x_i, the price of the shoe.

Constraints

$0 < X < 10^3$

$0 < N \leq 10^3$

$20 < x_i < 100$

$2 < shoe\ size < 20$

Output Format

Print the amount of money earned by $Raghu$.

Sample Input

```
10
2 3 4 5 6 8 7 6 5 18
6
6 55
6 45
6 55
4 40
18 60
10 50
```

Sample Output

```
200
```

Explanation

Customer 1: Purchased size 6 shoe for $55.

Customer 2: Purchased size 6 shoe for $45.

Customer 3: Size 6 no longer available, so no purchase.

Customer 4: Purchased size 4 shoe for $40.

Customer 5: Purchased size 18 shoe for $60.

Customer 6: Size 10 not available, so no purchase.

Total money earned = $55 + 45 + 40 + 60 = 200$

```python
# Enter your code here. Read input from STDIN. Print output to STDOUT
import collections

numShoes = int(input())
shoes = collections.Counter(map(int, input().split()))
numCust = int(input())

income = 0

for i in range(numCust):
    size, price = map(int, input().split())
    if shoes[size]:
        income += price
        shoes[size] -= 1

print(income)
```

21. Python Collections.namedtuple()

---

namedtuple

我们知道`tuple`可以表示不变集合，例如，一个点的二维坐标就可以表示成：

```
>>> p = (1, 2)
```

但是，看到`(1, 2)`，很难看出这个`tuple`是用来表示一个坐标的。

定义一个class又小题大做了，这时，`namedtuple`就派上了用场：

```
>>> from collections import namedtuple
>>> Point = namedtuple('Point', ['x', 'y'])
>>> p = Point(1, 2)
>>> p.x
1
>>> p.y
2
```

`namedtuple`是一个函数，它用来创建一个自定义的`tuple`对象，并且规定了`tuple`元素的个数，并可以用属性而不是索引来引用`tuple`的某个元素。

这样一来，我们用`namedtuple`可以很方便地定义一种数据类型，它具备tuple的不变性，又可以根据属性来引用，使用十分方便。
可以验证创建的`Point`对象是`tuple`的一种子类：

```
>>> isinstance(p, Point)
True
>>> isinstance(p, tuple)
True
```

类似的，如果要用坐标和半径表示一个圆，也可以用`namedtuple`定义：

```
# namedtuple('名称', [属性list]):
Circle = namedtuple('Circle', ['x', 'y', 'r'])
```

collections.namedtuple()

Basically, namedtuples are easy to create, lightweight object types.

They turn tuples into convenient containers for simple tasks.

With namedtuples, you don't have to use integer indices for accessing members of a tuple.

Example

Code 01

```
>>> from collections import namedtuple
>>> Point = namedtuple('Point','x,y')
>>> pt1 = Point(1,2)
>>> pt2 = Point(3,4)
>>> dot_product = (pt1.x * pt2.x) + (pt1.y * pt2.y)
```

```
>>> print dot_product
11
```

Code 02

```
>>> from collections import namedtuple
>>> Car = namedtuple('Car','Price Mileage Colour Class')
>>> xyz = Car(Price = 100000, Mileage = 30, Colour = 'Cyan', Class = 'Y')
>>> print xyz
Car(Price=100000, Mileage=30, Colour='Cyan', Class='Y')
>>> print xyz.Class
Y
```

Task

Dr. John Wesley has a spreadsheet containing a list of student's IDs, $marks$, $class$ and $name$.

Your task is to help Dr. Wesley calculate the average marks of the students.

$$Average = \frac{Sum\ of\ all\ marks}{Total\ Students}$$

Note:

1. Columns can be in any order. IDs, marks, class and name can be written in any order in the spreadsheet.

2. Column names are ID, MARKS, CLASS and NAME. (The spelling and case type of these names won't change.)

Input Format

The first line contains an integer N, the total number of students.

The second line contains the names of the columns in any order.

The next N lines contains the $marks$, IDs, $name$ and $class$, under their respective column names.

