```
#Say "Hello, World!" With Python

if __name__ == '__main__':
    print("Hello, World!")


#Python If-Else

if __name__ == '__main__':
    n = int(input().strip())
if n % 2 == 1:
    print("Weird")
elif n % 2 == 0 and 2<=n<=5:
    print("Not Weird")
elif n % 2 == 0 and 6<=n<=20:
    print("Weird")
elif n % 2 == 0 and n>20:
    print("Not Weird”)



#Arithmetic Operators

if __name__ == '__main__':
    a = int(input())
    b = int(input())
    print(a+b)
    print(a-b)
    print(a*b)


# Python: Division

if __name__ == '__main__':
    a = int(input())
    b = int(input())
    print(a//b)
    print(a/b)
```

```
#Loops

if __name__ == '__main__':
    n = int(input())
    i = 0
    while i < n:
        print(i**2)
        i += 1


#Write a function

def is_leap(year):
    leap = False

    # Write your logic here
    if year % 4 == 0:
        leap = True
    if year % 100 == 0:
        leap = False
    if year % 400 == 0:
        leap = True

    return leap



#Print Function

if __name__ == '__main__':
    n = int(input())
    for i in range(1,n+1):
        print(i, end = "")


#List Comprehensions

if __name__ == '__main__':
    x = int(input())
    y = int(input())
    z = int(input())
    n = int(input())
res = [[i,j,k] for i in range(x+1) for j in range(y+1) for k in range(z+1) if i

print(res)
```

```
#Find the Runner-Up Score!

if __name__ == '__main__':
    n = int(input())
    arr = map(int, input().split())
    arr_sorted = sorted(set(arr), reverse = True)
    print(arr_sorted[1])


#Nested Lists

if __name__ == '__main__':
    names = []
    for _ in range(int(input())):
        name = input()
        score = float(input())
        names.append([name, score])
    scores = sorted(set([i[1] for i in names]))
    second_lowest_score = scores[1]
    name_snd_lst_score = [i[0] for i in names if i[1] == second_lowest_score]

    name_snd_lst_score.sort() #aplhabetically

    for j in name_snd_lst_score:
        print(j)



#Finding the percentage

if __name__ == '__main__':
    n = int(input())
    student_marks = {}
    for _ in range(n):
        name, *line = input().split()
        scores = list(map(float, line))
        student_marks[name] = scores
    query_name = input()


def average(x):
    return sum(x)/len(x)

voti = student_marks[query_name]
media_voti = average(voti)
```

```
#Lists

if __name__ == '__main__':
    N = int(input())
    lista = []
    for e in range(N):
        command = input().split()
        if command[0] == "insert":
            lista.insert(int(command[1]), int(command[2]))
        elif command[0] == "print":
            print(lista)
        elif command[0] == "remove":
            lista.remove(int(command[1]))
        elif command[0] == "append":
            lista.append(int(command[1]))
        elif command[0] == "sort":
            lista.sort()
        elif command[0] == "pop":
            lista.pop()
        elif command[0] == "reverse":
            lista.reverse()



#Tuples

if __name__ == '__main__':
    n = int(input())
    integer_list = map(int, input().split())

    t = tuple(integer_list)
    print(hash(t))


#sWAP cASE

def swap_case(s):
    new_string = ""
    for i in s:
        if i.islower():
            new_string += i.upper()
        elif i.isupper():
            new_string += i.lower()
        else:
            new_string += i
    return new_string
```

```
#String Split and Join

def split_and_join(line):
    a = line.split(" ")
    return("-".join(a))

if __name__ == '__main__':
    line = input()
    result = split_and_join(line)
    print(result)



#What's Your Name?

# Complete the 'print_full_name' function below.
#
# The function is expected to return a STRING.
# The function accepts following parameters:
#  1. STRING first
#  2. STRING last
#

def print_full_name(first, last):
    print(f"Hello {first} {last}! You just delved into python.")


# Mutations

def mutate_string(string, position, character):
    l = list(string)
    l[position] = character
    string = ''.join(l)
    return string




#Find a string

def count_substring(string, sub_string):
    n = 0
    for i in range(len(string) - len(sub_string) + 1):
        if string[i:i+len(sub_string)] == sub_string:
            n += 1
    return n
```

```
#String Validators

if __name__ == '__main__':
    s = input()
    print(any(i.isalnum() for i in s))
    print(any(i.isalpha() for i in s))
    print(any(i.isdigit() for i in s))
    print(any(i.islower() for i in s))
    print(any(i.isupper() for i in s))


#Text Wrap

def wrap(string, max_width):
    return textwrap.fill(string, max_width)



#String Formatting

def print_formatted(number):
    len_b = len(format(number,'b'))
    for i in range(1, n+1):
        print(str(i).rjust(len_b), format(i,'o').rjust(len_b), format(i,'X').rj



#Capitalize!

