* Print Hello, World! to stdout.

Ans:- print("Hello, World!")

* Given an integer, , perform the following conditional actions:
* If  is odd, print Weird
* If  is even and in the inclusive range of  to , print Not Weird
* If  is even and in the inclusive range of  to , print Weird
* If  is even and greater than , print Not Weird

Ans:- n=int(input())

print("Weird") if n%2!=0 else print("Not Weird") if n>=2 and n<=5 else print("Weird") if n>=6 and n<=20 else print("Not Weird")

* The provided code stub reads two integers from STDIN,  a and b. Add code to print three lines where:

1. The first line contains the sum of the two numbers.
2. The second line contains the difference of the two numbers (first - second).
3. The third line contains the product of the two numbers.

Ans:- a = int(input())

b = int(input())

print(a+b)

print(a-b)

print(a\*b)

* The first line contains the first integer,a .  
  The second line contains the second integer,b

The result of the integer division a//b.

The result of the float division is a/b.

Ans:- a = int(input())

b = int(input())

print(a//b)

print(a/b)

* The provided code stub reads and integer, n, from STDIN. For all non-negative integers i<n , print i2 .

Ans:- n = int(input())

print(\*(i\*\*2 for i in range(n)), sep="\n")

* The included code stub will read an integer,n , from STDIN.Without using any string methods, try to print the following 123…n Print the list of integers from 1 through n as a string, without spaces.

Ans:- n = int(input())

print(\*(range(1, n + 1)), sep="")

* You are given a string and your task is to swap cases. In other words, convert all lowercase letters to uppercase letters and vice versa.

Ans:- swap\_case=lambda s: s.swapcase()

* You are given a string. Split the string on a " " (space) delimiter and join using a - hyphen.

Ans:- split\_and\_join=lambda line: "-".join(line.split(" "))

* Given a year, determine whether it is a leap year. If it is a leap year, return the Boolean True, otherwise return False.Note that the code stub provided reads from STDIN and passes arguments to the is\_leap function.

Ans:- is\_leap=lambda year: (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0)

* A close-up of a number

  Description automatically generated

*int arr:* an array of integers

**Returns**:*float:* the resulting float value rounded to 3 places after the decimal

Ans:- average=lambda array: f"{sum(set(array)) / len(set(array)):.3f}"

* The first line contains n. The second line contains an array A[]  of n integers each separated by a space. Print the runner-up score. Given list is [2,3,6,6,5]. The maximum score is 6, second maximum is 5. Hence, we print 5 as the runner-up score.

Ans:-     n = int(input())

    arr = map(int, input().split())

     print(sorted(set(arr),reverse=True)[1])

* You are given three integers  x,y and z representing the dimensions of a cuboid along with an integer n. Print a list of all possible coordinates given by (i,j,k) on a 3D grid where the sum of i+j+k is not equal to n. Four integers  x,y,z and n, each on a separate line.

Input:- 1

1

1

2

Output:- [[0, 0, 0], [0, 0, 1], [0, 1, 0], [1, 0, 0], [1, 1, 1]]

Ans:-  x = int(input())

     y = int(input())

     z = int(input())

     n = int(input())

     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])

* You are given the firstname and lastname of a person on two different lines. Your task is to read them and print the following:

Hello firstname lastname! You just delved into python.

Ans:- print\_full\_name=lambda first,last: print("Hello ",first," ",last,"! You just delved into python.",sep="")

* Complete the *mutate\_string* function.

*mutate\_string* has the following parameters:

*string string:* the string to change

*int position:* the index to insert the character at

*string character:* the character to insert

**Returns** *string:* the altered string

Ans:- mutate\_string=lambda string, position, character: "".join([character if x==position else y for x,y in enumerate(string)])

* The provided code stub will read in a dictionary containing key/value pairs of name:[marks] for a list of students. Print the average of the marks array for the student name provided, showing 2 places after the decimal.

Input: The first line contains the integer n, the number of students' records. The next n lines contain the names and marks obtained by a student, each value separated by a space. The final line contains **query\_name**, the name of a student to query.

**Output Format**:Print one line: The average of the marks obtained by the particular student correct to 2 decimal places.

Ans:-  n = int(input())

     student\_marks = {}

     for i in range(n):

        name, \*line = input().split()

        scores = list(map(float, line))

        student\_marks[name] = scores

     query\_name = input()

     print(f"{sum(student\_marks[query\_name])/len(student\_marks[query\_name]):.2f}")

* 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. The first line of input contains the original string. The next line contains the substring. Output the integer number indicating the total number of occurrences of the substring in the original string.

Input:-

ABCDCDC

CDC

Output:-

2

Ans:- count\_substring=lambda string, sub\_string: len(set(string.find(sub\_string,i) for i in range(len(string)) if string.find(sub\_string,i)!=-1))

* A single line of input containing the full name, S. Print the capitalized string, S.

**Sample Input:** chris alan

**Sample Output:** Chris Alan

Ans:- solve=lambda s: "".join([y.upper() if (x == " " and y != " ") else y for x, y in zip(" " + s, s)])

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

**Sample Input:** ABCDEFGHIJKLIMNOQRSTUVWXYZ

4

**Sample Output:**

ABCD

EFGH

IJKL

IMNO

QRST

UVWX

YZ

Ans:- wrap=lambda string, max\_width: "\n".join([string[i:i+max\_width] for i in range(0,len(string),max\_width)])

* **Input Format**: The first line contains a, the second line contains b, and the third line contains m.

**Output:** Print two lines.  
On the first line, print the result of pow(a,b). On the second line, print the result of pow(a,b,m),i.e., a^b mod m.

Ans:- a=int(input())

b=int(input())

m=int(input())

print(a\*\*b)

print((a\*\*b)%m)

* **Input Format**: The first line contains the first integer,a , and the second line contains the second integer, b.

The first line is the integer division a//b.  
The second line is the result of the modulo operator: a%b.  
The third line prints the divmod of a and b.

**Sample Input**

177

10

**Sample Output**

17

7

(17, 7)

Ans:- a=int(input())

b=int(input())

print(a//b)

print(a%b)

print((a//b,a%b))

* You are given a positive integer N. Print a numerical triangle of height N-1 like the one below:

1

22

333

4444

55555

......

Ans:- for i in range(1,int(input())):

    print(((10\*\*i-1)//9)\*i)

* Read four numbers,a ,b ,c and d, and print the result of ab+cd.

Ans:- a=int(input())

b=int(input())

c=int(input())

d=int(input())

print(a\*\*b+c\*\*d)

* [**Polar coordinates**](https://en.wikipedia.org/wiki/Polar_coordinate_system) are an alternative way of representing Cartesian coordinates or [Complex Numbers](https://en.wikipedia.org/wiki/Complex_number). A complex number  z=x+yj is completely determined by its real part x and imaginary part y.  
  Here, j is the [imaginary unit](https://en.wikipedia.org/wiki/Imaginary_unit). A polar coordinate (r,w) is completely determined by modulus r and phase angle w.

A blue arrow pointing up

Description automatically generated

If we convert complex number z to its polar coordinate, we find:  
r: Distance from z to origin, i.e., math.sqrt(x^2+y^2)   
w: Counter clockwise angle measured from the positive x-axis to the line segment that joins z to the origin.

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

Output two lines:  
The first line should contain the value of r.  
The second line should contain the value of w.

Ans:- import math

z=complex(input())

print((z.real\*\*2+z.imag\*\*2)\*\*0.5)

print(math.atan2(z.imag,z.real))

* 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.

One line of input: an integer N.

Output: 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]

Ans: cube = lambda x: x\*\*3

fibonacci=lambda n, seq=[0, 1]:[] if n==0 else [seq[0]] if n==1 else seq if len(seq) >= n else fibonacci(n, seq + [seq[-1] + seq[-2]])

     n = int(input())

     print(list(map(cube, fibonacci(n))))

* Given a list of rational numbers,find their product.

**Input Format**: First line contains n, the number of rational numbers.  
The ith of next n lines contain two integers each, the numerator(Ni) and denominator(Di) of the ith rational number in the list.

**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.

Ans:- from fractions import Fraction

from functools import reduce

product=lambda fracs:(reduce(lambda a,b:a\*b,fracs).numerator, reduce(lambda a,b:a\*b,fracs).denominator)

     fracs = []

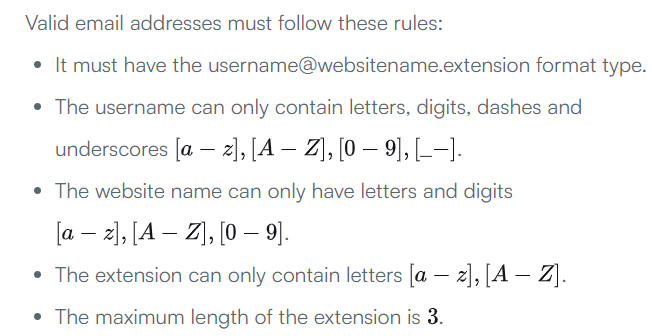
     for i in range(int(input())):

        fracs.append(Fraction(\*map(int, input().split())))

     result = product(fracs)

     print(\*result)

* You are given an integer N followed by N email addresses. Your task is to print a list containing only valid email addresses in lexicographical order.



**Concept**

A filter takes a function returning True or False and applies it to a sequence, returning a list of only those members of the sequence where the function returned True. A Lambda function can be used with filters.

**Input Format**

The first line of input is the integer N, the number of email addresses.  
 N lines follow, each containing a string.

**Sample Input**

3

lara@hackerrank.com

brian-23@hackerrank.com

britts\_54@hackerrank.com

**Sample Output**

['brian-23@hackerrank.com', 'britts\_54@hackerrank.com'

Ans:- def fun(email):

     import re

     pattern = r'^[a-zA-Z0-9\_-]+@[a-zA-Z0-9]+\.[a-zA-Z]{1,3}$'

     return re.match(pattern, email) is not None

def filter\_mail(emails):

     return list(filter(fun, emails))

n = int(input())

    emails = []

    for i in range(n):

        emails.append(input())

filtered\_emails = filter\_mail(emails)

filtered\_emails.sort()

print(filtered\_emails)

* A valid email address meets the following criteria:

It's composed of a username, domain name, and extension assembled in this format: username@domain.extension

The username starts with an English alphabetical character, and any subsequent characters consist of one or more of the following: [alphanumeric characters](https://en.wikipedia.org/wiki/Alphanumeric), -,., and \_.

The domain and extension contain only [English alphabetical characters](https://en.wikipedia.org/wiki/English_alphabet).

The extension is 1,2, or 3 characters in length.

Given  pairs of names and email addresses as input, print each name and email address pair having a valid email address on a new line.

**Input Format**:The first line contains a single integer, n, denoting the number of email address.  
Each line i of the n subsequent lines contains a name and an email address as two space-separated values following this format:

name [user@email.com](mailto:user@email.com)

**Output Format:**Print the space-separated name and email address pairs containing valid email addresses only. Each pair must be printed on a new line in the following format:

name [user@email.com](mailto:user@email.com)

You must print each valid email address in the same order as it was received as input.

**Sample Input**

2

DEXTER <dexter@hotmail.com>

VIRUS <virus!@variable.:p>

**Sample Output**

DEXTER [dexter@hotmail.com](mailto:dexter@hotmail.com)

Ans:- n=int(input())

name=[]

email=[]

for i in range(n):

     temp\_name, temp\_email = input().split()

     name.append(temp\_name)

     email.append(temp\_email)

def fun(email1):

     import re

     pattern = r'^<[a-zA-Z][a-zA-Z0-9\_.-]+@[a-zA-Z]+\.[a-zA-Z]{1,3}>$'

     return re.match(pattern, email1) is not None

for i in range(n):

     if fun(email[i]):

         print(name[i],email[i])

* You are given a string N.Your task is to verify that N is a floating point number.

In this task, a valid float number must satisfy all of the following requirements:

 Number can start with +, - or . symbol.  
For example:  
✔+4.50  
✔-1.0  
✔.5  
✔-.7  
✔+.4  
✖ -+4.5

 Number must contain at least 1 decimal value.  
For example:  
✖ 12.  
✔12.0

 Number must have exactly one . symbol.  
 Number must not give any exceptions when converted using float(N).

**Input Format**:The first line contains an integer T, the number of test cases.  
The next T line(s) contains a string S.

**Output Format**: Output True or False for each test case.

**Sample Input 0**

4

4.0O0

-1.00

+4.54

SomeRandomStuff

**Sample Output 0**

False

True

True

False

Ans:- T=int(input())

s=[]

for i in range(T):

     s.append(input())

def is\_valid\_float(s):

     import re

     pattern = r"^[+-]?(\d+\.\d+|\.\d+)$"

     return re.match(pattern, s) is not None

for i in s:

     print(is\_valid\_float(i))

### [Exceptions](https://docs.python.org/2/tutorial/errors.html#exceptions):Errors detected during execution are called exceptions.

**Examples**:

[**ZeroDivisionError**](https://docs.python.org/2/library/exceptions.html#exceptions.ZeroDivisionError)  
This error is raised when the second argument of a division or modulo operation is zero.

[**ValueError**](https://docs.python.org/2/library/exceptions.html#exceptions.ValueError):This error is raised when a built-in operation or function receives an argument that has the right type but an inappropriate value.

### [Handling Exceptions](https://docs.python.org/2/tutorial/errors.html#handling-exceptions): The statements try and except can be used to handle selected exceptions. A try statement may have more than one except clause to specify handlers for different exceptions.

You are given two values a and b.  
Perform integer division and print a/b.

**Input Format**: The first line contains T, the number of test cases.  
The next T lines each contain the space separated values of a and b.

**Output Format**

Print the value of a/b.  
In the case of ZeroDivisionError or ValueError, print the error code.

**Sample Input**

3

1 0

2 $

3 1

**Sample Output**

Error Code: integer division or modulo by zero

Error Code: invalid literal for int() with base 10: '$'

3

Ans:- T=int(input())

a, b=[], []

for i in range(T):

     temp\_a, temp\_b=input().split()

     a.append(temp\_a)

     b.append(temp\_b)

for i in range(T):

     try:

         print(int(a[i])//int(b[i]))

     except ZeroDivisionError:

print("Error Code: integer division or modulo by zero")

     except ValueError as e:

         print("Error Code:", e)

* [**shape**](http://docs.scipy.org/doc/numpy/reference/generated/numpy.ndarray.shape.html#numpy-ndarray-shape)

The *shape* tool gives a tuple of array dimensions and can be used to change the dimensions of an array.