Constraints

$0 < N \leq 100$

Output Format

Print the average marks of the list corrected to 2 decimal places.

Sample Input

TESTCASE 01

```
5
ID   MARKS   NAME   CLASS
1    97    Raymond  7
2    50    Steven  4
3    91    Adrian  9
4    72    Stewart  5
5    80    Peter  6
```

TESTCASE 02

```
5
MARKS   CLASS   NAME   ID
```

```
92      2       Calum   1
82      5       Scott   2
94      2       Jason   3
55      8       Glenn   4
82      2       Fergus  5
```

Sample Output

TESTCASE 01

```
78.00
```

TESTCASE 02

```
81.00
```

Explanation

TESTCASE 01

Average = $(97 + 50 + 91 + 72 + 80)/5$

Can you solve this challenge in 4 lines of code or less?

NOTE: There is no penalty for solutions that are correct but have more than 4 lines.

classmethod `somenamedtuple.`**`_make`**`(`*iterable*`)`

Class method that makes a new instance from an existing sequence or iterable.

```
>>> t = [11, 22]
>>> Point._make(t)
Point(x=11, y=22)
```

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
# Code Implementation
from collections import namedtuple
if __name__ == '__main__':
    n, categories = int(input()), input().split()
    Grade = namedtuple('Grade', categories)
    marks = [int(Grade._make(input().split()).MARKS) for _ in range(n)]
    #use _make to make the tuple
    print((sum(marks) / len(marks)))

```

22. Python Collections.OrderedDict()

collections.OrderedDict

An OrderedDict is a dictionary that remembers the order of the keys that were inserted first. If a new entry overwrites an existing entry, the original insertion position is left unchanged.

Python中的字典对象可以以“键：值”的方式存取数据。OrderedDict是它的一个子类，实现了对字典对象中元素的排序。比如下面比较了两种方式的不同：

Example

Code

```
>>> from collections import OrderedDict
>>>
>>> ordinary_dictionary = {}
>>> ordinary_dictionary['a'] = 1
>>> ordinary_dictionary['b'] = 2
>>> ordinary_dictionary['c'] = 3
>>> ordinary_dictionary['d'] = 4
>>> ordinary_dictionary['e'] = 5
>>>
>>> print ordinary_dictionary
{'a': 1, 'c': 3, 'b': 2, 'e': 5, 'd': 4}
>>>
>>> ordered_dictionary = OrderedDict()
>>> ordered_dictionary['a'] = 1
>>> ordered_dictionary['b'] = 2
>>> ordered_dictionary['c'] = 3
>>> ordered_dictionary['d'] = 4
>>> ordered_dictionary['e'] = 5
>>>
>>> print ordered_dictionary
OrderedDict([('a', 1), ('b', 2), ('c', 3), ('d', 4), ('e', 5)])
```

Task

You are the manager of a supermarket.

You have a list of N items together with their prices that consumers bought on a particular day.

Your task is to print each `item_name` and `net_price` in order of its first occurrence.

item_name = Name of the item.

net_price = Quantity of the item sold multiplied by the price of each item.

Input Format

The first line contains the number of items, N.

The next N lines contains the item's name and price, separated by a space.

Constraints

$0 < N \leq 100$

Output Format

Print the item_name and net_price in order of its first occurrence.

Sample Input

```
9
BANANA FRIES 12
POTATO CHIPS 30
APPLE JUICE 10
CANDY 5
APPLE JUICE 10
CANDY 5
CANDY 5
CANDY 5
POTATO CHIPS 30
```

Sample Output

```
BANANA FRIES 12
POTATO CHIPS 60
APPLE JUICE 20
CANDY 20
```

Explanation

BANANA FRIES: Quantity bought: 1, Price: 12

Net Price: 12

POTATO CHIPS: Quantity bought: 2, Price: 30

Net Price: 60

APPLE JUICE: Quantity bought: 2, Price: 10

Net Price: 20

CANDY: Quantity bought: 4, Price: 5

Net Price: 20