# Complete the solve function below.
def solve(s):
    a = s.split(' ')
    s2=[]
    for i in a:
        s2.append(i.capitalize())
    return ' '.join(s2)
```

```
#Merge the Tools!

def merge_the_tools(string, k):
    import textwrap
    lista = textwrap.wrap(string,k)
    for x in lista:
        result =[]
        for i in x:
            if i not in result:
                result.append(i)
        print("".join(result))



#Introduction to Sets

def average(array):
    array = set(array)
    average = sum(array)/len(array)
    return(format(average,'.3f'))


#Symmetric Difference

m = int(input())
a = set(map(int,input().split()[:m]))
n = int(input())
b = set(map(int,input().split()[:n]))
result = a ^ b
for i in sorted(result):
    print(i)
```

```
#No Idea!

n, m  = map(int,input().split())
arr = list(map(int,input().split()[:n]))
a = set(map(int,input().split()[:m]))
b = set(map(int,input().split()[:m]))

happiness = 0
for i in arr:
    if i in a:
        happiness += 1
    elif i in b:
        happiness -= 1

print(happiness)


#Set .add()

n = int(input())
cns = set()
[cns.add(input()) for i in range(n)]
print(len(cns))



#Set .union() Operation

n = int(input())
english = set(map(int, input().split()[:n]))
b = int(input())
french = list(map(int, input().split()[:b]))
count = 0
unione = english.union(french)
count = 0
for i in unione:
    count +=1
print(count)
```

```
#Set .intersection() Operation

n = int(input())
english = set(map(int, input().split()[:n]))
b = int(input())
french = set(map(int, input().split()[:b]))
intersection = english.intersection(french)
count = 0
for i in intersection:
    count += 1

print(count)
```

```
#Set .difference() Operation

n = int(input())
english = set(map(int, input().split()[:n]))
b = int(input())
french = set(map(int, input().split()[:b]))
difference = english - french
count = 0
for i in difference:
    count +=1

print(count)
```

```
#Set .symmetric_difference() Operation

n = int(input())
english = set(map(int, input().split()[:n]))
b = int(input())
french = set(map(int, input().split()[:b]))
sym_diff = english^french
count = 0
for i in sym_diff:
    count +=1

print(count)
```

```
#Set Mutations

n = int(input())
A = set(map(int, input().split()[:n]))
N = int(input())
for i in range(N):
    operation = input().split()
    sets = set(map(int, input().split()))
    if operation[0] == "intersection_update":
        A.intersection_update(sets)
    if operation[0] == "update":
        A.update(sets)
    if operation[0] == "symmetric_difference_update":
        A.symmetric_difference_update(sets)
    if operation[0] == "difference_update":
        A.difference_update(sets)

print(sum(A))



#The Captain's Room

K = int(input())
rooms = list(map(int, input().split()))
rooms_set = set(rooms)
for i in rooms_set:
    rooms.remove(i)
    if i not in rooms:
        print(i)


#Set .discard(), .remove() & .pop()

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()
    if command[0] == "discard":
        s.discard(int(command[1]))
    if command[0] == "remove":
        s.remove(int(command[1]))

print(sum(s))
```

```
#Check Subset

T = int(input())
for cases in range(T):
    n_A = int(input())
    A = set(map(int, input().split()))
    n_B = int(input())
    B = set(map(int, input().split()))

    if A.issubset(B):
        print(True)
    else:
        print(False)
```

```
#Check Strict Superset

A = set(map(int,input().split()))
n = int(input())
result = True
for i in range(n):
    N = set(map(int, input().split()))
    if not A.issuperset(N):
        result = False
        break

print(result)
```

```
#collections.Counter()

from collections import Counter

X = int(input())
sizes = Counter(map(int,input().split()))
N = int(input())
money = 0
for i in range(N):
    size, price = map(int, input().split())
    if sizes.get(size, 0) > 0:
        sizes[size] -= 1
        money += price
        if sizes[size] == 0:
            del sizes[size]
print(money)
```

```
#DefaultDict Tutorial

from collections import defaultdict

n, m = map(int,input().split())
A = [str(input()) for i in range(n)]
B = [str(input()) for j in range(m)]
res = defaultdict(list)

D = list(enumerate(A))

for _ in D:
    res[_[1]].append(_[0]+1)
for x in B:
    if x in A:
        print(*res[x])
    else:
        print(-1)
```

```
#Collections.namedtuple()

from collections import namedtuple
N = int(input())
columns = input().split()
students = namedtuple('students', columns)
marks = []
for i in range(N):
    data = input().split()
    students_2 = students(*data)
    marks.append(int(students_2.MARKS))

avg = sum(marks)/len(marks)
print(f"{avg:.2f}")


#Collections.OrderedDict()

from collections import OrderedDict
N = int(input())
dictionary = OrderedDict()
for i in range(N):
    item, price = input().rsplit(' ',1)
    if item in dictionary:
        dictionary[item] += int(price)
    else:
        dictionary[item] = int(price)

for item, price in dictionary.items():
    print(f"{item} {price}")
```

```
#Text Alignment

#Replace all ______ with rjust, ljust or center.