**(a). Using *shape* to get array dimensions**

import numpy

my\_\_1D\_array = numpy.array([1, 2, 3, 4, 5])

print my\_1D\_array.shape #(5,) -> 1 row and 5 columns

my\_\_2D\_array = numpy.array([[1, 2],[3, 4],[6,5]])

print my\_2D\_array.shape #(3, 2) -> 3 rows and 2 columns

[**reshape**](http://docs.scipy.org/doc/numpy/reference/generated/numpy.reshape.html#numpy.reshape)

The *reshape* tool gives a new shape to an array without changing its data. It creates a new array and does not modify the original array itself.

import numpy

my\_array = numpy.array([1,2,3,4,5,6])

print numpy.reshape(my\_array,(3,2))

#Output

[[1 2]

[3 4]

**Task:**

You are given a space separated list of nine integers. Your task is to convert this list into a X *NumPy* array.

**Input Format**

A single line of input containing  space separated integers.

**Output Format**

Print the X *NumPy* array.

**Sample Input**

1 2 3 4 5 6 7 8 9

**Sample Output**

[[1 2 3]

[4 5 6]

[7 8 9]]

Ans:- import numpy as np

l=np.array(list(map(int,input().split())))

print(np.reshape(l,(3,3)))

* **Input Format**

A single line of input containing space separated numbers.

**Output Format**

Print the reverse NumPy array with type float.

**Sample Input**

1 2 3 4 -8 -10

**Sample Output**

[-10. -8. 4. 3. 2. 1.]

Ans:- import numpy

arrays=lambda arr: numpy.array(arr[::-1],float)

arr = input().strip().split()

result = arrays(arr)

print(result)

* [**Transpose**](http://docs.scipy.org/doc/numpy/reference/generated/numpy.transpose.html#numpy-transpose)

We can generate the transposition of an array using the tool numpy.transpose.  
It will not affect the original array, but it will create a new array.

**Task**

You are given a NXM integer array matrix with space separated elements (N = rows and M = columns).  
Your task is to print the *transpose* and *flatten* results.

**Input Format**:The first line contains the space separated values of N and M.  
The next N lines contains the space separated elements of M columns.

**Output Format**:First, print the *transpose* array and then print the *flatten*.

**Sample Input**

2 2

1 2

3 4

**Sample Output**

[[1 3]

[2 4]]

[1 2 3 4]

Ans:- import numpy as np

N, M=list(map(int, input().split()))

l=[]

for i in range(N):

     l.append(input().split())

print(np.transpose(np.array(l)))

print(np.array(l).flatten())

* [**sum**](http://docs.scipy.org/doc/numpy/reference/generated/numpy.sum.html):The *sum* tool returns the sum of array elements over a given axis.

import numpy

my\_array = numpy.array([ [1, 2], [3, 4] ])

print numpy.sum(my\_array, axis = 0) #Output : [4 6]

print numpy.sum(my\_array, axis = 1) #Output : [3 7]

print numpy.sum(my\_array, axis = None) #Output : 10

print numpy.sum(my\_array) #Output : 10

By default, the axis value is None. Therefore, it performs a sum over all the dimensions of the input array.

[**prod**](http://docs.scipy.org/doc/numpy/reference/generated/numpy.prod.html):The *prod* tool returns the product of array elements over a given axis.

import numpy

my\_array = numpy.array([ [1, 2], [3, 4] ])

print numpy.prod(my\_array, axis = 0) #Output : [3 8]

print numpy.prod(my\_array, axis = 1) #Output : [ 2 12]

print numpy.prod(my\_array, axis = None) #Output : 24

print numpy.prod(my\_array) #Output : 24

By default, the axis value is None. Therefore, it performs the product over all the dimensions of the input array.

**Task**:You are given a 2-D array with dimensions N XM.  
Your task is to perform the sum tool over axis 0 and then find the product of that result.

**Input Format**:The first line of input contains space separated values of N and M.  
The next N lines contains M space separated integers.

**Output Format**:Compute the sum along axis 0. Then, print the product of that sum.

**Sample Input**

2 2

1 2

3 4

**Sample Output**

24

**Explanation**

The sum along axis  0= [4 6 ]  
The product of this sum = 24

Ans:- import numpy as np

N, M=list(map(int, input().split()))

l=[]

for i in range(N):

     l.append(input().split())

print(np.prod(np.sum(np.array(l),axis=0)))

* [**Concatenate**](http://docs.scipy.org/doc/numpy/reference/generated/numpy.concatenate.html): 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)

#Output

[[1 2 3 0 0 0]

[0 0 0 7 8 9]]

**Task**

You are given two integer arrays of size NXP and MXP (N & M are rows, and P is the column). Your task is to *concatenate* the arrays along axis 0.

**Input Format**:The first line contains space separated integers N,M  and P.  
The next N lines contains the space separated elements of the P columns.  
After that, the next M lines contains the space separated elements of the P columns.

**Output Format**: Print the concatenated array of size (N+M)XP.

**Sample Input**

4 3 2

1 2

1 2

1 2

1 2

3 4

3 4

3 4

**Sample Output**

[[1 2]

[1 2]

[1 2]

[1 2]

[3 4]

[3 4]

[3 4]]

Ans:- import numpy as np

N, M, P=list(map(int, input().split()))

l1,l2=[],[]

for i in range(N):

     l1.append(input().split())

for i in range(M):

l2.append(input().split())

print(np.concatenate((np.array(l1,int),np.array(l2,int)),axis=0))

* Basic mathematical functions operate element-wise on arrays. They are available both as operator overloads and as functions in the *NumPy* module.

import numpy

a = numpy.array([1,2,3,4], float)

b = numpy.array([5,6,7,8], float)

print a + b #[ 6. 8. 10. 12.]

print numpy.add(a, b) #[ 6. 8. 10. 12.]

print a - b #[-4. -4. -4. -4.]

print numpy.subtract(a, b) #[-4. -4. -4. -4.]

print a \* b #[ 5. 12. 21. 32.]

print numpy.multiply(a, b) #[ 5. 12. 21. 32.]

print a / b #[ 0.2 0.33333333 0.42857143 0.5 ]

print numpy.divide(a, b) #[ 0.2 0.33333333 0.42857143 0.5 ]

print a % b #[ 1. 2. 3. 4.]

print numpy.mod(a, b) #[ 1. 2. 3. 4.]

print a\*\*b #[ 1.00000000e+00 6.40000000e+01 2.18700000e+03 6.55360000e+04]

print numpy.power(a, b) #[ 1.00000000e+00 6.40000000e+01 2.18700000e+03 6.55360000e+04]

**Task**:You are given two integer arrays A  and B of dimensions NXM.  
Your task is to perform the following operations:

1. Add ( A+B )
2. Subtract (A -B )
3. Multiply ( A\*B )
4. Integer Division ( A//B )
5. Mod ( A%B )
6. Power ( A\*\*B )

**Note**:There is a method numpy.floor\_divide() that works like numpy.divide() except it performs a floor division.

**Input Format**:The first line contains two space separated integers,  N and M.  
The next N lines contains M space separated integers of array A.  
The following N lines contains M space separated integers of array B.

**Output Format**:Print the result of each operation in the given order under **Task**.

**Sample Input**

1 4

1 2 3 4

5 6 7 8

**Sample Output**

[[ 6 8 10 12]]

[[-4 -4 -4 -4]]

[[ 5 12 21 32]]

[[0 0 0 0]]

[[1 2 3 4]]

[[ 1 64 2187 65536]]

Ans:- import numpy as np

N,M=list(map(int,input().split()))

A,B=[],[]

for i in range(N):

     A.append(input().split())

for i in range(N):

     B.append(input().split())

A, B=np.array(A,int), np.array(B,int)

print(A+B,A-B,A\*B,A//B,A%B,A\*\*B,sep="\n")

* [**min**](http://docs.scipy.org/doc/numpy/reference/generated/numpy.ndarray.min.html): The tool *min* returns the minimum value along a given axis.

import numpy

my\_array = numpy.array([[2, 5],

[3, 7],

[1, 3],

[4, 0]])

print numpy.min(my\_array, axis = 0) #Output : [1 0]

print numpy.min(my\_array, axis = 1) #Output : [2 3 1 0]

print numpy.min(my\_array) #Output : 0

By default, the axis value is None. Therefore, it finds the minimum over all the dimensions of the input array.

[**max**](http://docs.scipy.org/doc/numpy/reference/generated/numpy.ndarray.max.html): The tool *max* returns the maximum value along a given axis.

import numpy

my\_array = numpy.array([[2, 5],

[3, 7],

[1, 3],

[4, 0]])

print numpy.max(my\_array, axis = 0) #Output : [4 7]

print numpy.max(my\_array) #Output : 7

By default, the axis value is None. Therefore, it finds the maximum over all the dimensions of the input array.

**Task:** You are given a 2-D array with dimensions NXM.  
Your task is to perform the *min* function over axis  1 and then find the *max* of that.

**Input Format:** The first line of input contains the space separated values of N and M.  
The next N lines contains M space separated integers.

**Output Format:** Compute the *min* along axis 1 and then print the *max* of that result.

**Sample Input**

4 2

2 5

3 7

1 3

4 0

**Sample Output**

3

**Explanation**

The *min* along axis  1=[2,3,1,0]   
The *max* of [2,3,1,0]  = 3

Ans:- import numpy as np

N,M=list(map(int,input().split()))

A=[]

for i in range(N):

    A.append(input().split())

print(np.max(np.min(np.array(A, int),axis=1)))

* [**floor**](http://docs.scipy.org/doc/numpy/reference/generated/numpy.floor.html#numpy-floor): The tool *floor* returns the floor of the input element-wise.

import numpy

my\_array = numpy.array([1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7, 8.8, 9.9])

print numpy.floor(my\_array) #[ 1. 2. 3. 4. 5. 6. 7. 8. 9.]

[**ceil**](http://docs.scipy.org/doc/numpy/reference/generated/numpy.ceil.html#numpy-ceil): The tool *ceil* returns the ceiling of the input element-wise.  
The ceiling of  is the smallest integer  where .

import numpy

my\_array = numpy.array([1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7, 8.8, 9.9])

print numpy.ceil(my\_array) #[ 2. 3. 4. 5. 6. 7. 8. 9. 10.]

[**rint**](http://docs.scipy.org/doc/numpy/reference/generated/numpy.rint.html): The *rint* tool rounds to the nearest integer of input element-wise.

import numpy

my\_array = numpy.array([1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7, 8.8, 9.9])

print numpy.rint(my\_array) #[ 1. 2. 3. 4. 6. 7. 8. 9. 10.]

**Task:** You are given a 1-D array, A. Your task is to print the floor, ceil  and rint of all the elements of A.

**Note:** In order to get the correct output format, add the line numpy.set\_printoptions(legacy='1.13')below the numpy import.

**Input Format**: A single line of input containing the space separated elements of array A.

**Output Format:** On the first line, print the floor of A.  
On the second line, print the ceil of A.  
On the third line, print the rint of A.

**Sample Input**

1.1 2.2 3.3 4.4 5.5 6.6 7.7 8.8 9.9

**Sample Output**

[ 1. 2. 3. 4. 5. 6. 7. 8. 9.]

[ 2. 3. 4. 5. 6. 7. 8. 9. 10.]

[ 1. 2. 3. 4. 6. 7. 8. 9. 10.]

Ans:- import numpy as np

np.set\_printoptions(legacy='1.13')

A=np.array(input().split(),float)

print(np.floor(A),np.ceil(A),np.rint(A),sep="\n")

* [**mean**](http://docs.scipy.org/doc/numpy/reference/generated/numpy.mean.html): The *mean* tool computes the arithmetic mean along the specified axis.

import numpy

my\_array = numpy.array([ [1, 2], [3, 4] ])

print numpy.mean(my\_array, axis = 0) #Output : [ 2. 3.]

print numpy.mean(my\_array) #Output : 2.5

By default, the axis is None. Therefore, it computes the mean of the flattened array.

[**var**](http://docs.scipy.org/doc/numpy/reference/generated/numpy.var.html#numpy-var): The *var* tool computes the arithmetic variance along the specified axis.

import numpy

my\_array = numpy.array([ [1, 2], [3, 4] ])

print numpy.var(my\_array, axis = 0) #Output : [ 1. 1.]

print numpy.var(my\_array) #Output : 1.25

By default, the axis is None. Therefore, it computes the variance of the flattened array.

[**std**](http://docs.scipy.org/doc/numpy/reference/generated/numpy.std.html#numpy.std): The *std* tool computes the arithmetic standard deviation along the specified axis.

import numpy

my\_array = numpy.array([ [1, 2], [3, 4] ])

print numpy.std(my\_array, axis = 0) #Output : [ 1. 1.]

print numpy.std(my\_array) #Output : 1.11803398875

By default, the axis is None. Therefore, it computes the standard deviation of the flattened array.

**Task:** You are given a 2-D array of size NXM.  
Your task is to find:

1. The mean along axis 1
2. The var along axis 0
3. The std along axis None

**Input Format:**The first line contains the space separated values of N and M.  
The next N lines contains M space separated integers.

**Output Format:** First, print the *mean*.  
Second, print the *var*.  
Third, print the *std*.

**Sample Input**

2 2

1 2

3 4

**Sample Output**

[ 1.5 3.5]

[ 1. 1.]

1.11803398875

Ans:- import numpy as np

N, M=list(map(int,input().split()))

A=[]

for i in range(N):

     A.append(input().split())

A=np.array(A, float) print(np.mean(A, axis=1),np.var(A, axis=0),np.around(np.std(A),decimals=11),sep="\n")

* 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

**Function Description**: Complete the *print\_formatted* function in the editor below.

*print\_formatted* has the following parameters:

* *int number:* the maximum value to print

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

**Input Format:** A single integer denoting 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 15 D 1101

14 16 E 1110

15 17 F 1111

16 20 10 10000

17 21 11 10001

Ans:- def print\_formatted(number):

    width = len(bin(number)[2:])

     for i in range(1, number + 1):

        print(f"{i:{width}d} {i:{width}o} {i:{width}X} {i:{width}b}")

n=int(input())

     print\_formatted(n)

* In Python, a string of text can be aligned *left, right* and *center*.

**.ljust(width)**

This method returns a left aligned string of length *width*.