```python
# Enter your code here. Read input from STDIN. Print output to STDOUT
from collections import OrderedDict

number_ = int(input())
odict = OrderedDict()
for i in range(number_):
    litem = input().split(' ')
    price = int(litem[-1])
    # when mentions the last element, use "-1"
    item_name = " ".join(litem[:-1])
    # before the last element, use " " to join the elements
    if odict.get(item_name):
        odict[item_name] += price
    else:
        odict[item_name] = price

for i,v in odict.items():
    print(i,v)

```

23. Python Collections Word Order

Using `list` as the `default_factory`, it is easy to group a sequence of key-value pairs into a dictionary of lists:

```
>>> s = [('yellow', 1), ('blue', 2), ('yellow', 3), ('blue', 4), ('red',
 1)]
>>> d = defaultdict(list)
>>> for k, v in s:
...     d[k].append(v)
...
>>> sorted(d.items())
[('blue', [2, 4]), ('red', [1]), ('yellow', [1, 3])]
```

You are given n words. Some words may repeat. For each word, output its number of occurrences. The output order should correspond with the input order of appearance of the word. See the sample input/output for clarification.

Note: Each input line ends with a **"\n"** character.

Constraints:

$1 \leq n \leq 10^5$

The sum of the lengths of all the words do not exceed 10^6

All the words are composed of lowercase English letters only.

Input Format

The first line contains the integer, n.

The next n lines each contain a word.

Output Format

Output 2 lines.

On the first line, output the number of distinct words from the input.

On the second line, output the number of occurrences for each distinct word according to their appearance in the input.

Sample Input

```
4
bcdef
abcdefg
bcde
bcdef
```

Sample Output

```
3
2 1 1
```

Explanation

There are 3 distinct words. Here, **"bcdef"** appears twice in the input at the first and last positions. The other words appear once each. The order of the first appearances are **"bcdef"**, **"abcdefg"** and **"bcde"** which corresponds to the output.

解释：一共有3个不同的字符串；然后分别每个字符串出现的次数为2，1，1

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
from collections import defaultdict
if __name__ == '__main__':
    n = int(input())
    #for i in range(n):
    if 1<=n<=10**5:
        d=defaultdict(int)
        for i in range(n):
            key=input()
            d[key] +=1
        print(len(d.keys()))
# the common usage 1)keys 2)values  3)items
        x=(d.values())
        print(*x)
```

Recall:

```
>>> from collections import Counter
>>>
>>> myList = [1,1,2,3,4,5,3,2,3,4,2,1,2,3]
>>> print Counter(myList)
Counter({2: 4, 3: 4, 1: 3, 4: 2, 5: 1})
>>>
```

`>>> print Counter(myList).items()`

```
[(1, 3), (2, 4), (3, 4), (4, 2), (5, 1)]
>>>
```

```
>>> print Counter(myList).keys()
```

```
[1, 2, 3, 4, 5]
>>>
```

```
>>> print Counter(myList).values()
```

```
[3, 4, 4, 2, 1]
```

24. Python Collections.deque()

collections.deque()

A deque is a double-ended queue. It can be used to add or remove elements from both ends.

Deques support thread safe, memory efficient appends and pops from either side of the deque with approximately the same $O(1)$ performance in either direction.

Click on the link to learn more about **deque() methods**.

Click on the link to learn more about various approaches to working with deques: **Deque Recipes**.

Example

Code

```
>>> from collections import deque
>>> d = deque()
>>> d.append(1)
>>> print d
deque([1])
>>> d.appendleft(2)
>>> print d
deque([2, 1])
>>> d.clear()
>>> print d
deque([])
>>> d.extend('1')
>>> print d
deque(['1'])
>>> d.extendleft('234')
>>> print d
deque(['4', '3', '2', '1'])
>>> d.count('1')
1
>>> d.pop()
'1'
>>> print d
deque(['4', '3', '2'])
>>> d.popleft()
'4'
>>> print d
deque(['3', '2'])
>>> d.extend('7896')
```

```
>>> print d
deque(['3', '2', '7', '8', '9', '6'])
>>> d.remove('2')
>>> print d
deque(['3', '7', '8', '9', '6'])
>>> d.reverse()
>>> print d
deque(['6', '9', '8', '7', '3'])
>>> d.rotate(3)
>>> print d
deque(['8', '7', '3', '6', '9'])
```

Task

Perform append, pop, popleft and appendleft methods on an empty deque d.