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(thi
```

```
#Designer Door Mat

n, m = map(int, input().split())
for i in range(1, n, 2):
    pattern = (".|." * i).center(m, "-")
    print(pattern)

print("WELCOME".center(m, "-"))

for i in range(n-2, 0 , -2):
    pattern = (".|." * i).center(m, "-")
    print(pattern)
```

```
#Recursive Digit Sum

#!/bin/python3

import math
import os
import random
import re
import sys

#
# Complete the 'superDigit' function below.
#
# The function is expected to return an INTEGER.
# The function accepts following parameters:
#  1. STRING n
#  2. INTEGER k
#

def superDigit(n, k):
    tot = 0
    p = str(sum(int(i) for i in str(n)))*k
    for _ in p:
        tot += int(_)
    while len(str(tot)) != 1:
        x = 0
        for i in str(tot):
            x +=int(i)
        tot = x
    return x






if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

    first_multiple_input = input().rstrip().split()

    n = first_multiple_input[0]

    k = int(first_multiple_input[1])

    result = superDigit(n, k)

    fptr.write(str(result) + '\n')

    fptr.close()
```

```
#Word Order

from collections import Counter

n = int(input())
words = [input() for i in range(n)]
d_words = Counter(words)
print(len(d_words))
print(*d_words.values())


#Collections.deque()

from collections import deque
N = int(input())
d = deque()
for i in range(N):
    operation = input().split()
    if operation[0] == "append":
        d.append(operation[1])
    if operation[0] == "pop":
        d.pop()
    if operation[0] == "popleft":
        d.popleft()
    if operation[0] == "appendleft":
        d.appendleft(operation[1])

print(*d)
```

```
#Company Logo

#!/bin/python3

import math
import os
import random
import re
import sys



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

from collections import Counter
c = Counter(s).most_common(3)
for key,value in c:
    print(key, value)



#Calendar Module

import calendar
MM, DD, YYYY = map(int, input().split())
print(calendar.day_name[calendar.weekday(YYYY, MM, DD)].upper())


#Exceptions

T = int(input())
for cases in range(T):
    try:
        a, b = map(int, input().split())
        print(a//b)
    except ZeroDivisionError as e:
        print("Error Code:", e)
    except ValueError as i:
        print("Error Code:", i)
```

```
#Zipped!

N, X = map(int, input().split())
elements = [list(map(float, input().split())) for i in range(X)]
for marks in zip(*elements):
    average = sum(marks)/X
    print(average)


#ginortS

S = str(input())
lower = []
upper = []
odd = []
even = []
for i in S:
    if i.isdigit():
        if int(i)%2 == 0:
            even.append(i)
        else:
            odd.append(i)
    elif i.isupper():
        upper += i
    elif i.islower():
        lower += i

lower = sorted(lower)
upper = sorted(upper)
odd = sorted(odd)
even = sorted(even)

final_result = print(''.join(lower+upper+odd+even))
```

```
#Map and Lambda Function

cube = lambda x: x**3

def fibonacci(n):
    l = []
    for i in range(n):
        if i == 0 or i == 1:
            l.append(i)
        else:
            x = l[i-1] + l[i-2]
            l.append(x)
    return l[:n]



#Detect Floating Point Number

import re
T = int(input())
regex = re.compile(r"^[+-]?(0|[1-9]\d*|)\.\d\d*$")
for _ in range(T):
    print(regex.match(input()) != None)



#Re.split()

regex_pattern = r"\.|\,"    # Do not delete 'r'.