>>> width = 20

>>> print 'HackerRank'.ljust(width,'-')

HackerRank----------

**.center(width)**

This method returns a centered string of length *width*.

>>> width = 20

>>> print 'HackerRank'.center(width,'-')

-----HackerRank-----

**.rjust(width)**

This method returns a right aligned string of length *width*.

>>> width = 20

>>> print 'HackerRank'.rjust(width,'-')

----------HackerRank

**Task**

You are given a partial code that is used for generating the *HackerRank Logo* of variable *thickness*.

**Input Format**

A single line containing the *thickness* value for the logo.

**Constraints**

The *thickness* must be an *odd* number.

**Output Format**

Output the desired logo.

**Sample Input**

5

**Sample Output**

H

HHH

HHHHH

HHHHHHH

HHHHHHHHH

HHHHH HHHHH

HHHHH HHHHH

HHHHH HHHHH

HHHHH HHHHH

HHHHH HHHHH

HHHHH HHHHH

HHHHHHHHHHHHHHHHHHHHHHHHH

HHHHHHHHHHHHHHHHHHHHHHHHH

HHHHHHHHHHHHHHHHHHHHHHHHH

HHHHH HHHHH

HHHHH HHHHH

HHHHH HHHHH

HHHHH HHHHH

HHHHH HHHHH

HHHHH HHHHH

HHHHHHHHH

HHHHHHH

HHHHH

HHH

H

Ans:-

thickness =int(input()) #This must be an odd number

c = 'H'

#Top Cone

for i in range(thickness):

    print((c\*i).rjust(thickness-1)+c+(c\*i).ljust(thickness-1))

#Top Pillars

for i in range(thickness+1):

    print((c\*thickness).center(thickness\*2)+(c\*thickness).center(thickness\*6))

#Middle Belt

for i in range((thickness+1)//2):

    print((c\*thickness\*5).center(thickness\*6))

#Bottom Pillars

for i in range(thickness+1):

    print((c\*thickness).center(thickness\*2)+(c\*thickness).center(thickness\*6))

#Bottom Cone

for i in range(thickness):

    print(((c\*(thickness-i-1)).rjust(thickness)+c+(c\*(thickness-i-1)).ljust(thickness)).rjust(thickness\*6))

* If we want to add a single element to an existing set, we can use the .add() operation.

**Example:** >>> s = set('HackerRank')

>>> s.add('H')

>>> print s

**Task**: Apply your knowledge of the .add() operation to help your friend Rupal.  
Rupal has a huge collection of country stamps. She decided to count the total number of distinct country stamps in her collection. She asked for your help. You pick the stamps one by one from a stack of N country stamps.  
Find the total number of distinct country stamps.

**Input Format**: The first line contains an integer N, the total number of country stamps.  
The next N lines contains the name of the country where the stamp is from.

**Output Format**: Output the total number of distinct country stamps on a single line.

**Sample Input**

7

UK

China

USA

France

New Zealand

UK

France

**Sample Output**

5

Ans:- N=int(input())

print(len({input() for i in range(N)}))

* **.union()**

The *.union()* operator returns the union of a set and the set of elements in an iterable.  
Sometimes, the *|* operator is used in place of *.union()* operator, but it operates only on the set of elements in *set*.  
Set is immutable to the *.union()* operation (or *|* operation).

**Example**: >>> s = set("Hacker")

>>> print s.union("Rank")

set(['a', 'R', 'c', 'r', 'e', 'H', 'k', 'n'])

>>> print s.union(set(['R', 'a', 'n', 'k']))

set(['a', 'R', 'c', 'r', 'e', 'H', 'k', 'n'])

>>> print s.union(['R', 'a', 'n', 'k'])

set(['a', 'R', 'c', 'r', 'e', 'H', 'k', 'n'])

>>> print s.union(enumerate(['R', 'a', 'n', 'k']))

set(['a', 'c', 'r', 'e', (1, 'a'), (2, 'n'), 'H', 'k', (3, 'k'), (0, 'R')])

>>> print s.union({"Rank":1})

set(['a', 'c', 'r', 'e', 'H', 'k', 'Rank'])

>>> s | set("Rank")

set(['a', 'R', 'c', 'r', 'e', 'H', 'k', 'n'])

**Task**: The students of District College have subscriptions to *English* and *French* newspapers. Some students have subscribed only to *English*, some have subscribed to only *French* and some have subscribed to both newspapers.

You are given two sets of student roll numbers. One set has subscribed to the *English* newspaper, and the other set is subscribed to the *French* newspaper. The same student could be in both sets. Your task is to find the total number of students who have subscribed to *at least one* newspaper.

**Input Format**: The first line contains an integer, n, the number of students who have subscribed to the *English* newspaper.  
The second line contains n space separated roll numbers of those students.  
The third line contains b, the number of students who have subscribed to the *French* newspaper.  
The fourth line contains b space separated roll numbers of those students.

**Output Format**: Output the total number of students who have *at least one* subscription.

**Sample Input**

9

1 2 3 4 5 6 7 8 9

9

10 1 2 3 11 21 55 6 8

**Sample Output**: 13

Ans:- n=int(input())

s1=set(input().split())

b=int(input())

s2=set(input().split())

print(len(s1| s2))

* **.intersection()**

The .intersection() operator returns the intersection of a set and the set of elements in an iterable.  
Sometimes, the & operator is used in place of the .intersection() operator, but it only operates on the set of elements in set.  
The set is immutable to the .intersection() operation (or & operation).

>>> s = set("Hacker")

>>> print s.intersection("Rank")

set(['a', 'k'])

>>> print s.intersection(set(['R', 'a', 'n', 'k']))

set(['a', 'k'])

>>> print s.intersection(['R', 'a', 'n', 'k'])

set(['a', 'k'])

>>> print s.intersection(enumerate(['R', 'a', 'n', 'k']))

set([])

>>> print s.intersection({"Rank":1})

set([])

>>> s & set("Rank")

set(['a', 'k'])

**Continuation to the above question.**

**Output Format**: Output the total number of students who have subscriptions to **both** English and French newspapers.

Ans:- n=int(input())

s1=set(input().split())

b=int(input())

s2=set(input().split())

print(len(s1& s2))

* **.difference()**: The tool .difference() returns a set with all the elements from the set that are not in an iterable.  
  Sometimes the - operator is used in place of the .difference() tool, but it only operates on the set of elements in set.  
  Set is immutable to the .difference() operation (or the - operation).

>>> s = set("Hacker")

>>> print s.difference("Rank")

set(['c', 'r', 'e', 'H'])

>>> print s.difference(set(['R', 'a', 'n', 'k']))

set(['c', 'r', 'e', 'H'])

>>> print s.difference(['R', 'a', 'n', 'k'])

set(['c', 'r', 'e', 'H'])

>>> print s.difference(enumerate(['R', 'a', 'n', 'k']))

set(['a', 'c', 'r', 'e', 'H', 'k'])

>>> print s.difference({"Rank":1})

set(['a', 'c', 'e', 'H', 'k', 'r'])

>>> s - set("Rank")

set(['H', 'c', 'r', 'e'])

**Continuation to the above question.**

**Output Format**: Output the total number of students who are subscribed to the English newspaper only.

Ans:- n=int(input())

s1=set(input().split())

b=int(input())

s2=set(input().split())

print(len(s1-s2))

* **.symmetric\_difference()**: The .symmetric\_difference() operator returns a set with all the elements that are in the set and the iterable but not both.  
  Sometimes, a ^ operator is used in place of the .symmetric\_difference() tool, but it only operates on the set of elements in set.  
  The set is immutable to the .symmetric\_difference() operation (or ^ operation).

>>> s = set("Hacker")

>>> print s.symmetric\_difference("Rank")

set(['c', 'e', 'H', 'n', 'R', 'r'])

>>> print s.symmetric\_difference(set(['R', 'a', 'n', 'k']))

set(['c', 'e', 'H', 'n', 'R', 'r'])

>>> print s.symmetric\_difference(['R', 'a', 'n', 'k'])

set(['c', 'e', 'H', 'n', 'R', 'r'])

>>> print s.symmetric\_difference(enumerate(['R', 'a', 'n', 'k']))

set(['a', 'c', 'e', 'H', (0, 'R'), 'r', (2, 'n'), 'k', (1, 'a'), (3, 'k')])

>>> print s.symmetric\_difference({"Rank":1})

set(['a', 'c', 'e', 'H', 'k', 'Rank', 'r'])

>>> s ^ set("Rank")

set(['c', 'e', 'H', 'n', 'R', 'r'])

**Continuation to the above question.**

**Output Format**: Output total number of students who have subscriptions to the English or the French newspaper but not both.

Ans:- n=int(input())

s1=set(input().split())

b=int(input())

s2=set(input().split())

print(len(s1^ s2))

* You are given two sets, A and B.  
  Your job is to find whether set  A is a subset of set B.  
  If set A is subset of set B, print **True**.  
  If set A is not a subset of set B, print **False**.

**Input Format:** The first line will contain the number of test cases,T .  
The first line of each test case contains the number of elements in set A.  
The second line of each test case contains the space separated elements of set A.  
The third line of each test case contains the number of elements in set B.  
The fourth line of each test case contains the space separated elements of set B.

**Output Format**: Output **True** or **False** for each test case on separate lines.

**Sample Input**

3

5

1 2 3 5 6

9

9 8 5 6 3 2 1 4 7

1

2

5

3 6 5 4 1

7

1 2 3 5 6 8 9

3

9 8 2

**Sample Output**: True

False

False

Ans:- T=int(input())

for i in range(T):

     nA=int(input())

     A=set(input().split())

     nB=int(input())

     B=set(input().split())

     print(A.issubset(B))

* You are given a set A and n other sets.  
  Your job is to find whether set A is a strict superset of each of the N sets.

Print True, if A is a *strict superset* of each of the N sets. Otherwise, print False.

A strict superset has at least one element that does not exist in its subset.

**Input Format**: The first line contains the space separated elements of set A.  
The second line contains integer n, the number of other sets.  
The next n lines contains the space separated elements of the other sets.

**Output Format**: Print True if set A is a *strict superset* of all other N sets. Otherwise, print False.

**Sample Input:** 1 2 3 4 5 6 7 8 9 10 11 12 23 45 84 78

2

1 2 3 4 5

100 11 12

**Sample Output:** False

Ans:- A=set(input().split())

n=int(input())

B = set().union(\*[set(input().split()) for \_ in range(n)])

print(A.issuperset(B))

### [any()](https://docs.python.org/2/library/functions.html#any): This expression returns True if **any** element of the iterable is true. If the iterable is empty, it will return False.

### [all()](https://docs.python.org/2/library/functions.html#all): This expression returns True if **all** of the elements of the iterable are true. If the iterable is empty, it will return True.

**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](https://en.wikipedia.org/wiki/Palindromic_number).

**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.

**Output Format:** Print True if all the conditions of the problem statement are satisfied. Otherwise, print False.

**Sample Input:** 5

12 9 61 5 14

**Sample Output**: True

**Explanation**: **Condition 1**: All the integers in the list are positive.  
**Condition 2**: 5 is a palindromic integer.

Ans:- N=int(input())

l=list(map(int, input().split()))

print(all(i > 0 for i in l) and any(str(i) == str(i)[::-1] for i in l))

* You are given a string S.  
  S contains alphanumeric characters only.  
  A close up of a letter

  Description automatically generated Your task is to sort the string S in the following manner:
* All sorted *lowercase letters* are ahead of *uppercase letters*.
* All sorted *uppercase letters* are ahead of digits.
* All sorted *odd digits* are ahead of sorted *even digits*.

**Input Format:** A single line of input contains the string S.

**Output Format**: Output the sorted string S.

**Sample Input**: Sorting1234

**Sample Output**: ginortS1324

Ans:- s=input()

lower\_case = sorted([i for i in s if i.islower()])

upper\_case = sorted([i for i in s if i.isupper()])

odd\_digits = sorted([i for i in s if i.isdigit() and int(i)%2!= 0])

even\_digits = sorted([i for i in s if i.isdigit() and int(i)%2==0])

print(''.join(lower\_case+upper\_case+odd\_digits+even\_digits))

* You are given a string s consisting only of digits 0-9, commas ,, and dots .

Your task is to complete the regex\_pattern defined below, which will be used to re.split() all of the , and . symbols in s.

It’s guaranteed that every comma and every dot in s is preceeded and followed by a digit.

**Sample Input:** 100,000,000.000

**Sample Output:** 100

000

000

000

Ans:- regex\_pattern = "[,.]"

import re

print("\n".join(re.split(regex\_pattern, input())))

* Let's dive into the interesting topic of regular expressions! You are given some input, and you are required to check whether they are valid mobile numbers.

A valid mobile number is a ten digit number starting with a 7,8 or 9.

<https://developers.google.com/edu/python/regular-expressions>

**Input Format**

The first line contains an integer N, the number of inputs.  
 N lines follow, each containing some string.

**Output Format**: For every string listed, print "YES" if it is a valid mobile number and "NO" if it is not on separate lines. Do not print the quotes.

**Sample Input:** 2

9587456281

1252478965

**Sample Output:** YES

NO

Ans:- import re

l=[input() for i in range(int(input()))]

for i in l: print("YES") if re.match(r"^[789]\d{9}$",i) else print("NO")

* You are given a string, and you have to validate whether it's a valid Roman numeral. If it is valid, print True. Otherwise, print False. Try to create a regular expression for a valid Roman numeral.

**Input Format**: A single line of input containing a string of Roman characters.

**Output Format:** Output a single line containing True or False according to the instructions above.

**Sample Input:** CDXXI

**Sample Output:** True

Ans:- regex\_pattern = r'^M{0,3}(CM|CD|D?C{0,3})(XC|XL|L?X{0,3})(IX|IV|V?I{0,3})$'

import re

print(str(bool(re.match(regex\_pattern, input()))))

* Consider a list (list = []). You can perform the following commands:

1. insert i e: Insert integer e at position i.
2. print: Print the list.
3. remove e: Delete the first occurrence of integer e.
4. append e: Insert integer e at the end of the list.
5. sort: Sort the list.
6. pop: Pop the last element from the list.
7. reverse: Reverse the list.

**Input Format:** The first line contains an integer,n , denoting the number of commands.  
Each line i of the n subsequent lines contains one of the commands described above.