Input Format

The first line contains an integer N, the number of operations.

The next N lines contains the space separated names of methods and their values.

Constraints

$0 < N \leq 100$

Output Format

Print the space separated elements of deque d.

Sample Input

```
6
append 1
append 2
append 3
appendleft 4
pop
popleft
```

Sample Output

```
1 2
```

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
from collections import deque
if __name__ == '__main__':
```

```
4
5       d = deque()
6       for i in range(int(input())):
7           s = input().split()
8           if s[0] == 'append':
9               d.append(s[1])
10          elif s[0] == 'appendleft':
11              d.appendleft(s[1])
12          elif s[0] == 'pop':
13              d.pop()
14          else:
15              d.popleft()
16      print (" ".join(d) )
```

25. Python Collections Company Logo

A newly opened multinational brand has decided to base their company logo on the three most common characters in the company name. They are now trying out various combinations of company names and logos based on this condition. Given a string s, which is the company name in lowercase letters, your task is to find the top three most common characters in the string.

- Print the three most common characters along with their occurrence count.
- Sort in descending order of occurrence count.
- If the occurrence count is the same, sort the characters in alphabetical order.

For example, according to the conditions described above,

GOOGLE would have it's logo with the letters **G, O, E**.

Input Format

A single line of input containing the string S.

Constraints

- $3 < len(S) \leq 10^4$

Output Format

Print the three most common characters along with their occurrence count each on a separate line.
Sort output in descending order of occurrence count.
If the occurrence count is the same, sort the characters in alphabetical order.

Sample Input 0

```
aabbbccde
```

Sample Output 0

```
b 3
a 2
c 2
```

Explanation 0

aabbbccde

Here, b occurs 3 times. It is printed first.

Both a and c occur 2 times. So, a is printed in the second line and c in the third line because a comes before c in the alphabet.

Note: The string S has at least 3 distinct characters.

`most_common`([*n*])

Return a list of the *n* most common elements and their counts from the most common to the least. If *n* is omitted or `None`, `most_common()` returns *all* elements in the counter. Elements with equal counts are ordered arbitrarily:

```
>>> Counter('abracadabra').most_common(3)  # doctest: +SKIP
[('a', 5), ('r', 2), ('b', 2)]
```

```
#!/bin/python3

import math
import os
import random
import re
import sys
from collections import Counter
from collections import OrderedDict
class OrderedCounter(Counter, OrderedDict):
    pass
[print(*c) for c in OrderedCounter(sorted(input())).most_common(3)]
 # see the most_common updates on the upfront

if __name__ == '__main__':
    s = input()
```

Calendar Module

The calendar module allows you to output calendars and provides additional useful functions for them.

直接查API吧，内容比价琐碎，但是简单！

class calendar.TextCalendar([firstweekday])

This class can be used to generate plain text calendars.

Sample Code

```
>>> import calendar
>>>
>>> print calendar.TextCalendar(firstweekday=6).formatyear(2015)
                              2015

      January               February               March
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
             1  2  3      1  2  3  4  5  6  7      1  2  3  4  5  6  7
 4  5  6  7  8  9 10      8  9 10 11 12 13 14      8  9 10 11 12 13 14
11 12 13 14 15 16 17     15 16 17 18 19 20 21     15 16 17 18 19 20 21
18 19 20 21 22 23 24     22 23 24 25 26 27 28     22 23 24 25 26 27 28
25 26 27 28 29 30 31                              29 30 31

       April                  May                   June
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
          1  2  3  4                  1  2         1  2  3  4  5  6
 5  6  7  8  9 10 11      3  4  5  6  7  8  9      7  8  9 10 11 12 13
12 13 14 15 16 17 18     10 11 12 13 14 15 16     14 15 16 17 18 19 20
19 20 21 22 23 24 25     17 18 19 20 21 22 23     21 22 23 24 25 26 27
26 27 28 29 30           24 25 26 27 28 29 30     28 29 30
                  31