#Group(), Groups() & Groupdict()

import re
S = input()
pattern = r"([a-zA-Z0-9])\1+"
x = re.search(pattern,S)
if x:
    print(x.group(1))
else:
    print('-1')
```

```
#Standardize Mobile Number Using Decorators

def wrapper(f):
    def fun(l):
        l = ["+91" + " " + i[-10:-5] + " " + i[-5:] for i in l]
        return f(l)
    return fun



#Arrays

def arrays(arr):
    a = numpy.array(arr, float)
    return a[::-1]



#Shape and Reshape

import numpy
a = input().split()
arr = numpy.array(a, int)
print(numpy.reshape(arr, (3,3)))


#Transpose and Flatten

import numpy
N,M = map(int,input().split())
arr = [input().split() for rows in range(N)]
arr2 = numpy.array(arr, int)
print(numpy.transpose(arr2))
print(arr2.flatten())


#Concatenate

import numpy
N,M,P = map(int, input().split())
arr1 = [input().split() for i in range(N)]
arr2 = [input().split() for j in range(M)]
arr12 = numpy.array(arr1, int)
arr21 = numpy.array(arr2, int)
print(numpy.concatenate((arr12, arr21), axis = 0))
```

```
#Zeros and Ones

import numpy
D = list(map(int, input().split()))


print(numpy.zeros((D), dtype = int))


print(numpy.ones((D), dtype = int))
```

```
#Eye and Identity

import numpy
numpy.set_printoptions(legacy='1.13')
N, M = map(int, input().split())
print(numpy.eye(N, M, k=0))
```

```
#Array Mathematics

import numpy
N, M = map(int, input().split())
a = [input().split() for i in range(N)]
b = [input().split() for j in range(N)]
a1 = numpy.array(a, int)
b1 = numpy.array(b, int)
print(a1+b1)
print(a1-b1)
print(a1*b1)
print(a1//b1)
print(a1%b1)
print(a1**b1)
```

```
#Floor, Ceil and Rint

import numpy
numpy.set_printoptions(legacy='1.13')
A = list(map(float,input().split()))
arr = numpy.array(A)
print(numpy.floor(arr))
print(numpy.ceil(arr))
print(numpy.rint(arr))
```

```
#Sum and Prod

import numpy
N, M = map(int, input().split())
A = [input().split() for i in range(N)]
arr = numpy.array(A, int)
somma = numpy.sum(arr, axis = 0)
print(numpy.prod(somma, axis = 0))


#Min and Max

import numpy
N, M = map(int, input().split())
A = [input().split() for i in range(N)]
arr = numpy.array(A, int)
minimum = numpy.min(arr, axis = 1)
print(numpy.max(minimum))


#Mean, Var, and Std

import numpy
N, M = map(int, input().split())
A = [input().split() for i in range(N)]
arr = numpy.array(A, int)
print(numpy.mean(arr, axis=1))
print(numpy.var(arr, axis=0))
print(round(numpy.std(arr),11))


#Dot and Cross

import numpy
N = int(input())
a = [input().split() for i in range(N)]
b = [input().split() for j in range(N)]
A = numpy.array(a, int)
B = numpy.array(b, int)
print(numpy.dot(A,B))
```

```
#Inner and Outer

import numpy
A = numpy.array(input().split(), int)
B = numpy.array(input().split(), int)
print(numpy.inner(A,B))
print(numpy.outer(A,B))
```

```
#Polynomials

import numpy
P = input().split()
arr = numpy.array(P, float)
x = int(input())
print(numpy.polyval(arr, x))
```

```
#Linear Algebra

import numpy
N = int(input())
A = [input().split() for i in range(N)]
arr = numpy.array(A, float)
print(round(numpy.linalg.det(arr),2))
```

```
#Birthday Cake Candles

#!/bin/python3

import math
import os
import random
import re
import sys

#
# Complete the 'birthdayCakeCandles' function below.
#
# The function is expected to return an INTEGER.
# The function accepts INTEGER_ARRAY candles as parameter.
#

def birthdayCakeCandles(candles):
    return candles.count(max(candles))

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

    candles_count = int(input().strip())

    candles = list(map(int, input().rstrip().split()))

    result = birthdayCakeCandles(candles)

    fptr.write(str(result) + '\n')

    fptr.close()
```

```
#Viral Advertising

#!/bin/python3

import math
import os
import random
import re
import sys