**Output Format**: For each command of type print, print the list on a new line.

**Sample Input:** 12

insert 0 5

insert 1 10

insert 0 6

print

remove 6

append 9

append 1

sort

print

pop

reverse

print

**Sample Output:** [6, 5, 10]

[1, 5, 9, 10]

[9, 5, 1]

Ans:-l=[]

n=int(input())

for \_ in range(n):

     command = input().split()

     if command[0] == "insert":

        # Insert integer e at position i

         i, e = map(int, command[1:])

        l.insert(i, e)

     elif command[0] == "print":

         # Print the list

         print(l)

     elif command[0] == "remove":

         # Delete the first occurrence of integer e

         e = int(command[1])

         l.remove(e)

     elif command[0] == "append":

         # Insert integer e at the end of the list

         e = int(command[1])

         l.append(e)

     elif command[0] == "sort":

         # Sort the list

         l.sort()

     elif command[0] == "pop":

         # Pop the last element from the list

         l.pop()

     elif command[0] == "reverse":

         # Reverse the list

         l.reverse()

* Given the names and grades for each student in a class of N 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 second lowest grade, order their names alphabetically and print each name on a new line.

**Input Format:** The first line contains an integer, N, the number of students.  
The 2N  subsequent lines describe each student over 2 lines.  
- The first line contains a student's name.  
- The second line contains their grade.

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

**Sample Input:** 5

Harry

37.21

Berry

37.21

Tina

37.2

Akriti

41

Harsh

39

**Sample Output:** Berry

Harry

Ans:- n = int(input())

s=[[input(), float(input())] for \_ in range(n)]

# Sort the s list based on grades in ascending order

s=sorted(s, key=lambda x: x[1])

# Find the second lowest grade among the students

slg=[s[i][1] for i in range(n) if s[i][1] != s[0][1]][0]

# Print the names of students with the slg

print("\n".join(sorted([s[i][0] for i in range(n) if s[i][1] == slg])))

* If the inputs are given on one line separated by a character (the delimiter), use split() to get the separate values in the form of a list. The delimiter is space (ascii 32) by default. To specify that comma is the delimiter, use string.split(',').

>> a = raw\_input()

5 4 3 2

>> lis = a.split()

>> print (lis)

['5', '4', '3', '2']

If the list values are all integer types, use the map() method to convert all the strings to integers.

>> newlis = list(map(int, lis))

>> print (newlis)

[5, 4, 3, 2]

Sets are an unordered collection of unique values. A single set contains values of any immutable data type.  
**CREATING SETS:** >> myset = {1, 2} # Directly assigning values to a set

>> myset = set() # Initializing a set

>> myset = set(['a', 'b']) # Creating a set from a list

>> myset

{'a', 'b'}

**MODIFYING SETS**: Using the add() function:

>> myset.add('c')

>> myset

{'a', 'c', 'b'}

>> myset.add('a') # As 'a' already exists in the set, nothing happens

>> myset.add((5, 4))

>> myset

{'a', 'c', 'b', (5, 4)}  
Using the update() function:

>> myset.update([1, 2, 3, 4]) # update() only works for iterable objects

>> myset

{'a', 1, 'c', 'b', 4, 2, (5, 4), 3}

>> myset.update({1, 7, 8})

>> myset

{'a', 1, 'c', 'b', 4, 7, 8, 2, (5, 4), 3}

>> myset.update({1, 6}, [5, 13])

>> myset

{'a', 1, 'c', 'b', 4, 5, 6, 7, 8, 2, (5, 4), 13, 3}  
**REMOVING ITEMS**: Both the discard() and remove() functions take a single value as an argument and removes that value from the set. If that value is not present, discard() does nothing, but remove() will raise a KeyError exception.

>> myset.discard(10)

>> myset

{'a', 1, 'c', 'b', 4, 5, 7, 8, 2, 12, (5, 4), 13, 11, 3}

>> myset.remove(13)

>> myset

{'a', 1, 'c', 'b', 4, 5, 7, 8, 2, 12, (5, 4), 11, 3}

**COMMON SET OPERATIONS** Using union(), intersection() and difference() functions.

>> a = {2, 4, 5, 9}

>> b = {2, 4, 11, 12}

>> a.union(b) # Values which exist in a or b

{2, 4, 5, 9, 11, 12}

>> a.intersection(b) # Values which exist in a and b

{2, 4}

>> a.difference(b) # Values which exist in a but not in b

{9, 5}  
The union() and intersection() functions are symmetric methods:

>> a.union(b) == b.union(a)

True

>> a.intersection(b) == b.intersection(a)

True

>> a.difference(b) == b.difference(a)

False

These [other built-in data structures in Python](http://www.thelearningpoint.net/computer-science/learning-python-programming-and-data-structures/learning-python-programming-and-data-structures--tutorial-4--built-in-data-structures-strings-lists-tuples-dictionaries-mutability) are also useful.

**Input Format:** The first line of input contains an integer,M .  
The second line contains M space-separated integers.  
The third line contains an integer, N.  
The fourth line contains N space-separated integers.

**Output Format**: Output the symmetric difference integers in ascending order, one per line.

**Sample Input:**

STDIN Function

----- --------

4 set a size M = 4

2 4 5 9 a = {2, 4, 5, 9}

4 set b size N = 4

2 4 11 12 b = {2, 4, 11, 12}

**Sample Output:** 5

9

11

12

Ans:- M=int(input())

s1=set(map(int,input().split()))

N=int(input())

s2=set(map(int,input().split()))

print(\*sorted(s1^s2),sep="\n")

* [zip([iterable, ...])](https://docs.python.org/2/library/functions.html#zip)

This function returns a list of tuples. The ith tuple contains the ith 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.

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.

**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

Ans:- N, X = map(int, input().split())

marks=[list(map(float, input().split())) for \_ in range(X)]

for i in range(N):

     print(f"{sum([marks[j][i] for j in range(X)])/X:.1f}")

* **Task**: You are given a [polynomial](https://en.wikipedia.org/wiki/Polynomial) P of a single indeterminate (or variable),x .  
  You are also given the values of x and k. Your task is to verify if P(x)=k.

**Input Format:** The first line contains the space separated values of x and k.  
The second line contains the polynomial P.

**Output Format**: Print True if P(x)=k. Otherwise, print False.

**Sample Input:** 1 4

x\*\*3 + x\*\*2 + x + 1

**Sample Output:** True

Ans:- x,k =map(int, input().split())

P=input()

print(eval(P)==k)

* The eval() expression is a very powerful built-in function of Python. It helps in evaluating an expression. The expression can be a Python statement.Here, eval() can also be used to work with Python keywords or defined functions and variables. These would normally be stored as strings.

>>> type(eval("len"))

<type 'builtin\_function\_or\_method'>

Without eval()

>>> type("len")

<type 'str'>

**Task**: You are given an expression in a line. Read that line as a string variable, such as var, and print the result using eval(var).

**Sample Input:** print(2 + 3)

**Sample Output:** 5

Ans:- eval(input())

* Mr. Vincent works in a door mat manufacturing company. One day, he designed a new door mat with the following specifications:
* Mat size must be NXM . (N is an odd natural number, and M is 3 times N.)
* The design should have 'WELCOME' written in the center.
* The design pattern should only use |, . and - characters.

**Input Format:** A single line containing the space separated values of  N and M.

**Output Format**: Output the design pattern.

**Sample Input:** 9 27

**Sample Output:**

------------.|.------------

---------.|..|..|.---------

------.|..|..|..|..|.------

---.|..|..|..|..|..|..|.---

----------WELCOME----------

---.|..|..|..|..|..|..|.---

------.|..|..|..|..|.------

---------.|..|..|.---------

------------.|.------------

Ans:- N, M = map(int, input().split())

for i in range(1, N, 2): print((i \* ".|.").center(M, "-"))

print("WELCOME".center(M, "-"))

for i in range(N - 2, -1, -2): print((i \* ".|.").center(M, "-"))

* [**dot**](http://docs.scipy.org/doc/numpy/reference/generated/numpy.dot.html): 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**](http://docs.scipy.org/doc/numpy/reference/generated/numpy.cross.html): 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

**Task:** You are given two arrays A and B. Both have dimensions of NXN.  
Your task is to compute their [matrix product](https://en.wikipedia.org/wiki/Matrix_multiplication#Matrix_product_.28two_matrices.29).

**Input Format:** The first line contains the integer N.  
The next N lines contains N space separated integers of array A.  
The following N lines contains N space separated integers of array B.

**Output Format:** Print the matrix multiplication of A and B.

**Sample Input:** 2

1 2

3 4

1 2

3 4

**Sample Output:** [[ 7 10]

[15 22]]

Ans:- import numpy as np

n = int(input())

A=np.array([list(map(int, input().split())) for \_ in range(n)])

B=np.array([list(map(int, input().split())) for \_ in range(n)])

print(np.dot(A, B))

* [**inner**](http://docs.scipy.org/doc/numpy/reference/generated/numpy.inner.html): The inner tool returns the [inner product](https://en.wikipedia.org/wiki/Inner_product_space) of two arrays.

import numpy

A = numpy.array([0, 1])

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

print numpy.inner(A, B) #Output : 4

[**outer**](http://docs.scipy.org/doc/numpy/reference/generated/numpy.outer.html): The outer tool returns the [outer product](https://en.wikipedia.org/wiki/Outer_product) of two arrays.

import numpy

A = numpy.array([0, 1])

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

print numpy.outer(A, B) #Output : [[0 0]

# [3 4]]

**Task:** You are given two arrays: A and B.  
Your task is to compute their inner and outer product.

**Input Format:** The first line contains the space separated elements of array A.  
The second line contains the space separated elements of array B.

**Output Format:** First, print the inner product.  
Second, print the outer product.

**Sample Input:** 0 1

2 3

**Sample Output:** 3

[[0 0]

[2 3]]

Ans:- import numpy as np

A=np.array(list(map(int, input().split())))

B=np.array(list(map(int, input().split())))

print(np.inner(A, B),np.outer(A, B),sep="\n")

* [zeros](http://docs.scipy.org/doc/numpy/reference/generated/numpy.zeros.html#numpy-zeros)**:** The zeros tool returns a new array with a given shape and type filled with 0's.

import numpy

print numpy.zeros((1,2)) #Default type is float

#Output : [[ 0. 0.]]

print numpy.zeros((1,2), dtype = numpy.int) #Type changes to int

#Output : [[0 0]]

[ones](http://docs.scipy.org/doc/numpy/reference/generated/numpy.ones.html#numpy-ones)**:** The ones tool returns a new array with a given shape and type filled with 1's.

import numpy

print numpy.ones((1,2)) #Default type is float

#Output : [[ 1. 1.]]

print numpy.ones((1,2), dtype = numpy.int) #Type changes to int

#Output : [[1 1]]

**Task:** You are given the shape of the array in the form of space-separated integers, each integer representing the size of different dimensions, your task is to print an array of the given shape and integer type using the tools numpy.zeros and numpy.ones.

**Input Format:** A single line containing the space-separated integers.

**Output Format:** First, print the array using the numpy.zeros tool and then print the array with the numpy.ones tool.

**Sample Input:** 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]]]

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

Ans:- import numpy as np

n=list(map(int,input().split()))

print(np.zeros(n,dtype=int),np.ones(n,dtype=int),sep="\n")

* **.remove(x):** This operation removes element  from the set.  
  If element  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])

**.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])

**.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([])

**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.

**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

Ans: n = int(input())

s = set(map(int, input().split()))

N = int(input())

for \_ in range(N):

      command = input().split()

     if command[0]=='pop':s.pop()

     elif command[0]=='remove':s.remove(int(command[1]))

     elif command[0]=='discard':s.discard(int(command[1]))

print(sum(s))

* **Input Format:** The first line contains an integer,n , denoting the number of elements in the tuple.  
  The second line contains n space-separated integers describing the elements in tuple t.

**Output Format:** Print the result of hash(t).

**Sample Input:** 2

1 2

**Sample Output:** 3713081631934410656

Ans: n = int(input())

print(hash(tuple(map(int, input().split()))))

* **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.

**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

Ans: n =int(input())

A =set(map(int,input().split()))

N =int(input())

for \_ in range(N):

     operation,\_=input().split()

     other\_set=set(map(int, input().split()))

     if operation=='intersection\_update':A&=other\_set

     elif operation=='update':A|=other\_set

     elif operation=='symmetric\_difference\_update':A^=other\_set

     elif operation=='difference\_update':A-=other\_set

print(sum(A))

* [**poly**](http://docs.scipy.org/doc/numpy/reference/generated/numpy.poly.html): 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**](http://docs.scipy.org/doc/numpy/reference/generated/numpy.roots.html): The roots tool returns the roots of a polynomial with the given coefficients.

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

[**polyint**](http://docs.scipy.org/doc/numpy/reference/generated/numpy.polyint.html): 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**](http://docs.scipy.org/doc/numpy/reference/generated/numpy.polyder.html#numpy.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**](http://docs.scipy.org/doc/numpy/reference/generated/numpy.polyval.html#numpy.polyval): The polyval tool evaluates the polynomial at specific value.

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

[**polyfit**](http://docs.scipy.org/doc/numpy/reference/generated/numpy.polyfit.html): 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](http://docs.scipy.org/doc/numpy/reference/generated/numpy.polyadd.html" \l "numpy.polyadd" \t "_blank), [polysub](http://docs.scipy.org/doc/numpy/reference/generated/numpy.polysub.html" \l "numpy.polysub), [polymul](http://docs.scipy.org/doc/numpy/reference/generated/numpy.polymul.html), and [polydiv](http://docs.scipy.org/doc/numpy/reference/generated/numpy.polydiv.html" \l "numpy.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

Ans: import numpy as np

P=list(map(float,input().split()))

x=float(input())

print(np.polyval(P,x))

* [**identity**](http://docs.scipy.org/doc/numpy/reference/generated/numpy.identity.html#numpy.identity): The *identity* tool returns an identity array. An identity array is a square matrix with all the main diagonal elements as 1 and the rest as 0. The default type of elements is float.

import numpy

print numpy.identity(3) #3 is for dimension 3 X 3

#Output: [[ 1. 0. 0.]

[ 0. 1. 0.]

[ 0. 0. 1.]]

[**eye**](http://docs.scipy.org/doc/numpy/reference/generated/numpy.eye.html#numpy-eye): The *eye* tool returns a 2-D array with 1's as the diagonal and 0's elsewhere. The diagonal can be main, upper or lower depending on the optional parameter k. A positive k is for the upper diagonal, a negative k is for the lower, and a 0 k (default) is for the main diagonal.

import numpy

print numpy.eye(8, 7, k = 1) # 8 X 7 Dimensional array with first upper diagonal 1.

