```
#----------------PROBLEM 1----------------------------------

# Say "Hello, World!" with python
print('Hello, World!')

#Python If-Else
import math
import os
import random
import re
import sys
n = int(input())

if n % 2 != 0:
     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
a = int(input())
b = int(input())
print(a+b)
print(a-b)
print(a*b)

#Python Division
a = int(input())
b = int(input())
print(a//b)
print(a/b)

#Loops
n = int(input())
for i in range(n):
    print(i*i)

#Write a function
def is_leap(year):
    if year % 400 == 0:
        return True
    elif year % 100 == 0:
        return False
    elif year % 4 == 0:
        return True
    else:
        return False
    print(is_leap(year))
```

```

#Print Function
n = int(input())
for i in range(1, n+1):
    print(i, end='')

#Find the Runner-Up Score!
n = int(input())
arr = list(map(int, input().split()))
first_max = max(arr)
while first_max in arr:
    arr.remove(first_max)
second_max = max(arr)
print(second_max)

#Nested List
student = []
for i in range(int(input())):
    name = input()
    score = float(input())
    student.append([name, score])
score = sorted(set([score for name, score in student]))
second_lowest_score= score[1]
second_lowest_students= sorted([name for name, score in student if score== second_lowe
for student in second_lowest_students:
    print(student)

#Finding the percentage
n= int(input())
student_marks = {}
for el in range(n):
    data=input().split()
    name = data[0]
    marks = list(map(float, data[1:]))
    student_marks[name] = marks
query_name= input()
query_marks= student_marks [query_name]
mean = sum(query_marks) / len(query_marks)
print(f"{mean:.2f}")

#Lists
if __name__ == '__main__':
    list= []
    for _ in range(int(input())):
        command= input().split()
        if command[0] == "insert":
            list.insert(int(command[1]), int(command[2]))
        elif command[0] == "print":
            print(list)
        elif command[0] == "remove":
            list.remove(int(command[1]))
        elif command[0] == "append":
            list.append(int(command[1]))
```

```python
        elif command[0] == "sort":
            list.sort()
        elif command[0] == "pop":
            list.pop()
        elif command[0] == "reverse":
            list.reverse()

#Tuples
if __name__ == '__main__':
    n = int(input())
    integer_list = map(int, input().split())
    t = tuple(integer_list)
    print(hash(t))

#List Comprehension
if __name__ == '__main__':
    x = int(input())
    y = int(input())
    z = int(input())
    n = int(input())
    r= [[i, j, k] for i in range (x+1)
                  for j in range (y+1)
                  for k in range (z+1)
                  if i+j+k != n]
print(r)

#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?
def print_full_name(first, last):
    print("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):
    count=0
    for i in range(len(string)-len(sub_string) + 1):
        if string[i:i+len(sub_string)] == sub_string:
            count += 1
    return count

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

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

#Shape and Reshape
import numpy as np
my_array = np.array(list(map(int, input().split())))
my_array.shape =  (3, 3)
print(my_array)

#Capitalize!
def solve(s):
    name = s.split(' ')
    new_name= []
    for word in name:
        if len(word)>0:
            new_name.append(word[0].upper()+ word[1:])
        else:
            new_name.append(word)
    return' '.join(new_name)

#Designer Door Mat
N, M = map(int,input().split())
for i in range(1, N, 2):
    pattern = ".|." * i
    print(pattern.center(M, '-'))
print("WELCOME".center(M, '-'))
for i in range(N-2, 0, -2):
    pattern = ".|." * i
    print(pattern.center(M, '-'))

#Symmetric Difference
m = int(input())
my_set = set(map(int, input().split()))
n = int(input())
my_set2 = set(map(int, input().split()))
[print(i) for i in sorted(my_set^my_set2)]

#Set.add()
n= int(input())
country_name = set()
for i in range(n):
    country_name.add(input())
print(len(country_name))

#Introduction to sets
def average(array):
    n = set(array)
    return sum(n) / len(n)

#Set.union() Operation
n = int(input())
set_one = set(map(int, input().split()))
b= int(input())
set_two = set(map(int,input().split()))
union = set_one.union(set_two)
print(len(union))

#Set.intersection() Operation
n = int(input())
set_one = set(map(int,input().split()))
b = int(input())