#
# Complete the 'viralAdvertising' function below.
#
# The function is expected to return an INTEGER.
# The function accepts INTEGER n as parameter.
#

def viralAdvertising(n):
    total = 0
    k = 5//2
    for i in range(n):
        total = total + k
        k = (k*3)//2

    return total



if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

    n = int(input().strip())

    result = viralAdvertising(n)

    fptr.write(str(result) + '\n')

    fptr.close()
```

```
#Re.findall() & Re.finditer()

import re
S = input()
p = r'(?<=[bcdfghjklmnpqrstvwxyz])[aeiou]{2,}(?=[bcdfghjklmnpqrstvwxyz])'
m = re.findall(p, S, re.IGNORECASE)
if m:
    print('\n'.join(m))
else:
    print('-1')


#XML 1 - Find the Score

def get_attr_number(node):
    total = 0
    for child in node.iter():
        total += int(len(child.attrib))
    return total



#XML2 - Find the Maximum Depth

maxdepth = 0
def depth(elem, level):
    global maxdepth
    [depth(e,level+1) for e in elem]
    maxdepth = max(maxdepth, level + 1)



#Re.start() & Re.end()

import re
s = input()
k = input()
m = list(re.finditer(f'(?={re.escape(k)})',s))
if not k:
    print((-1,-1))
else:
    if m:
        for i in m:
            print((i.start(),i.start()+len(k)-1))
    else:
        print((-1,-1))
```

```
#Regex Substitution

import re
n = int(input())
for _ in range(n):
    print(re.sub(r"(?<= )\|\|(?= )", "or", re.sub(r"(?<= )&&(?= )", "and", inpu


#Validating Roman Numerals

regex_pattern = r"^(M){0,3}(CM)?(CD|D)?(C){0,3}(XC)?(XL|L)?(X){0,3}(IX)?(IV|V)?
```

```python
#Insertion Sort - Part 1

#!/bin/python3

import math
import os
import random
import re
import sys

#
# Complete the 'insertionSort1' function below.
#
# The function accepts following parameters:
#  1. INTEGER n
#  2. INTEGER_ARRAY arr
#

def insertionSort1(n, arr):
    x = arr[-1]
    i = n - 2
    while i >= 0 and arr[i] > x:
        arr[i + 1] = arr[i]
        print(*arr)
        i -= 1

    arr[i + 1] = x
    print(*arr)

if __name__ == '__main__':
    n = int(input().strip())

    arr = list(map(int, input().rstrip().split()))

    insertionSort1(n, arr)
```

```
#Insertion Sort - Part 2

#!/bin/python3

import math
import os
import random
import re
import sys

#
# Complete the 'insertionSort2' function below.
#
# The function accepts following parameters:
#  1. INTEGER n
#  2. INTEGER_ARRAY arr
#

def insertionSort2(n, arr):
    for i in range(1,n):
        x = arr[i]
        j = i - 1
        while j >= 0 and arr[j] > x:
            arr[j + 1] = arr[j]
            j -= 1

        arr[j + 1] = x
        print(*arr)

if __name__ == '__main__':
    n = int(input().strip())

    arr = list(map(int, input().rstrip().split()))

    insertionSort2(n, arr)


#Number Line Jumps

#!/bin/python3

import math
import os
import random
import re
import sys

#
```

```
# Complete the 'kangaroo' function below.
#
# The function is expected to return a STRING.
# The function accepts following parameters:
#  1. INTEGER x1
#  2. INTEGER v1
#  3. INTEGER x2
#  4. INTEGER v2
#

def kangaroo(x1, v1, x2, v2):
    if v1 == v2:
        if x1 == x2:
            return "YES"
        else:
            return "NO"

    if (x2-x1)%(v1-v2) == 0 and (x2-x1)/(v1-v2) > 0:
        return "YES"
    else:
        return "NO"

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

    first_multiple_input = input().rstrip().split()

    x1 = int(first_multiple_input[0])

    v1 = int(first_multiple_input[1])

    x2 = int(first_multiple_input[2])

    v2 = int(first_multiple_input[3])

    result = kangaroo(x1, v1, x2, v2)

    fptr.write(result + '\n')

    fptr.close()
```

```
#Decorators 2 - Name Directory

from operator import itemgetter
from itertools import groupby

def person_lister(f):
    def inner(people):
        people_sorted = sorted(people, key = lambda x: int(x[2]))
        return [f(person) for person in people_sorted]
    return inner
```