#Output: [[ 0. 1. 0. 0. 0. 0. 0.]

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

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

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

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

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

[ 0. 0. 0. 0. 0. 0. 0.]

[ 0. 0. 0. 0. 0. 0. 0.]]

print numpy.eye(8, 7, k = -2) # 8 X 7 Dimensional array with second lower diagonal 1.

**Task:** Your task is to print an array of size NXM with its main diagonal elements as 1's and 0's everywhere else.

**Note**: In order to get alignment correct, please insert the line

np.set\_printoptions(legacy="1.13")

below the numpy import.

**Input Format**: A single line containing the space separated values of  N and M.  
N denotes the rows.  
M denotes the columns.

**Output Format:** Print the desired N X M array.

**Sample Input**:3 3

**Sample Output**: [[ 1. 0. 0.]

[ 0. 1. 0.]

[ 0. 0. 1.]]

Ans: import numpy as np

np.set\_printoptions(legacy="1.13")

a=np.array(input().split(),dtype=int)

print(np.eye(a[0],a[1]))

* A valid postal code P have to fullfil both below requirements:

1. P must be a number in the range from 100000 to 999999 inclusive.
2. P must not contain more than one alternating repetitive digit pair.

Alternating repetitive digits are digits which repeat immediately after the next digit. In other words, an alternating repetitive digit pair is formed by two equal digits that have just a single digit between them.

For example: 121426 # Here, 1 is an alternating repetitive digit.

**Input Format**: Locked stub code in the editor reads a single string denoting P from stdin and uses provided expression and your regular expressions to validate if P is a valid postal code.

**Output Format:** You are not responsible for printing anything to stdout. Locked stub code in the editor does that.

**Sample Input:** 110000

**Sample Output:** False

Ans: regex\_integer\_in\_range = r"^[1-9][0-9]{5}$"

regex\_alternating\_repetitive\_digit\_pair = r"(?=(\d)\d\1)"

import re

P=input()

print(bool(re.match(regex\_integer\_in\_range, P))

and len(re.findall(regex\_alternating\_repetitive\_digit\_pair, P)) < 2)

* Neo has a complex matrix script. The matrix script is a N X M grid of strings. It consists of alphanumeric characters, spaces and symbols (!,@,#,$,%,&).

A screenshot of a matrix script

Description automatically generated

To decode the script, Neo needs to read each column and select only the alphanumeric characters and connect them. Neo reads the column from top to bottom and starts reading from the leftmost column.

If there are symbols or spaces between two alphanumeric characters of the decoded script, then Neo replaces them with a single space ' ' for better readability.

Neo feels that there is no need to use 'if' conditions for decoding.

Alphanumeric characters consist of: [A-Z, a-z, and 0-9].

**Input Format:** The first line contains space-separated integers N (rows) and M (columns) respectively.  
The next N lines contain the row elements of the matrix script.

**Output Format:** Print the decoded matrix script.

**Sample Input:** 7 3

Tsi

h%x

i #

sM

$a

#t%

ir!

**Sample Output:** This is Matrix# %!

**Explanation:** The decoded script is:

This$#is% Matrix# %!

Neo replaces the symbols or spaces between two alphanumeric characters with a single space   ' ' for better readability.

So, the final decoded script is:

This is Matrix# %!

Ans: import re

n,m=list(map(int,input().split()))

l=[]

for \_ in range(n):l.append(input())

ttl=list(zip(\*l))

s=''

for i in ttl:s+=''.join(i)

print(re.sub(r'(?<=[a-zA-Z0-9])[!@#$%&\s]+(?=[a-zA-Z0-9])',' ',s))

* You and Fredrick are good friends. Yesterday, Fredrick received N credit cards from **ABCD Bank**. He wants to verify whether his credit card numbers are valid or not. You happen to be great at regex so he is asking for your help!

A valid credit card from **ABCD Bank** has the following characteristics:  
  
► It must start with a 4, 5 or 6.  
► It must contain exactly 16 digits.  
► It must only consist of digits (0-9).  
► It may have digits in groups of 4, separated by one hyphen **"-"**.  
► It must NOT use any other separator like '  ' , '\_', etc.  
► It must NOT have 4 or more consecutive repeated digits.

**Examples**:

**Valid Credit Card Numbers**

4253625879615786

4424424424442444

5122-2368-7954-3214

**Invalid Credit Card Numbers**

42536258796157867 #17 digits in card number → Invalid

4424444424442444 #Consecutive digits are repeating 4 or more times → Invalid

5122-2368-7954 - 3214 #Separators other than '-' are used → Invalid

44244x4424442444 #Contains non digit characters → Invalid

0525362587961578 #Doesn't start with 4, 5 or 6 → Invalid

**Input Format**: The first line of input contains an integer N.  
The next N lines contain credit card numbers.

**Output Format:** Print 'Valid' if the credit card number is valid. Otherwise, print 'Invalid'. Do not print the quotes.

**Sample Input:** 6

4123456789123456

5123-4567-8912-3456

61234-567-8912-3456

4123356789123456

5133-3367-8912-3456

5123 - 3567 - 8912 - 3456

**Sample Output:** Valid

Valid

Invalid

Valid

Invalid

Invalid

Comments:

1. [re.match(pattern, string) checks for a match only at the beginning of the string](https://stackoverflow.com/questions/58774029/differences-between-re-match-re-search-re-fullmatch). [It returns a match object if the beginning of the string matches the pattern, otherwise it returns None](https://stackoverflow.com/questions/58774029/differences-between-re-match-re-search-re-fullmatch).
2. [re.fullmatch(pattern, string) checks for a match over the entire string](https://stackoverflow.com/questions/58774029/differences-between-re-match-re-search-re-fullmatch). [It returns a match object if the whole string matches the pattern, otherwise it returns None](https://stackoverflow.com/questions/58774029/differences-between-re-match-re-search-re-fullmatch).

Ans: import re

p1=r'[456]\d{3}\-?\d{4}\-?\d{4}\-?\d{4}'

p2=r'(\d)\1{3,}'

for \_ in range(int(input())):

     card\_num=input()

     print("Valid") if (re.fullmatch(p1,card\_num) and not re.search(p2, card\_num.replace('-',''))) else print('Invalid')

* 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**](http://docs.scipy.org/doc/numpy/reference/generated/numpy.linalg.det.html): The linalg.det tool computes the determinant of an array.

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

[**linalg.eig**](http://docs.scipy.org/doc/numpy/reference/generated/numpy.linalg.eig.html): 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**](http://docs.scipy.org/doc/numpy/reference/generated/numpy.linalg.inv.html): 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](http://docs.scipy.org/doc/numpy/reference/routines.linalg.html)

**Task:** You are given a square matrix A with dimensions NXN. Your task is to find the determinant. Note: Round the answer to 2 places after the decimal.

**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

Ans: import numpy as np

A=[] for \_ in range(int(input())):A.append(list(map(float,input().split())))

print(round(np.linalg.det(A),2))

* 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.

**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.

**Sample Input:** 1222311

**Sample Output:** (1, 1) (3, 2) (1, 3) (2, 1)

Ans: S,c=input(),1

for i in range(len(S)):

     if i < len(S) - 1 and S[i] == S[i + 1]:c+=1

     else:print(f"({c}, {int(S[i])})", end=" ");c=1

* 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.

**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

Ans: from collections import Counter

l=[]

for \_ in range(int(input())):l.append(input())

count=Counter(l)

print(len(count))

print(\*count.values())

* Consider that vowels in the alphabet are a, e, i, o, u and y.

Function score\_words takes a list of lowercase words as an argument and returns a score as follows:

**The score of a single word is 2 if the word contains an even number of vowels. Otherwise, the score of this word is 1. The score for the whole list of words is the sum of scores of all words in the list.**

**Input Format:** The input is read by the provided locked code template. In the first line, there is a single integer n denoting the number of words. In the second line, there are n space-separated lowercase words.

**Output Format:** The output is produced by the provided and locked code template. It calls function score\_words with the list of words read from the input as the argument and prints the returned score to the output.

**Sample Input:** 2

hacker book

**Sample Output:** 4

Ans: is\_vowel=lambda letter:letter in ['a', 'e', 'i', 'o', 'u', 'y']

def score\_words(words):

    score = 0

    for word in words:

        num\_vowels = 0

        for letter in word:

            if is\_vowel(letter):num\_vowels+= 1

        score+=2 if num\_vowels%2==0 else 1

    return score

n = int(input())

words = input().split()

print(score\_words(words))

* Python supports a useful concept of default argument values. For each keyword argument of a function, we can assign a default value which is going to be used as the value of said argument if the function is called without it.

Debug the given function print\_from\_stream using the default value of one of its arguments.

The function has the following signature: def print\_from\_stream(n, stream)

This function should print the first n values returned by get\_next() method of stream object provided as an argument. Each of these values should be printed in a separate line.

Whenever the function is called without the stream argument, it should use an instance of EvenStream class defined in the code stubs below as the value of stream.

**Input Format:** The input is read by the provided locked code template. In the first line, there is a single integer q denoting the number of queries. Each of the following q lines contains a stream\_name followed by integer n, and it corresponds to a single test for your function.

**Output Format:** The output is produced by the provided and locked code template. For each of the queries (stream\_name, n), if the stream\_name is even then print\_from\_stream(n) is called. Otherwise, if the stream\_name is odd, then print\_from\_stream(n, OddStream()) is called.

**Sample Input:** 3

odd 2

even 3

odd 5

**Sample Output:** 1

3

0

2

4

1

3

5

7

9

Ans:

class EvenStream(object):

    def \_\_init\_\_(self):

        self.current = 0

    def get\_next(self):

        to\_return = self.current

        self.current += 2

        return to\_return

class OddStream(object):

    def \_\_init\_\_(self):

        self.current = 1

    def get\_next(self):

        to\_return = self.current

        self.current += 2

        return to\_return

def print\_from\_stream(n,stream=None):

    if stream is None:stream = EvenStream()

    for \_ in range(n):print(stream.get\_next())

queries = int(input())

for \_ in range(queries):

    stream\_name, n = input().split()

    n = int(n)

    if stream\_name == "even":

        print\_from\_stream(n)

    else:

        print\_from\_stream(n, OddStream())

* The *re.sub()* tool (*sub* stands for *substitution*) evaluates a pattern and, for each valid match, it calls a *method* (or *lambda*).  
  The *method* is called for all matches and can be used to modify strings in different ways.  
  The *re.sub()* method returns the modified string as an output.

**Task:** You are given a text of N lines. The text contains && and || symbols.  
Your task is to modify those symbols to the following:

&& → and

|| → or

Both && and || should have a *space* " " on both sides.

**Input Format:** The first line contains the integer, N.  
The next N lines each contain a line of the text.

**Output Format:** Output the modified text.

**Sample Input**: 11

a = 1;

b = input();

if a + b > 0 && a - b < 0:

start()

elif a\*b > 10 || a/b < 1:

stop()

print set(list(a)) | set(list(b))

#Note do not change &&& or ||| or & or |

#Only change those '&&' which have space on both sides.

#Only change those '|| which have space on both sides.

**Sample Output**: a = 1;

b = input();

if a + b > 0 and a - b < 0:

start()

elif a\*b > 10 or a/b < 1:

stop()

print set(list(a)) | set(list(b))

#Note do not change &&& or ||| or & or |

#Only change those '&&' which have space on both sides.

#Only change those '|| which have space on both sides

Ans: import re

text="\n".join([input() for \_ in range(int(input()))])

modified\_text=re.sub(r'(?<= )&&(?= )', 'and',text)

modified\_text=re.sub(r'(?<= )\|\|(?= )', 'or',modified\_text)

print(modified\_text)

* *ABCXYZ* company has up to 100 employees.The company decides to create a unique identification number (UID) for each of its employees.The company has assigned you the task of validating all the randomly generated UIDs.

A valid UID must follow the rules below:

* It must contain at least 2 uppercase English alphabet characters.
* It must contain at least 3 digits (0 - 9).
* It should only contain alphanumeric characters ( a-z , A - Z & 0 -9 ).
* No character should repeat.
* There must be exactly 10 characters in a valid UID.

**Input Format:** The first line contains an integer T, the number of test cases.  
The next T lines contains an employee's UID.

**Output Format:** For each test case, print 'Valid' if the UID is valid. Otherwise, print 'Invalid', on separate lines. Do not print the quotation marks.

**Sample Input:** 2

B1CD102354

B1CDEF2354

**Sample Output**: Invalid

Valid

Ans: import re

for \_ in range(int(input())):

     s = input()

     print('Valid') if len(re.findall(r'[A-Z]',s))>=2 and len(re.findall(r'[0-9]',s))>=3 and re.match(r'^[A-Za-z0-9]+$',s) and len(re.findall(r'(.).\*\1',s))==0 and len(s)==10 else print('Invalid')

* 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.

**Input Format**: A single line of input containing the string S.

**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:** aabbbccde

**Sample Output:** b 3

a 2

c 2

Ans: from collections import Counter

cd=Counter(sorted(input()))

for k, v in cd.most\_common(3):

     print(k, v)

### [start() & end()](https://docs.python.org/2/library/re.html#re.MatchObject.start): These expressions return the indices of the start and end of the substring matched by the group.

**Code:** >>> import re

>>> m = re.search(r'\d+','1234')

>>> m.end()

4

>>> m.start()

0

**Task:** You are given a string S.  
Your task is to find the indices of the start and end of string k in S.

**Input Format:** The first line contains the string S.  
The second line contains the string k.

**Output Format:** Print the tuple in this format: (start \_index, end \_index).  
If no match is found, print (-1, -1).

**Sample Input:** aaadaa

aa

**Sample Output:** (0, 1)

(1, 2)

(4, 5)

Ans: import re

S=input()

k=input()

matches = list(re.finditer(r"(?="+k+")", S))

if matches:

     for match in matches:

         print((match.start(),match.start()+len(k)-1))

else:print((-1, -1))

### [re.findall()](https://docs.python.org/2/library/re.html#re.findall): The expression re.findall() returns all the non-overlapping matches of patterns in a string as a list of strings. **Code:** >>> import re

>>> re.findall(r'\w','http://www.hackerrank.com/')

['h', 't', 't', 'p', 'w', 'w', 'w', 'h', 'a', 'c', 'k', 'e', 'r', 'r', 'a', 'n', 'k', 'c', 'o', 'm']

### [re.finditer()](https://docs.python.org/2/library/re.html#re.finditer): The expression re.finditer() returns an iterator yielding MatchObject instances over all non-overlapping matches for the re pattern in the string. **Code:** >>> import re

>>> re.finditer(r'\w','http://www.hackerrank.com/')

<callable-iterator object at 0x0266C790>

>>> map(lambda x: x.group(),re.finditer(r'\w','http://www.hackerrank.com/'))

['h', 't', 't', 'p', 'w', 'w', 'w', 'h', 'a', 'c', 'k', 'e', 'r', 'r', 'a', 'n', 'k', 'c', 'o', 'm']

**Task:** You are given a string S. It consists of alphanumeric characters, spaces and symbols(+,-).  
Your task is to find all the substrings of S that contains 2 or more vowels.  
Also, these substrings must lie in between 2 consonants and should contain vowels only.