set_two = set(map(int,input().split()))
inters = set_one.intersection(set_two)
print(len(inters))

#Set.difference() Operation
n = int(input())
set_one = set(map(int, input().split()))
b = int(input())
set_two = set(map(int, input().split()))
diff = set_one.difference(set_two)
print(len(diff))

#Set.symmetric_difference() Operation
n = int(input())
set_one = set(map(int, input().split()))
b = int(input())
set_two = set(map(int, input().split()))
symm = set_one ^ set_two
print(len(symm))

#Set Mutations
n = int(input())
a = set(map(int, input().split()))
num = int(input())
for i in range(num):
    m, _ = input().split()
    new_set = set(map(int, input().split()))
    if m == "update":
        a.update(new_set)
    elif m == "intersection_update":
        a.intersection_update(new_set)
    elif m == "difference_update":
        a.difference_update(new_set)
    elif m == "symmetric_difference_update":
        a.symmetric_difference_update(new_set)
print(sum(a))

#The Captain's Room
k = int(input())
room = list(map(int,input().split()))
a = set(room)
cap= (sum(a) * k - sum(room)) // (k-1)
print(cap)

#Check Subset
set = int(input())
for el in range(test):
    set_a= int(input())
    a = set(map(int, input().split()))
    set_b = int(input())
    b = set(map(int, input().split()))

    if a <= b:
        print(True)
    else:
        print(False)

#Check Strict Superset
a = set(map(int, input().split()))
n = int(input())

for el in range(n):
    b = set(map(int, input().split()))
    if not a.issuperset(b) or a == b:
        print(False)
        break
else:
    print(True)

#No Idea!
n, m = map(int, input().split())
arr = list(map(int, input().split()))
a= set(map(int, input().split()))
b= set(map(int, input().split()))
happiness = 0
for i in arr:
    if i in a:
        happiness += 1
    elif i in b:
        happiness -= 1
print(happiness)

#The Minion Game
def minion_game(string):
    stuart = 0
    kevin = 0
    v = "AEIOU"
    n = len(string)
    for i in range(n):
        if string[i] in v:
            kevin += n - i
        else:
            stuart += n - i
    if stuart > kevin:
        print(f"Stuart {stuart}")
    elif kevin > stuart:
        print(f"Kevin {kevin}")
    else:
        print("Draw")

#Calendar Module
import calendar
month, day, year = map(int, input().split())
week_day = calendar.weekday(year, month, day)
days= ["MONDAY", "TUESDAY", "WEDNESDAY", "THURSDAY", "FRIDAY", "SATURDAY", "SUNDAY"]

print(days[week_day])

#Time Delta
import math
import os
import random
import re
import sys
from datetime import datetime

def time_delta(t1, t2):
    fmt = '%a %d %b %Y %H:%M:%S %z'
    dt1 = datetime.strptime(t1, fmt)
    dt2 = datetime.strptime(t2, fmt)
    delta_seconds = abs(int((dt1 - dt2).total_seconds()))
    return str(delta_seconds)

#sWAP cASE
def swap_case(s):
    ml= list(s)
    for i in range(len(ml)):
        if ml[i].islower():
            ml[i]= ml[i].upper()
        elif ml[i].isupper():
            ml[i]=ml[i].lower()
    return "".join(ml)

#String Formatting
def print_formatted(number):
    x= format(number, "b")
    l= len(x)

    for i in range(1, number+1):
        octal = format(i,"o")
        octal = format(i,"o")
        hexa = format(i,"X")
        bina = format(i,"b")
        print(str(i).rjust(l), str(octal).rjust(l), str(hexa).rjust(l),str(bina).rjus

#Set.discard().remove() &.pop()
n = int(input())
s = set(map(int, input().split()))
a = int(input())
for i in range(a):
    command = input().split()
    if command[0] == "pop":
        s.pop()
    elif command[0] == "discard":
        s.discard(int(command[1]))
    elif command[0] == "remove":
        s.remove(int(command[1]))

print(sum(s))

#Exceptions
n = int(input())
s = set(map(int, input().split()))
a = int(input())
for i in range(a):
    command = input().split()
    if command[0] == "pop":
        s.pop()
    elif command[0] == "discard":
        s.discard(int(command[1]))
    elif command[0] == "remove":
        s.remove(int(command[1]))

print(sum(s))

#Transpose and Flatten
import numpy as np

n, m = map(int, input().split())
arr = np.array([input().split() for i in range(n)], int)
print(np.transpose(arr))
print(arr.flatten())

#Concatenate
import numpy as np

n, m, p = map(int, input().split())
arr_1 = np.array([input().split() for _ in range(n)], int)
arr_2 = np.array([input().split() for _ in range(m)], int)
result = np.concatenate((arr_1, arr_2), axis=0)
print(result)

#Arrays
def arrays(arr):
```