       July                  August              September
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
          1  2  3  4                  1            1  2  3  4  5
 5  6  7  8  9 10 11      2  3  4  5  6  7  8      6  7  8  9 10 11 12
12 13 14 15 16 17 18      9 10 11 12 13 14 15     13 14 15 16 17 18 19
19 20 21 22 23 24 25     16 17 18 19 20 21 22     20 21 22 23 24 25 26
26 27 28 29 30 31        23 24 25 26 27 28 29     27 28 29 30
                  30 31
```

```
      October                 November                December
Su Mo Tu We Th Fr Sa    Su Mo Tu We Th Fr Sa    Su Mo Tu We Th Fr Sa
                1  2     1  2  3  4  5  6  7             1  2  3  4  5
 4  5  6  7  8  9 10     8  9 10 11 12 13 14     6  7  8  9 10 11 12
11 12 13 14 15 16 17    15 16 17 18 19 20 21    13 14 15 16 17 18 19
18 19 20 21 22 23 24    22 23 24 25 26 27 28    20 21 22 23 24 25 26
25 26 27 28 29 30 31    29 30                   27 28 29 30 31
```

To learn more about different calendar functions, click here.

Task

You are given a date. Your task is to find what the day is on that date.

Input Format

A single line of input containing the space separated month, day and year, respectively, in $MM\ DD\ YYYY$ format.

Constraints

- $2000 < year < 3000$

Output Format

Output the correct day in capital letters.

Sample Input

```
08 05 2015
```

Sample Output

```
WEDNESDAY
```

Explanation

The day on August $5^{th}\ 2015$ was `WEDNESDAY`.

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
import calendar
#calendar.Calendar(calendar.SUNDAY)
user_input = input().split()
# the input is divided into arrays...
month = int(user_input[0])
```

```
day = int(user_input[1])
year = int(user_input[2])
c = calendar.weekday(year, month, day)
# Directly use the python library, I don't even know why.
if c == 0:
    print("MONDAY")
elif c == 1:
    print("TUESDAY")
elif c == 2:
    print("WEDNESDAY")
elif c==3:
    print("THURSDAY")
elif c==4:
    print("FRIDAY")
elif c== 5:
    print("SATURDAY")
elif c==6:
    print("SUNDAY")
```

27. Python Functionals Map and Lambda Function

Let's learn some new Python concepts! You have to generate a list of the first N fibonacci numbers, 0 being the first number. Then, apply the map function and a lambda expression to cube each fibonacci number and print the list.

Concept

The `map()` function applies a function to every member of an iterable and returns the result. It takes two parameters: first, the function that is to be applied and secondly, the iterables.
Let's say you are given a list of names, and you have to print a list that contains the length of each name.

```
>> print(list(map(len, ['Tina', 'Raj', 'Tom'])))
[4, 3, 3]
```

Lambda is a single expression anonymous function often used as an inline function. In simple words, it is a function that has only one line in its body. It proves very handy in functional and GUI programming.

```
>> sum = lambda a, b, c: a + b + c
>> sum(1, 2, 3)
6
```

Note:

Lambda functions cannot use the return statement and can only have a single expression. Unlike def, which creates a function and assigns it a name, lambda creates a function and returns the function itself. Lambda can be used inside lists and dictionaries.

Input Format

One line of input: an integer N.

Constraints

$0 \leq N \leq 15$

Output Format

A list on a single line containing the cubes of the first N fibonacci numbers.