**Note : Vowels are defined as:**AEIOU**and**aeiou**.  
Consonants are defined as:**QWRTYPSDFGHJKLZXCVBNM**and**qwrtypsdfghjklzxcvbnm.

**Input Format:** A single line of input containing string S.

**Output Format:** Print the matched substrings in their order of occurrence on separate lines.  
If no match is found, print -1.

**Sample Input:** rabcdeefgyYhFjkIoomnpOeorteeeeet

**Sample Output:** ee

Ioo

Oeo

eeeee

Ans: import re

vowels = 'aeiou'

consonants = 'qwrtypsdfghjklzxcvbnm' match = re.findall(r'(?<=[' + consonants + '])([' + vowels + ']{2,})(?=[' + consonants + '])',input(),flags=re.I)

print('\n'.join(match)) if match else print(-1)

* You are given a positive integer N.  
  Your task is to print a palindromic triangle of size N.

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.

**Output Format:** Print the palindromic triangle of size  as explained above.

**Sample Input:** 5

**Sample Output:** 1

121

12321

1234321

123454321

Ans:- for i in range(1,int(input())+1):

     print(((10\*\*i) // 9) \*\* 2)

* **Function Description:** Complete the *merge\_the\_tools* function in the editor below.

*merge\_the\_tools* has the following parameters:

* *string s:* the string to analyze
* *int k:* the size of substrings to analyze

**Prints:** Print each subsequence on a new line. There will be n/k of them. No return value is expected.

**Input Format:** The first line contains a single string, s .  
The second line contains an integer, k, the length of each substring.

**Sample Input:** STDIN Function

----- --------

AABCAAADA s = 'AABCAAADA'

3 k = 3

**Sample Output:** AB

CA

AD

Ans: def merge\_the\_tools(string, k):

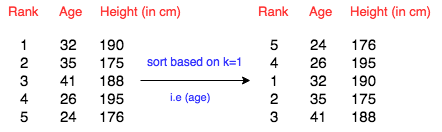
     for i in range(0, len(string), k):

         print(\*{c:1 for c in string[i:i+k]}, sep="")

   string, k = input(), int(input())

   merge\_the\_tools(string, k)

* 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 Kth attribute and print the final resulting table. Follow the example given below for better understanding.



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.

**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:** 5 3

10 2 5

7 1 0

9 9 9

1 23 12

6 5 9

1

**Sample Output:** 7 1 0

10 2 5

6 5 9

9 9 9

1 23 12

Ans: n,m=map(int, input().split())

data=[list(map(int, input().split())) for \_ in range(n)]

k=int(input())

sorted\_data=sorted(data, key=lambda x: x[k])

for i in sorted\_data:

     print(\*i)

* You are given a string S.  
  Your task is to find out whether S is a valid [regex](https://en.wikipedia.org/wiki/Regular_expression) or not.

**Input Format:** The first line contains integer T, the number of test cases.  
The next T lines contains the string S.

**Output Format:** Print "True" or "False" for each test case without quotes.

**Sample Input:** 2

.\*\+

.\*+

**Sample Output:** True

False

Ans: import re

for \_ in range(int(input())):

     try:re.compile(input());print("True")

     except re.error:print("False")

* Python has built-in string validation methods for basic data. It can check if a string is composed of alphabetical characters, alphanumeric characters, digits, etc.

[str.isalnum()](https://docs.python.org/2/library/stdtypes.html#str.isalnum)**:** This method checks if all the characters of a string are alphanumeric (a-z, A-Z and 0-9).

>>> print 'ab123'.isalnum()

True

[str.isalpha()](https://docs.python.org/2/library/stdtypes.html#str.isalpha)**:** This method checks if all the characters of a string are alphabetical (a-z and A-Z).

>>> print 'abcD'.isalpha()

True

[str.isdigit()](https://docs.python.org/2/library/stdtypes.html#str.isdigit)**:** This method checks if all the characters of a string are digits (0-9).

>>> print '1234'.isdigit()

True

[str.islower()](https://docs.python.org/2/library/stdtypes.html#str.islower)**:** This method checks if all the characters of a string are lowercase characters (a-z).

>>> print 'abcd123#'.islower()

True

>>> print 'Abcd123#'.islower()

False

[str.isupper()](https://docs.python.org/2/library/stdtypes.html#str.isupper)**:** This method checks if all the characters of a string are uppercase characters (A-Z).

>>> print 'ABCD123#'.isupper()

True

>>> print 'Abcd123#'.isupper()

False

**Task:** You are given a string S.  
Your task is to find out if the string S contains: alphanumeric characters, alphabetical characters, digits, lowercase and uppercase characters.

**Input Format:** A single line containing a string S.

**Output Format:** In the first line, print True if S has any alphanumeric characters. Otherwise, print False.  
In the second line, print True if S has any alphabetical characters. Otherwise, print False.  
In the third line, print True if S has any digits. Otherwise, print False.  
In the fourth line, print True if S has any lowercase characters. Otherwise, print False.  
In the fifth line, print True if S has any uppercase characters. Otherwise, print False.

**Sample Input:** qA2

**Sample Output:** True

True

True

True

True

Ans: s = input() print(any(char.isalnum() for char in s),any(char.isalpha() for char in s),any(char.isdigit() for char in s),any(char.islower() for char in s),any(char.isupper() for char in s),sep="\n")

* Mr. Anant Asankhya is the manager at the INFINITE hotel. The hotel has an infinite amount of rooms.

One fine day, a finite number of tourists come to stay at the hotel.  
The tourists consist of:  
→ A Captain.  
→ An unknown group of families consisting of K members per group where K ≠ 1.

The Captain was given a separate room, and the rest were given one room per group.

Mr. Anant has an unordered list of randomly arranged room entries. The list consists of the room numbers for all of the tourists. The room numbers will appear K times per group except for the Captain's room.

Mr. Anant needs you to help him find the Captain's room number.  
The total number of tourists or the total number of groups of families is not known to you.  
You only know the value of K and the room number list.

**Input Format:** The first line consists of an integer,K , the size of each group.  
The second line contains the unordered elements of the room number list.

**Output Format:** Output the Captain's room number.

**Sample Input:** 5

1 2 3 6 5 4 4 2 5 3 6 1 6 5 3 2 4 1 2 5 1 4 3 6 8 4 3 1 5 6 2

**Sample Output:** 8

Ans: from collections import Counter

K=int(input())

c=Counter(input().split())

print(\*[key for key,value in c.items() if value==1])

* [collections.Counter()](https://docs.python.org/2/library/collections.html#collections.Counter)**:** A counter is a container that stores elements as dictionary keys, and their counts are stored as dictionary values.

**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 xi 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 xi, the price of the shoe.

**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

Ans: from collections import Counter

x=int(input())

shoe\_sizes=Counter(map(int, input().split()))

N=int(input())

total\_earnings=0

for \_ in range(N):

     size,price=map(int, input().split())

     if shoe\_sizes[size]>0:

         total\_earnings+=price

         shoe\_sizes[size]-=1

print(total\_earnings)

* In this challenge, you will be given 2 integers, n and m. There are n words, which might repeat, in word group A. There are m words belonging to word group B. For each m words, check whether the word has appeared in group A or not. Print the indices of each occurrence of m in group A. If it does not appear, print -1.

**Input Format:** The first line contains integers n,  and m separated by a space.  
The next n lines contains the words belonging to group A.  
The next m lines contains the words belonging to group B.

**Output Format:** Output m lines.  
The ith line should contain the 1-indexed positions of the occurrences of the ith word separated by spaces.

**Sample Input:** STDIN Function

----- --------

5 2 group A size n = 5, group B size m = 2

a group A contains 'a', 'a', 'b', 'a', 'b'

a

b

a

b

a group B contains 'a', 'b'

b

**Sample Output:** 1 2 4

3 5

Ans: n,m=map(int,input().split())

A,B=[input() for \_ in range(n)],[input() for \_ in range(m)]

for i in B:

     print(-1) if i not in A else print(\*[p+1 for p,v in enumerate(A) if i==v])

### [collections.deque()](https://docs.python.org/2/library/collections.html#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**](https://docs.python.org/2/library/collections.html#deque-objects).  
Click on the link to learn more about various approaches to working with deques: [**Deque Recipes**](https://docs.python.org/2.7/library/collections.html#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.

**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

Ans: from collections import deque

d=deque()

for \_ in range(int(input())):

     command=input().split()

     if command[0]=='append':d.append(int(command[1]))

     elif command[0]=='pop':d.pop()

     elif command[0]=='popleft':d.popleft()

     elif command[0]=='appendleft':d.appendleft(int(command[1]))

print(\*d)

### [collections.namedtuple()](https://docs.python.org/2/library/collections.html#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.

**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.

**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

Ans: N,l=int(input()),[]

nc=[p for p,v in enumerate(input().split()) if v=="MARKS"][0]

for i in range(N): l.append(input().split())

print(round((sum(int(i[nc]) for i in l)/N),2))

* **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.

**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

Ans: d={}

for \_ in range(int(input())):

     item\_n,sp,item\_p=input().rpartition(" ")

     d[item\_n]=int(item\_p) if item\_n not in d else int(d[item\_n])+int(item\_p)

for i in d:print(f"{i} {d[i]}")

### [group()](https://docs.python.org/2/library/re.html#re.MatchObject.group): A group() expression returns one or more subgroups of the match. **Code:** >>> import re

>>> m = re.match(r'(\w+)@(\w+)\.(\w+)','username@hackerrank.com')

>>> m.group(0) # The entire match

'username@hackerrank.com'

>>> m.group(1) # The first parenthesized subgroup.

'username'

>>> m.group(1,2,3) # Multiple arguments give us a tuple.

('username', 'hackerrank', 'com')

### [groups()](https://docs.python.org/2/library/re.html#re.MatchObject.groups): A groups() expression returns a tuple containing all the subgroups of the match. **Code:** >>> import re

>>> m = re.match(r'(\w+)@(\w+)\.(\w+)','username@hackerrank.com')

>>> m.groups()

('username', 'hackerrank', 'com')

### [groupdict()](https://docs.python.org/2/library/re.html#re.MatchObject.groupdict): A groupdict() expression returns a dictionary containing all the named subgroups of the match, keyed by the subgroup name. **Code:** >>> m = re.match(r'(?P<user>\w+)@(?P<website>\w+)\.(?P<extension>\w+)','myname@hackerrank.com')

>>> m.groupdict()

{'website': 'hackerrank', 'user': 'myname', 'extension': 'com'}

**Task:** You are given a string S.  
Your task is to find the first occurrence of an alphanumeric character in  S(read from left to right) that has consecutive repetitions.

**Input Format:** A single line of input containing the string S.

**Output Format:** Print the first occurrence of the repeating character. If there are no repeating characters, print -1.

**Sample Input:** ..12345678910111213141516171820212223

**Sample Output:** 1

**Explanation:** .. is the first repeating character, but it is not alphanumeric.  
1 is the first (from left to right) alphanumeric repeating character of the string in the substring 111.

Change Theme

Ans: import re

m=re.search(r'([a-zA-Z0-9])\1',input())

print(m.group(1) if m else -1)

* [itertools.product()](https://docs.python.org/2/library/itertools.html#itertools.product)**:** This tool computes the [cartesian product](https://en.wikipedia.org/wiki/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 A[X](https://en.wikipedia.org/wiki/Cartesian_product)B.

**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.

**Output Format:** Output the space separated tuples of the cartesian product.

**Sample Input:**  1 2

3 4

**Sample Output:**  (1, 3) (1, 4) (2, 3) (2, 4)

Ans: from itertools import product A,B=list(map(int,input().split())),list(map(int,input().split()))

print(\*list(product(A,B)))

* [itertools.permutations(iterable[, r])](https://docs.python.org/2/library/itertools.html#itertools.permutations)**:** 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.

**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

Ans: from itertools import permutations

S,k=input().split() print(\*[''.join(p) for p in permutations(sorted(S),int(k))],sep="\n")

* [itertools.combinations(iterable, r)](https://docs.python.org/2/library/itertools.html#itertools.combinations)  
  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.

**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

Ans: from itertools import combinations

S,k=input().split()

print(\*[''.join(p) for i in range(1,int(k)+1) for p in combinations(sorted(S),i)],sep="\n")

* [itertools.combinations\_with\_replacement(iterable, r)](https://docs.python.org/2/library/itertools.html#itertools.combinations_with_replacement)**:**This tool returns r length subsequences of elements from the input iterable allowing individual elements to be repeated more than once.

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\_with\_replacement

>>> print list(combinations\_with\_replacement('12345',2))

[('1', '1'), ('1', '2'), ('1', '3'), ('1', '4'), ('1', '5'), ('2', '2'), ('2', '3'), ('2', '4'), ('2', '5'), ('3', '3'), ('3', '4'), ('3', '5'), ('4', '4'), ('4', '5'), ('5', '5')]

>>> A = [1,1,3,3,3]

>>> print list(combinations(A,2))

[(1, 1), (1, 3), (1, 3), (1, 3), (1, 3), (1, 3), (1, 3), (3, 3), (3, 3), (3, 3)]

**Task:** You are given a string S.  
Your task is to print all possible size k replacement combinations of the string in lexicographic sorted order.

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

**Output Format:** Print the combinations with their replacements of string S on separate lines.

**Sample Input:** HACK 2

**Sample Output:** AA

AC

AH

AK

CC

CH

CK

HH

HK

KK

Ans: from itertools import combinations\_with\_replacement

S,k=input().split()

print(\*[''.join(p) for p in combinations\_with\_replacement(sorted(S),int(k))],sep="\n")

* itertools module: To read more about the functions in this module, check out their [documentation here](https://docs.python.org/2/library/itertools.html).