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

#Zeros and Ones
import numpy as np
arr= tuple(map(int, input().split()))
print(np.zeros(arr, dtype=int))
print(np.ones(arr, dtype= int))

#Eye and Identity
import numpy
numpy.set_printoptions(legacy='1.13')
n, m= map(int,input().split())
print(numpy.eye(n,m))

#Sum and Prod
import numpy
n, m= map(int,input().split())
arr = numpy.array([input().split() for _ in range(n)], int)
result_s = numpy.sum(arr, axis=0)
result_p = numpy.prod(result_s)

print(result_p)

#Text Alignment
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

#Array Mathematics
import numpy as np

n, m = map(int, input().split())
a = np.array([list(map(int,input().split())) for i in range(n)], int)
```

```
a = np.array([list(map(int,input().split())) for i in range(n)], int)
b = np.array([list(map(int,input().split())) for i in range(n)], int)
print(a + b)
print(a - b)
print(a * b)
print(np.floor_divide(a, b))
print(a % b)
print(a ** b)

#Floor, Ceil and Rint
import numpy as np
np.set_printoptions(legacy= '1.13')
a = np.array(list(map(float, input().split())))

print(np.floor(a))
print(np.ceil(a))
print(np.rint(a))

#Min and Max
import numpy as np
n,m = map(int, input().split())
my_arr = np.array([list(map(int, input().split())) for i in range(n)])

minimo = np.min(my_arr, axis= 1)
massimo = np.max (minimo)

print(massimo)

#Mean, var and Std
import numpy as np
n, m = map(int, input().split())
my_arr = np.array([(list(map(int, input().split()))) for i in range(n)])
print(np.mean(my_arr, axis=1))
print(np.var (my_arr, axis = 0))
print(np.round(np.std(my_arr, axis=None), 11))

#Dot and Cross
import numpy as np
n =int(input())

a = np.array([list(map(int,input().split())) for i in range(n)])
b = np.array([list(map(int,input().split())) for i in range(n)])

dot= np.dot(a,b)
print(dot)

#Inner and Outer
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 ))
print(np.outer(a ,b ))
```

```

#Polynomials
import numpy as np

c = list(map(float, input().split()))
x = float(input())

value = np.polyval(c, x)
print(value)

#Linear Algebra
import numpy as np
n= int(input())
a= np.array([list(map(float, input().split())) for i in range(n)])
det= np.linalg.det(a)
print(np.round(det ,2))

#Map and Lamba Function
cube = lambda x: x**3

def fibonacci(n):
    if n==0:
        return []
    elif n==1:
        return [0]
    elif n==2:
        return [0,1]
    else:
        result = fibonacci(n-1)
        result.append(result[-1] + result[-2])
    return result

#Zipped!
n, x = map(int, input().split())
my_l = []
for i in range(x):
    my_l.append(list(map(float, input().split())))
for m in zip(*my_l):
   print(sum(m)/len(m))

#Athlete Sort
import math
import os
import random
import re
import sys



if __name__ == '__main__':
    nm = input().split()

    n = int(nm[0])
    m = int(nm[1])

    arr = []

    for _ in range(n):
        arr.append(list(map(int, input().rstrip().split())))

    k = int(input())
    arr.sort(key=lambda x: x[k])
    for row in arr:
        print(*row)

#Collections.counter
from collections import Counter
input()
i = Counter(map(int, input().split()))
c = int(input())
p = 0

for _ in range(c):
    t, z = map(int, input().split())
    if i[t] > 0:
        p += z
        i[t] -= 1

print(p)

#Collections.namedtuple()
from collections import namedtuple
n = int(input())
columns = input().split()
students = namedtuple('students', columns)
tm = []
for i in range(n):
    student = students(*input().split())
    tm.append(int(student.MARKS))

print(round(sum(tm)/len(tm), 2))

#DefaultDict Tutorial
from collections import defaultdict
n, m = map(int, input().split())
A= defaultdict(list)
for i in range(n):
    w = input()
    A[w].append(i+1)
for _ in range(m):
    w= input()
    if w in A:
        print(" ".join(map(str, A[w])))
    else:
        print(-1)
```

```python

#Collections.OrderedDict()
from collections import OrderedDict
dic = OrderedDict()
n = int(input())
for i in range(n):
    *item_name, price = input().split()
    item_name = " ".join(item_name)
    price = int(price)

    if item_name in dic:
        dic[item_name] += price
    else:
        dic[item_name] = price

for item, price in dic.items():
    print(item, price)

#Collections.deque()
from collections import deque
d = deque()
n = int(input())
for i in range(n):
    o = input().split()
    c = o[0]

    if c == "append":
        d.append(int(o[1]))
    if c == "pop":
        d.pop()
    if c == "appendleft":
        d.appendleft(int(o[1]))
    if c== "popleft":
        d.popleft()

print(" ".join(map(str, d)))