Sample Input

```
5
```

Sample Output

```
[0, 1, 1, 8, 27]
```

Explanation

The first 5 fibonacci numbers are $[0, 1, 1, 2, 3]$, and their cubes are $[0, 1, 1, 8, 27]$.

```python
cube = lambda x:x*x*x
# complete the lambda function


def fibonacci(n):
    # return a list of fibonacci numbers
    #fibonacci(1)=0
    #fibonacci(2)=1
    lis = [0,1]
    #if n >=2:
    #    lis[n]=lis[n-1]+lis[n-2]
    for i in range(2,n):
        lis.append(lis[i-1]+lis[i-2])
    return lis[0:n]



if __name__ == '__main__':
    n = int(input())
    print(list(map(cube, fibonacci(n))))
```

28. Python Reduce Function（分数运算）

语法

reduce() 函数语法：

```
reduce(function, iterable[, initializer])
```

参数

function -- 函数，有两个参数

iterable -- 可迭代对象

initializer -- 可选，初始参数

返回值

返回函数计算结果。

实例

以下实例展示了 reduce( ) 的使用方法：

```
>>>def add(x, y) :            # 两数相加
...     return x + y
...
>>> reduce(add, [1,2,3,4,5])   # 计算列表和：1+2+3+4+5
15
>>> reduce(lambda x, y: x+y, [1,2,3,4,5])  # 使用 lambda 匿名函数
15
```

Given a list of rational numbers,find their product.

Concept

The `reduce()` function applies a function of two arguments cumulatively on a list of objects in succession from left to right to reduce it to one value. Say you have a list, say `[1,2,3]` and you have to find its sum.

```
>>> reduce(lambda x, y : x + y,[1,2,3])
6
```

You can also define an initial value. If it is specified, the function will assume initial value as the value given, and then reduce. It is equivalent to adding the initial value at the beginning of the list. For example:

```
>>> reduce(lambda x, y : x + y, [1,2,3], -3)
3

>>> from fractions import gcd
>>> reduce(gcd, [2,4,8], 3)
1
```

Input Format

First line contains n, the number of rational numbers.

The i^{th} of next n lines contain two integers each, the numerator(N_i) and denominator(D_i) of the i^{th} rational number in the list.

Constraints

- $1 \leq n \leq 100$
- $1 \leq N_i, D_i \leq 10^9$

Output Format

Print only one line containing the numerator and denominator of the product of the numbers in the list in its simplest form, i.e. numerator and denominator have no common divisor other than 1.

Sample Input 0

```
3
1 2
3 4
10 6
```

Sample Output 0

```
5 8
```

Explanation 0

$$\frac{1}{2}\frac{3}{4}\frac{10}{6} = \frac{5}{8}$$

Required product is

```python
from fractions import Fraction
from functools import reduce
import operator
def product(fracs):
    #t = # complete this line with a reduce statement
    #fracs1=[]
    #for i in range(len(fracs)):
    t =  reduce(operator.mul , fracs) # complete this line with a reduce statement
    return t.numerator, t.denominator

if __name__ == '__main__':
    fracs = []
    for _ in range(int(input())):
        fracs.append(Fraction(*map(int, input().split())))
    result = product(fracs)
    print(*result)
```

29. Python Classes Dealing with Complex Numbers

For this challenge, you are given two complex numbers, and you have to print the result of their addition, subtraction, multiplication, division and modulus operations.

复数的加减乘除以及模运算

The real and imaginary precision part should be correct up to two decimal places.

Input Format

One line of input: The real and imaginary part of a number separated by a space.

Output Format

For two complex numbers C and D, the output should be in the following sequence on separate lines:

- $C+D$
- $C-D$
- $C*D$
- C/D
- $mod(C)$
- $mod(D)$

For complex numbers with non-zero real(A) and complex part(B), the output should be in the following format:

$A+Bi$

Replace the plus symbol $(+)$ with a minus symbol $(-)$ when $B<0$.

For complex numbers with a zero complex part i.e. real numbers, the output should be:

$A+0.00i$

For complex numbers where the real part is zero and the complex part(B) is non-zero, the output should be:

$0.00+Bi$

Sample Input

```
2 1
5 6
```

Sample Output

```
7.00+7.00i
-3.00-5.00i
4.00+17.00i
0.26-0.11i
2.24+0.00i
7.81+0.00i
```

Concept

Python is a fully object-oriented language like C++, Java, etc. For reading about classes, refer here.