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.

**Sample Input:** 4

a a c d

2

**Sample Output:** 0.8333

Ans: from itertools import combinations

N=int(input())

comb=list(combinations(input().split(),int(input())))

print(f"{len([1 for c in comb if 'a' in c])/len(comb):.3f}")

* You are given a function f(X)=X2. You are also given K lists. The ith list consists of Ni elements.

You have to pick one element from each list so that the value from the equation below is maximized:

S=(f(X1)+ f(X2)+…+ f(Xk))%M

Xi denotes the element picked from the ith list . Find the maximized value Smax obtained.

% denotes the modulo operator.

Note that you need to take exactly one element from each list, not necessarily the largest element. You add the squares of the chosen elements and perform the modulo operation. The maximum value that you can obtain, will be the answer to the problem.

**Input Format:** The first line contains 2 space separated integers  K and M.  
The next K lines each contains an integer Ni, denoting the number of elements in the ith list, followed by Ni space separated integers denoting the elements in the list.

**Output Format:** Output a single integer denoting the value Smax.

**Sample Input:** 3 1000

2 5 4

3 7 8 9

5 5 7 8 9 10

**Sample Output:** 206

**Explanation:** Picking 5 from the 1st list, 9 from the 2nd list and 10 from the 3rd list gives the maximum S value equal to (52+92+102)%1000 =206 .

Ans: from itertools import product

K,M=map(int,input().split())

l=[]

for \_ in range(K):

     l.append(list(map(int,input().split()[1:])))

print(max([sum(i\*\*2 for i in combination)%M for combination in product(\*l)]))

### [Calendar Module](https://docs.python.org/2/library/calendar.html#module-calendar)**:** The calendar module allows you to output calendars and provides additional useful functions for them.

[class calendar.TextCalendar([firstweekday])](https://docs.python.org/2/library/calendar.html#calendar.TextCalendar)**:** 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 3 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](https://docs.python.org/2/library/calendar.html#calendar.setfirstweekday).

**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.

**Output Format:** Output the correct day in capital letters.

**Sample Input:** 08 05 2015

**Sample Output:** WEDNESDAY

Ans: import calendar

month,day,year=[int(i) for i in input().split()]

day\_num=calendar.weekday(year,month,day)

print(calendar.day\_name[day\_num].upper())

* CSS colors are defined using a hexadecimal (HEX) notation for the combination of Red, Green, and Blue color values (RGB).

Specifications of HEX Color Code:

■ It must start with a '#' symbol.  
■ It can have 3 or 6 digits.  
■ Each digit is in the range of 0 to F. (0-9 and A-F).  
■  A-F letters can be lower case. (a-f are also valid digits).

**Examples:** Valid Hex Color Codes: #FFF

#025

#F0A1FB

Invalid Hex Color Codes: #fffabg

#abcf

#12365erff

You are given N lines of CSS code. Your task is to print all valid Hex Color Codes, in order of their occurrence from top to bottom.

CSS Code Pattern: Selector

{

Property: Value;

}

**Input Format:** The first line contains N, the number of code lines.  
The next N lines contains CSS Codes.

**Output Format:** Output the color codes with '#' symbols on separate lines.

**Sample Input:** 11

#BED

{

color: #FfFdF8; background-color:#aef;

font-size: 123px;

background: -webkit-linear-gradient(top, #f9f9f9, #fff);

}

#Cab

{

background-color: #ABC;

border: 2px dashed #fff;

}

**Sample Output:** #FfFdF8

#aef

#f9f9f9

#fff

#ABC

#fff

Ans: import re

for \_ in range(int(input())):

     s=re.findall(r'(?<!^)(#(?:[\da-fA-F]{3}){1,2})',input())

     if len(s)!=0:print(\*s,sep="\n")

* You are given an integer, N. Your task is to print an alphabet rangoli of size N. (Rangoli is a form of Indian folk art based on creation of patterns.)

The center of the rangoli has the first alphabet letter *a*, and the boundary has the Nth alphabet letter (in alphabetical order).

**Function Description:** Complete the *rangoli* function in the editor below.

*rangoli* has the following parameters:*int size:* the size of the rangoli

**Returns:** *string:* a single string made up of each of the lines of the rangoli separated by a newline character (\n)

**Input Format:** Only one line of input containing size, the size of the rangoli.

**Sample Input**: 5

**Sample Output:** --------e--------

------e-d-e------

----e-d-c-d-e----

--e-d-c-b-c-d-e--

e-d-c-b-a-b-c-d-e

--e-d-c-b-c-d-e--

----e-d-c-d-e----

------e-d-e------

--------e--------

Ans: def print\_rangoli(size):

     for i in range(size):

         row="-".join(chr(97+size-1-j) for j in range(i+1))

         print((row+row[-2::-1]).center(4\*size-3,"-"))

     for i in range(size-2,-1,-1):

         row="-".join(chr(97+size-1-j) for j in range(i+1))

         print((row+row[-2::-1]).center(4\*size-3,"-"))

   n = int(input())

   print\_rangoli(n)

* When users post an update on social media,such as a URL, image, status update etc., other users in their network are able to view this new post on their news feed. Users can also see exactly when the post was published, i.e, how many hours, minutes or seconds ago.

Since sometimes posts are published and viewed in different time zones, this can be confusing. You are given two timestamps of one such post that a user can see on his newsfeed in the following format:Day dd Mon yyyy hh:mm:ss +xxxx

Here +xxxx represents the time zone. Your task is to print the absolute difference (in seconds) between them.

**Input Format:** The first line contains T, the number of testcases.  
Each testcase contains 2 lines, representing time t1 and time t2.

**Output Format:** Print the absolute difference (t1-t2) in seconds.

**Sample Input:** 2

Sun 10 May 2015 13:54:36 -0700

Sun 10 May 2015 13:54:36 -0000

Sat 02 May 2015 19:54:36 +0530

Fri 01 May 2015 13:54:36 -0000

**Sample Output:** 25200

88200

Ans: from datetime import datetime t\_delta=lambda t1,t2:int(abs(datetime.strptime(t1,"%a %d %b %Y %H:%M:%S %z")-datetime.strptime(t2,"%a %d %b %Y %H:%M:%S %z")).total\_seconds())

print(\*[t\_delta(input(),input()) for \_ in range(int(input()))],sep="\n")

* A white triangle with black background

  Description automatically generatedABC  is a right triangle, 90° at B.  
  Therefore, <ABC=90°.

Point M is the midpoint of hypotenuse AC.

You are given the lengths  AB and BC.  
Your task is to find <MBC(angle O°, as shown in the figure) in degrees.

**Input Format:** The first line contains the length of side AB.  
The second line contains the length of side BC.

**Output Format**: Output <MBC in degrees.

**Note:** Round the angle to the nearest integer.

**Sample Input:** 10

10

**Sample Output:** 45°

Ans: import math

angle=round(math.degrees(math.atan2(float(input()),float(input()))))

print(angle,"\u00B0",sep="")

* 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.

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©
* 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](http://www.diveintopython3.net/iterators.html#defining-classes).  
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](http://docs.python.org/3.2/reference/datamodel.html).

Ans:

import math

class Complex:

    def \_\_init\_\_(self,real,imag):

        self.real=real

        self.imag=imag

    \_\_add\_\_=lambda self,no:Complex(self.real+no.real,self.imag+no.imag)

    \_\_sub\_\_=lambda self,no:Complex(self.real-no.real,self.imag -no.imag)

    def \_\_mul\_\_(self,no):

        prod=complex(self.real,self.imag)\*complex(no.real,no.imag)

        return Complex(prod.real,prod.imag)

    def \_\_truediv\_\_(self,no):

        div=complex(self.real,self.imag)/complex(no.real,no.imag)

        return Complex(div.real,div.imag)

    mod=lambda self:Complex(math.sqrt(self.real\*\*2 + self.imag\*\*2),0)

    def \_\_str\_\_(self):

        if self.imag==0:result="%.2f+0.00i"%(self.real)

        elif self.real==0:

            if self.imag>=0:result="0.00+%.2fi"%(self.imag)

            else:result="0.00-%.2fi"%(abs(self.imag))

        elif self.imag>0:result="%.2f+%.2fi"%(self.real,self.imag)

        else:result="%.2f-%.2fi"%(self.real, abs(self.imag))

        return result

if \_\_name\_\_ == '\_\_main\_\_':

    c = map(float, input().split())

    d = map(float, input().split())

    x = Complex(\*c)

    y = Complex(\*d)

    print(\*map(str, [x+y, x-y, x\*y, x/y, x.mod(), y.mod()]), sep='\n')

* 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 P H I.

Cos(P H I)=( X.Y)/| X || Y | where  X=AB\* BC and  Y=BC\*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:** 0 4 5

1 7 6

0 5 9

1 7 2

**Sample Output:** 8.19

Ans:

import math

class Points:

    def \_\_init\_\_(self,x,y,z):

        self.x=x

        self.y=y

        self.z=z

    \_\_sub\_\_=lambda self,no:Points((self.x-no.x),(self.y-no.y),(self.z-no.z))

    dot=lambda self,no:(self.x\*no.x)+(self.y\*no.y)+(self.z\*no.z)

    cross=lambda self,no:Points((self.y\*no.z-self.z\*no.y),(self.z\*no.x-self.x\*no.z),(self.x\*no.y-self.y\*no.x))

    absolute=lambda self:pow((self.x\*\*2+self.y\*\*2+self.z\*\*2),0.5)

if \_\_name\_\_ == '\_\_main\_\_':

    points = list()

    for i in range(4):

        a = list(map(float, 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))

* There is a horizontal row of n cubes. The length of each cube is given. You need to create a new vertical pile of cubes. The new pile should follow these directions: if cube[i] is on top of cube[j] then sideLength[j]>=sideLength[i].

When stacking the cubes, you can only pick up either the leftmost or the rightmost cube each time. Print Yes if it is possible to stack the cubes. Otherwise, print No.

**Example**: blocks=[1,2,3,7,8]

Result: No

After choosing the rightmost element, 7, choose the leftmost element,1 . After than, the choices are 2 and 8. These are both larger than the top block of size 1.

blocks=[1,2,3,7,8]

Result: Yes

Choose blocks from right to left in order to successfully stack the blocks.

**Input Format:** The first line contains a single integer T, the number of test cases.  
For each test case, there are 2 lines.  
The first line of each test case contains n, the number of cubes.  
The second line contains n space separated integers, denoting the sideLengths of each cube in that order.

**Output Format:** For each test case, output a single line containing either Yes or No.

**Sample Input:** STDIN Function

----- --------

2 T = 2

6 blocks[] size n = 6

4 3 2 1 3 4 blocks = [4, 3, 2, 1, 3, 4]

3 blocks[] size n = 3

1 3 2 blocks = [1, 3, 2]

**Sample Output:** Yes

No

Ans:

from collections import deque

def check(d):

    while d:

        big=d.popleft() if d[0]>d[-1] else d.pop()

        if not d:return "Yes"

        if d[-1]>big or d[0]>big:return "No"

for \_ in range(int(input())):

    \_,d=input(),deque(map(int,input().split()))

    print(check(d))

* There is an array of n integers. There are also 2 **disjoint sets**, A and B, each containing m integers. You like all the integers in set A and dislike all the integers in set B. Your initial happiness is 0. For each  integer i in the array, if iEA, you add 1 to your happiness. If iEB, you add -1 to your happiness. Otherwise, your happiness does not change. Output your final happiness at the end.

**Note:** Since A and B are sets, they have no repeated elements. However, the array might contain duplicate elements.

**Input Format:** The first line contains integers n and m separated by a space.  
The second line contains n integers, the elements of the array.  
The third and fourth lines contain m integers, A and B, respectively.

**Output Format:** Output a single integer, your total happiness.

**Sample Input:** 3 2

1 5 3

3 1

5 7

**Sample Output:** 1

Ans:

n,m=map(int, input().split())

arr=list(map(int, input().split()))

set\_A,set\_B=set(map(int, input().split())),set(map(int, input().split()))

print(sum([(i in set\_A)-(i in set\_B) for i in arr]))

* Kevin and Stuart want to play the '**The Minion Game**'.

**Game Rules:** Both players are given the same string, S.  
Both players have to make substrings using the letters of the string S.  
Stuart has to make words starting with *consonants*.  
Kevin has to make words starting with *vowels*.  
The game ends when both players have made all possible substrings.

**Scoring:** A player gets +1 point for each occurrence of the substring in the string S.

**For Example**: String S = *BANANA*  
Kevin's vowel beginning word = *ANA*  
Here, *ANA* occurs twice in *BANANA*. Hence, Kevin will get 2 Points.  
For better understanding, see the image below:

A screenshot of a game

Description automatically generated

Your task is to determine the winner of the game and their score.

**Function Description:** Complete the *minion\_game* in the editor below.

*minion\_game* has the following parameters:

* *string string:* the string to analyze

**Prints:** *string:* the winner's name and score, separated by a space on one line, or Draw if there is no winner

**Input Format:** A single line of input containing the string S.  
**Note**: The string S will contain only uppercase letters: [A-Z].

**Sample Input:** BANANA

**Sample Output**: Stuart 12

Ans:

def minion\_game(string):

    stuart,kevin=0,0

    for i in range(len(string)):

        if string[i] in {'A','E','I','O','U'}:kevin+=(len(string)-i)

        else:stuart+=(len(string)-i)

    print('Draw') if stuart==kevin else print('Stuart',stuart) if stuart>kevin else print('Kevin',kevin)

if \_\_name\_\_ == '\_\_main\_\_':

    s = input()

    minion\_game(s)

* [HTML](https://www.google.co.in/webhp?sourceid=chrome-instant&ion=1&espv=2&ie=UTF-8#q=What+is+HTML)**:** Hypertext Markup Language is a standard markup language used for creating World Wide Web pages.

[Parsing](https://en.wikipedia.org/wiki/Parsing)**:** Parsing is the process of syntactic analysis of a string of symbols. It involves resolving a string into its component parts and describing their syntactic roles.

[HTMLParser](https://docs.python.org/2/library/htmlparser.html#HTMLParser.HTMLParser)**:** An HTMLParser instance is fed HTML data and calls handler methods when start tags, end tags, text, comments, and other markup elements are encountered.