#Word Order
n = int(input())
dic = {}

for i in range(n):
    w = input()
    if w in dic:
        dic[w] += 1
    else:
        dic[w] = 1

print(len(dic))
print(*dic.values())

#Company Logo
import math
import os
import random
import re
import sys
from collections import Counter


if __name__ == '__main__':
    s = input().strip()
    counter = Counter(s)
    sorted_counter = sorted(counter.items(), key=lambda x: (-x[1], x[0]))

    for i in range(3):
        print(f"{sorted_counter[i][0]} {sorted_counter[i][1]}")
        # i looked the solutions

#Piling Up!
from collections import deque

for _ in range(int(input())):
    n = int(input())
    blocks = deque(map(int, input().split()))
    possible = True
    last = max(blocks[0], blocks[-1])
    while blocks:
        if blocks[0] >= blocks[-1]:
            current = blocks.popleft()
        else:
            current = blocks.pop()
        if current > last:
            possible = False
            break
        last = current
    print("Yes" if possible else "No")

    #i looked the solutions

#Alphabet Rangoli
def print_rangoli(size):
    import string
    alpha = string.ascii_lowercase
    lines = []
    for i in range(size):
        s = "-".join(alpha[size-1:i:-1] + alpha[i:size])
        lines.append(s.center(4 * size - 3, '-'))
    print('\n'.join(lines[::-1] + lines[1:]))

#Decorators 2- Name directory
def person_lister(f):
    def inner(people):
        return [f(person) for person in sorted(people, key=lambda person: int(person[
    return inner

    # i looked the solutions

#Merge the tools!
def merge_the_tools(string, k):
    for i in range(0, len(string), k):
        substring = string[i:i+k]
        result = ""
        for char in substring:
            if char not in result:
                result += char
        print(result)

#i looked the solutions

#Standardize Mobile Number Using Decorators
def wrapper(f):
    def fun(l):
        l = ['+91 ' + p[-10:-5] + ' ' + p[-5:] for p in [num.strip()[-10:] for num in
        f(l)
    return fun
#i looked the solutions

#Detect Floating point Number
t = int(input())
for i in range(t):
    n = input().strip()
    if n.startswith(('+', '-')) or '.' in n:
        try:
            if float(n):
                print(True)
        except ValueError:
            print(False)
    else:
        print(False)

#ginortS
def order_letters(s):
    lower = []
    upper = []
    odd = []
    even = []

    for i in s:
        if i.islower():
            lower.append(i)
        elif i.isupper():
            upper.append(i)
        elif i.isdigit():
            if int(i) % 2 != 0:
                odd.append(i)
            else:
                even.append(i)

    lower.sort()
    upper.sort()
    odd.sort()
    even.sort()


    result = ''.join(lower + upper + odd + even)


    return result

if __name__ == "__main__":
    s = input()
    print(order_letters(s))


#---------------PROBLEM 2---------------------------------------------------
#Birthday Cake Candles
import math
import os
import random
import re
import sys

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

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()

#Number Line Jump
import math
import os
import random
import re
import sys

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()


print(kangaroo(x1, v1, x2, v2))

#Viral Advertising
import math
import os
import random
import re
import sys

def viralAdvertising(n):
    p = 5
    c = 0
    like_today=0

    for i in range(n):
        like_today = p//2
        c += like_today
        p = like_today*3
    return c

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

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

    result = viralAdvertising(n)

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

    fptr.close()

#Recursive Digit Sum
def superDigit(n, k):
    c = str(sum(int(i) for i in str(n)))*k
    t=0
    a=0
    for i in c:
        t += int(i)
    while len(str(t))>1:
        a=0
        for i in str(t):
            a+= int(i)
        t=a
    return a

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()

#Inserion Sort 1
import math
import os
import random
import re
import sys

def insertionSort1(n, arr):
    e = arr[-1]
    for i in range(n-2, -1, -1):
        if arr[i] > e:
            arr[i+1] = arr[i]
            print(*arr)
        else:
            arr[i+1] = e
            print(*arr)
            break
            break
    else:
        arr[0]= e
        print(*arr)

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

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

    insertionSort1(n, arr)

#Insertion Sort 2
import math
import os
import random
import re
import sys

def insertionSort2(n, arr):
    for i in range(1, n):
        k = arr[i]
        for j in range(i - 1, -2, -1):
            if j >= 0 and arr[j] > k:
                arr[j + 1] = arr[j]
            else:
                arr[j + 1] = k
                break
        print(*arr)

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

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

    insertionSort2(n, arr)
```

```
1008
1009
1010
```

Mostra output nascosti

Nuova sezione