Methods with a double underscore before and after their name are considered as built-in methods. They are used by interpreters and are generally used in the implementation of overloaded operators or other built-in functionality.

```
__add__-> Can be overloaded for + operation
```

```
__sub__ -> Can be overloaded for - operation
```

```
__mul__ -> Can be overloaded for * operation
```

For more information on operator overloading in Python, refer here.

```python
import math

class Complex(object):
    def __init__(self, real, imaginary):
        self.real = real
        self.imaginary = imaginary

    def __add__(self, no):
        real = self.real + no.real
```

```
        imaginary = self.imaginary + no.imaginary
        return Complex(real, imaginary)

    def __sub__(self, no):
        real = self.real - no.real
        imaginary = self.imaginary - no.imaginary
        return Complex(real, imaginary)

    def __mul__(self, no):
        real = self.real * no.real - self.imaginary * no.imaginary
        imaginary = self.real * no.imaginary + self.imaginary * no.real
        return Complex(real, imaginary)

    def __div__(self, no):
        x = float(no.real ** 2 + no.imaginary ** 2)
        y = self * Complex(no.real, -no.imaginary)
        real = y.real / x
        imaginary = y.imaginary / x
        return Complex(real, imaginary)

    def mod(self):
        real = math.sqrt(self.real ** 2 + self.imaginary ** 2)
        return Complex(real, 0)

    def __str__(self):
        if self.imaginary == 0:
            result = "%.2f+0.00i" % (self.real)
        elif self.real == 0:
            if self.imaginary >= 0:
                result = "0.00+%.2fi" % (self.imaginary)
            else:
                result = "0.00-%.2fi" % (abs(self.imaginary))
        elif self.imaginary > 0:
            result = "%.2f+%.2fi" % (self.real, self.imaginary)
        else:
            result = "%.2f-%.2fi" % (self.real, abs(self.imaginary))
        return result


C = map(float, raw_input().split())
D = map(float, raw_input().split())
x = Complex(*C)
y = Complex(*D)
print '\n'.join(map(str, [x+y, x-y, x*y, x/y, x.mod(), y.mod()]))
```

30. Python Classes Find the Torsional Angle

You are given four points A, B, C and D in a 3-dimensional Cartesian coordinate system. You are required to print the angle between the plane made by the points A, B, C and B, C, D in degrees(**not radians**). Let the angle be PHI.

$Cos(PHI) = (X.Y)/|X||Y|$ where $X = AB$ x BC and $Y = BC$ x CD.

Here, $X.Y$ means the dot product of X and Y, and AB x BC means the cross product of vectors AB and BC. Also, $AB = B - A$.

Input Format

One line of input containing the space separated floating number values of the X, Y and Z coordinates of a point.

Output Format

Output the angle correct up to two decimal places.

Sample Input

```
045
176
059
172
```

Sample Output

```
8.19
```

```python
# python2
import math
class Points(object):
    def __init__(self, x, y, z):
        self.x = x
        self.y = y
        self.z = z

    def __sub__(self, no):
        x = self.x - no.x
        y = self.y - no.y
        z = self.z - no.z
        return Points(x, y, z)

    def dot(self, no):
```

```
        x = self.x * no.x
        y = self.y * no.y
        z = self.z * no.z
        return x + y + z

    def cross(self, no):
        x = self.y * no.z - self.z * no.y
        y = self.z * no.x - self.x * no.z
        z = self.x * no.y - self.y * no.x
        return Points(x, y, z)

    def absolute(self):
        return pow((self.x ** 2 + self.y ** 2 + self.z ** 2), 0.5)


points = list()
for i in range(4):
    a = map(float, raw_input().split())
    points.append(a)

A, B, C, D = Points(*points[0]), Points(*points[1]), Points(*points[2]), Points(*points[
3])
X = (B - A).cross(C - B)
Y = (C - B).cross(D - C)
angle = math.acos(X.dot(Y) / (X.absolute() * Y.absolute()))

print "%.2f" % math.degrees(angle)
```