**Example**: **Code:**from HTMLParser import HTMLParser

# create a subclass and override the handler methods

class MyHTMLParser(HTMLParser):

def handle\_starttag(self, tag, attrs):

print "Found a start tag :", tag

def handle\_endtag(self, tag):

print "Found an end tag :", tag

def handle\_startendtag(self, tag, attrs):

print "Found an empty tag :", tag

# instantiate the parser and fed it some HTML

parser = MyHTMLParser()

parser.feed("<html><head><title>HTML Parser - I</title></head>"

+"<body><h1>HackerRank</h1><br /></body></html>")

**Output:** Found a start tag : html

Found a start tag : head

Found a start tag : title

Found an end tag : title

Found an end tag : head

Found a start tag : body

Found a start tag : h1

Found an end tag : h1

Found an empty tag : br

Found an end tag : body

Found an end tag : html

[.handle\_starttag(tag, attrs)](https://docs.python.org/2/library/htmlparser.html#HTMLParser.HTMLParser.handle_starttag)

This method is called to handle the start tag of an element. (For example: <div class='marks'>)  
The tag argument is the name of the tag converted to lowercase.  
The attrs argument is a list of (name, value) pairs containing the attributes found inside the tag’s <> brackets.

[.handle\_endtag(tag)](https://docs.python.org/2/library/htmlparser.html#HTMLParser.HTMLParser.handle_endtag)

This method is called to handle the end tag of an element. (For example: </div>)  
The tag argument is the name of the tag converted to lowercase.

[.handle\_startendtag(tag,attrs)](https://docs.python.org/2/library/htmlparser.html#HTMLParser.HTMLParser.handle_startendtag)

This method is called to handle the empty tag of an element. (For example: <br />)  
The tag argument is the name of the tag converted to lowercase.  
The attrs argument is a list of (name, value) pairs containing the attributes found inside the tag’s <> brackets.

**Task:** You are given an HTML code snippet of N lines.  
Your task is to print start tags, end tags and empty tags separately.

Format your results in the following way:

Start : Tag1

End : Tag1

Start : Tag2

-> Attribute2[0] > Attribute\_value2[0]

-> Attribute2[1] > Attribute\_value2[1]

-> Attribute2[2] > Attribute\_value2[2]

Start : Tag3

-> Attribute3[0] > None

Empty : Tag4

-> Attribute4[0] > Attribute\_value4[0]

End : Tag3

End : Tag2

Here, the -> symbol indicates that the tag contains an attribute. It is immediately followed by the name of the attribute and the attribute value.  
The > symbol acts as a separator of the attribute and the attribute value.

If an HTML tag has no attribute then simply print the name of the tag.  
If an attribute has no attribute value then simply print the name of the attribute value as None.

**Note**: Do not detect any HTML tag, attribute or attribute value inside the HTML comment tags (<!-- Comments -->).Comments can be multiline as well.

**Input Format:** The first line contains integer N, the number of lines in a HTML code snippet.  
The next N lines contain HTML code.

**Output Format:** Print the HTML tags, attributes and attribute values in order of their occurrence from top to bottom in the given snippet.

Use proper formatting as explained in the problem statement.

**Sample Input:** 2

<html><head><title>HTML Parser - I</title></head>

<body data-modal-target class='1'><h1>HackerRank</h1><br /></body></html>

**Sample Output:** Start : html

Start : head

Start : title

End : title

End : head

Start : body

-> data-modal-target > None

-> class > 1

Start : h1

End : h1

Empty : br

End : body

End : html

Ans:

from html.parser import HTMLParser

class CustomHTMLParser(HTMLParser):

    def handle\_attr(self,attrs):

        for attr\_val\_tuple in attrs:

            print("->",attr\_val\_tuple[0],">",attr\_val\_tuple[1])

    def handle\_starttag(self,tag,attrs):

        print("Start :",tag)

        self.handle\_attr(attrs)

    handle\_endtag=lambda self,tag:print("End   :",tag)

    def handle\_startendtag(self,tag,attrs):

        print("Empty :",tag)

        self.handle\_attr(attrs)

parser = CustomHTMLParser()

parser.feed("".join(input() for \_ in range(int(input()))))

* [*.handle\_comment(data)*](https://docs.python.org/2/library/htmlparser.html#HTMLParser.HTMLParser.handle_comment): This method is called when a comment is encountered (e.g. <!--comment-->).  
  The *data* argument is the content inside the comment tag:

from HTMLParser import HTMLParser

class MyHTMLParser(HTMLParser):

def handle\_comment(self, data):

print "Comment :", data

[*.handle\_data(data)*](https://docs.python.org/2/library/htmlparser.html#HTMLParser.HTMLParser.handle_data): This method is called to process arbitrary data (e.g. text nodes and the content of <script>...</script> and <style>...</style>).  
The *data* argument is the text content of HTML.

from HTMLParser import HTMLParser

class MyHTMLParser(HTMLParser):

def handle\_data(self, data):

print "Data :", data

**Task:** You are given an *HTML* code snippet of N lines.  
Your task is to print the *single-line comments, multi-line comments* and the *data*.

Print the result in the following format:

>>> Single-line Comment

Comment

>>> Data

My Data

>>> Multi-line Comment

Comment\_multiline[0]

Comment\_multiline[1]

>>> Data

My Data

>>> Single-line Comment:

**Note**: Do not print *data* if data == '\n'.

**Input Format:** The first line contains integer N, the number of lines in the *HTML* code snippet.  
The next N lines contains *HTML* code.

**Output Format:** Print the *single-line comments, multi-line comments* and the *data* in order of their occurrence from top to bottom in the snippet.

Format the answers as explained in the problem statement.

**Sample Input:** 4

<!--[if IE 9]>IE9-specific content

<![endif]-->

<div> Welcome to HackerRank</div>

<!--[if IE 9]>IE9-specific content<![endif]-->

**Sample Output**: >>> Multi-line Comment

[if IE 9]>IE9-specific content

<![endif]

>>> Data

Welcome to HackerRank

>>> Single-line Comment

[if IE 9]>IE9-specific content<![endif]

Ans:

from html.parser import HTMLParser

class MyHTMLParser(HTMLParser):

    def handle\_comment(self,data):

        prefix='Multi' if '\n' in data else 'Single'

        print(f">>> {prefix}-line Comment",data,sep="\n")

    def handle\_data(self, data):

        if data!="\n":print(">>> Data",data,sep="\n")

html=""

for i in range(int(input())):

    html+=input()+"\n"

parser=MyHTMLParser()

parser.feed(html)

* You are given an HTML code snippet of N lines.  
  Your task is to detect and print all the HTML tags, attributes and attribute values.

Print the detected items in the following format:

Tag1

Tag2

-> Attribute2[0] > Attribute\_value2[0]

-> Attribute2[1] > Attribute\_value2[1]

-> Attribute2[2] > Attribute\_value2[2]

Tag3

-> Attribute3[0] > Attribute\_value3[0]

The -> symbol indicates that the tag contains an attribute. It is immediately followed by the name of the attribute and the attribute value.  
The > symbol acts as a separator of attributes and attribute values.

If an HTML tag has no attribute then simply print the name of the tag.

**Note:** Do not detect any HTML tag, attribute or attribute value inside the HTML comment tags (<!-- Comments -->). Comments can be multiline.  
All attributes have an attribute value.

**Input Format:** The first line contains an integer N, the number of lines in the HTML code snippet.  
The next N lines contain HTML code.

**Output Format:** Print the HTML tags, attributes and attribute values in order of their occurrence from top to bottom in the snippet.

Format your answers as explained in the problem statement.

**Sample Input:** 9

<head>

<title>HTML</title>

</head>

<object type="application/x-flash"

data="your-file.swf"

width="0" height="0">

<!-- <param name="movie" value="your-file.swf" /> -->

<param name="quality" value="high"/>

</object>

**Sample Output:** head

title

object

-> type > application/x-flash

-> data > your-file.swf

-> width > 0

-> height > 0

param

-> name > quality

-> value > high

Ans:

from html.parser import HTMLParser

class MyHTMLParser(HTMLParser):

    def handle\_starttag(self,tag,attrs):

        print(tag)

        [print(f'-> {attr} > {value}') for attr,value in attrs]

parser=MyHTMLParser()

for \_ in range(int(input())):

    parser.feed(input())

* You are given a valid XML document, and you have to print its score. The score is calculated by the sum of the score of each element. For any element, the score is equal to the number of attributes it has.

**Input Format:** The first line contains N, the number of lines in the XML document.  
The next N lines follow containing the XML document.

**Output Format:** Output a single line, the integer score of the given XML document.

**Sample Input:** 6

<feed xml:lang='en'>

<title>HackerRank</title>

<subtitle lang='en'>Programming challenges</subtitle>

<link rel='alternate' type='text/html' href='http://hackerrank.com/'/>

<updated>2013-12-25T12:00:00</updated>

</feed>

**Sample Output:** 5

**Explanation:** The feed and subtitle tag have one attribute each - lang.  
The title and updated tags have no attributes.  
The link tag has three attributes - rel, type and href.

So, the total score is 1+1+3=5.  
There may be any level of nesting in the XML document. To learn about XML parsing, refer [here](http://www.diveintopython3.net/xml.html).

**NOTE**: In order to parse and generate an XML element tree, use the following code:

>> import xml.etree.ElementTree as etree

>> tree = etree.ElementTree(etree.fromstring(xml))

Here, XML is the variable containing the string.  
Also, to find the number of keys in a dictionary, use the len function:

>> dicti = {'0': 'This is zero', '1': 'This is one'}

>> print (len(dicti))

2

Ans:

import sys

import xml.etree.ElementTree as etree

get\_attr\_number=lambda node:len(node.attrib)+sum(get\_attr\_number(child) for child in node)

if \_\_name\_\_ == '\_\_main\_\_':

    sys.stdin.readline()

    xml = sys.stdin.read()

    tree = etree.ElementTree(etree.fromstring(xml))

    root = tree.getroot()

    print(get\_attr\_number(root))

* You are given a valid XML document, and you have to print the maximum level of nesting in it. Take the depth of the root as 0.

**Input Format:** The first line contains N, the number of lines in the XML document.  
The next N lines follow containing the XML document.

**Output Format:** Output a single line, the integer value of the maximum level of nesting in the XML document.

**Sample Input:** 6

<feed xml:lang='en'>

<title>HackerRank</title>

<subtitle lang='en'>Programming challenges</subtitle>

<link rel='alternate' type='text/html' href='http://hackerrank.com/'/>

<updated>2013-12-25T12:00:00</updated>

</feed>

**Sample Output:** 1

**Explanation:** Here, the root is a feed tag, which has depth of 0.  
The tags title, subtitle, link and updated all have a depth of 1.

Thus, the maximum depth is 1.

Ans:

import xml.etree.ElementTree as etree

maxdepth=0

def depth(elem,level):

    global maxdepth

    if level==maxdepth:maxdepth+=1

    for child in elem:

        depth(child,level+1)

if \_\_name\_\_ == '\_\_main\_\_':

    n = int(input())

    xml = ""

    for i in range(n):

        xml =  xml + input() + "\n"

    tree = etree.ElementTree(etree.fromstring(xml))

    depth(tree.getroot(), -1)

    print(maxdepth)

* Let's dive into decorators! You are given N mobile numbers. Sort them in ascending order then print them in the standard format shown below:+91 xxxxx xxxxx  
  The given mobile numbers may have +91, 91 or 0 written before the actual 10 digit number. Alternatively, there may not be any prefix at all.

**Input Format:** The first line of input contains an integer N, the number of mobile phone numbers.  
N lines follow each containing a mobile number.

**Output Format:** Print N mobile numbers on separate lines in the required format.

**Sample Input:** 3

07895462130

919875641230

9195969878

**Sample Output:** +91 78954 62130

+91 91959 69878

+91 98756 41230

**Concept:** Like most other programming languages, Python has the concept of closures. Extending these closures gives us decorators, which are an invaluable asset. You can learn about decorators in 12 easy steps [here](http://simeonfranklin.com/blog/2012/jul/1/python-decorators-in-12-steps/).  
To solve the above question, make a list of the mobile numbers and pass it to a function that sorts the array in ascending order. Make a decorator that standardizes the mobile numbers and apply it to the function.

Ans:

def wrapper(f):

    def fun(l):

        f(['+91 '+c[-10:-5]+' '+c[-5:] for c in l])

    return fun

@wrapper

def sort\_phone(l):

    print(\*sorted(l), sep='\n')

if \_\_name\_\_ == '\_\_main\_\_':

    l = [input() for \_ in range(int(input()))]

    sort\_phone(l)

* Let's use decorators to build a name directory! You are given some information about N people. Each person has a first name, last name, age and sex. Print their names in a specific format sorted by their age in ascending order i.e. the youngest person's name should be printed first. For two people of the same age, print them in the order of their input.  
  For Henry Davids, the output should be:

Mr. Henry Davids

For Mary George, the output should be:

Ms. Mary George

**Input Format:** The first line contains the integer N, the number of people.  
N lines follow each containing the space separated values of the first name, last name, age and sex, respectively.

**Output Format:** Output N names on separate lines in the format described above in ascending order of age.

**Sample Input:** 3

Mike Thomson 20 M

Robert Bustle 32 M

Andria Bustle 30 F

**Sample Output:** Mr. Mike Thomson

Ms. Andria Bustle

Mr. Robert Bustle

**Concept:** For sorting a nested list based on some parameter, you can use the itemgetter library. You can read more about it [here](http://stackoverflow.com/questions/409370/sorting-and-grouping-nested-lists-in-python?answertab=votes#tab-top).

Ans:

import operator

def person\_lister(f):

    inner=lambda people:map(f,sorted(people,key=lambda x:int(x[2])))

    return inner

@person\_lister

def name\_format(person):

    return ("Mr. " if person[3] == "M" else "Ms. ") + person[0] + " " + person[1]

if \_\_name\_\_ == '\_\_main\_\_':

    people = [input().split() for i in range(int(input()))]

    print(\*name\_format(people), sep='\n')

* Write a function to input variable no of arguments and print its average.

Ans:

def avg(\*args):

l=[arg for arg in args]

return (sum(l)/len(l))

* Write a function to input a string containing lowercase letters and digits and return those digits and letters which are not present in the string. Digits should be before letters sorted in ascending order.

Ans:

def find\_missing\_characters(s):

import string

ad=set(string.digits)

al=set(string.ascii\_lowercase)

s=set(s)

return “”.join(sorted(ad-s)+sorted(al-s))