* An *array* is a type of data structure that stores elements of the same type in a contiguous block of memory. In an array, A, of size N, each memory location has some unique index, i (where 0<=i<N), that can be referenced as A[i] or Ai.

Reverse an array of integers.

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

*reverseArray* has the following parameter(s):

* *int A[n]*: the array to reverse

**Returns:** *int[n]*: the reversed array

**Input Format:** The first line contains an integer, N, the number of integers in A.  
The second line contains N space-separated integers that make up A.

**Sample Input:** 4  
1 4 3 2

**Sample Output:**2 3 4 1

Ans:

import os

def reverseArray(a):

    return a[::-1]

if \_\_name\_\_ == '\_\_main\_\_':

    fptr = open(os.environ['OUTPUT\_PATH'], 'w')

    arr\_count = int(input().strip())

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

    res = reverseArray(arr)

    fptr.write(' '.join(map(str, res)))

    fptr.write('\n')

    fptr.close()

* Given a 6\*6 *2D Array*, arr:1 1 1 0 0 0

0 1 0 0 0 0

1 1 1 0 0 0

0 0 0 0 0 0

0 0 0 0 0 0

0 0 0 0 0 0

An hourglass in A is a subset of values with indices falling in this pattern in arr's graphical representation:a b c

d

e f g

There are 16 hourglasses in arr. An *hourglass sum* is the sum of an hourglass' values. Calculate the hourglass sum for every hourglass in arr, then print the *maximum* hourglass sum. The array will always be 6\*6.

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

hourglassSum has the following parameter(s):

* *int arr[6][6]*: an array of integers

**Returns:** *int:* the maximum hourglass sum

**Input Format:** Each of the 6 lines of inputs arr[i] contains 6 space-separated integers arr[i][j].

**Output Format:** Print the largest (maximum) hourglass sum found in arr.

**Sample Input:** 1 1 1 0 0 0

0 1 0 0 0 0

1 1 1 0 0 0

0 0 2 4 4 0

0 0 0 2 0 0

0 0 1 2 4 0

**Sample Output:** 19

Ans:

import os

def hourglassSum(arr):

    return max(sum(arr[i][j:j+3]+[arr[i+1][j+1]]+arr[i+2][j:j+3]) for i in range(4) for j in range(4))

if \_\_name\_\_ == '\_\_main\_\_':

    fptr = open(os.environ['OUTPUT\_PATH'], 'w')

    arr = []

    for \_ in range(6):

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

    result = hourglassSum(arr)

    fptr.write(str(result) + '\n')

    fptr.close()

* A *left rotation* operation on an array of size n shifts each of the array's elements 1 unit to the left. Given an integer, d, rotate the array that many steps left and return the result.

**Example:** d=2

arr=[1,2,3,4,5]

After 2 rotations,arr’=[3,4,5,1,2] .

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

*rotateLeft* has the following parameters:

* *int d:* the amount to rotate by
* *int arr[n]:* the array to rotate

**Returns:** *int[n]:* the rotated array

**Input Format:** The first line contains two space-separated integers that denote n, the number of integers, and ,d the number of left rotations to perform.  
The second line contains n space-separated integers that describe arr[].

**Sample Input:** 5 4

1 2 3 4 5

**Sample Output:** 5 1 2 3 4

Ans: import os

rotateLeft=lambda d,arr:arr[d:]+arr[:d]

if \_\_name\_\_ == '\_\_main\_\_':

    fptr = open(os.environ['OUTPUT\_PATH'], 'w')

    first\_multiple\_input = input().rstrip().split()

    n = int(first\_multiple\_input[0])

    d = int(first\_multiple\_input[1])

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

    result = rotateLeft(d, arr)

    fptr.write(' '.join(map(str, result)))

    fptr.write('\n')

    fptr.close()

* There is a collection of input strings and a collection of query strings. For each query string, determine how many times it occurs in the list of input strings. Return an array of the results.

**Function Description:** Complete the function *matchingStrings* in the editor below. The function must return an array of integers representing the frequency of occurrence of each query string in *stringList*.

matchingStrings has the following parameters:

* *string stringList[n]* - an array of strings to search
* *string queries[q]* - an array of query strings

**Returns:** *int[q]:* an array of results for each query

**Input Format:** The first line contains and integer n, the size of stringList[].  
Each of the next n lines contains a string stringList[i].  
The next line contains q, the size of queries[].  
Each of the next q lines contains a string queries[i].

**Sample Input:** 3  
def  
de  
fgh  
3  
de  
lmn  
fgh

**Sample Output:**1  
0  
1

Ans:

from collections import Counter

import os

def matchingStrings(stringList,queries):

    c=Counter(stringList)

    return [c[q] for q in queries]

if \_\_name\_\_ == '\_\_main\_\_':

    fptr = open(os.environ['OUTPUT\_PATH'], 'w')

    stringList\_count = int(input().strip())

    stringList = []

    for \_ in range(stringList\_count):

        stringList\_item = input()

        stringList.append(stringList\_item)

    queries\_count = int(input().strip())

    queries = []

    for \_ in range(queries\_count):

        queries\_item = input()

        queries.append(queries\_item)

    res = matchingStrings(stringList, queries)

    fptr.write('\n'.join(map(str, res)))

    fptr.write('\n')

    fptr.close()

* Complete the function *solveMeFirst* to compute the sum of two integers.

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

*solveMeFirst* has the following parameters:

* *int a*: the first value
* *int b*: the second value

Returns: - *int*: the sum of a and b

**Sample Input:** a = 2

b = 3

**Sample Output:** 5

Ans:

solveMeFirst=lambda a,b:a+b

print(solveMeFirst(int(input()),int(input())))

* Given an array of integers, find the sum of its elements.

**Function Description:** Complete the *simpleArraySum* function in the editor below. It must return the sum of the array elements as an integer.

simpleArraySum has the following parameter(s):

* *ar*: an array of integers

**Input Format:** The first line contains an integer, n, denoting the size of the array.  
The second line contains n space-separated integers representing the array's elements.

**Output Format:** Print the sum of the array's elements as a single integer.

**Sample Input:** 6

1 2 3 4 10 11

**Sample Output:** 31

Ans:

import os

simpleArraySum=lambda ar:sum(ar)

if \_\_name\_\_ == '\_\_main\_\_':

    fptr = open(os.environ['OUTPUT\_PATH'], 'w')

    ar\_count = int(input().strip())

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

    result = simpleArraySum(ar)

    fptr.write(str(result) + '\n')

    fptr.close()

* Alice and Bob each created one problem for HackerRank. A reviewer rates the two challenges, awarding points on a scale from *1* to *100* for three categories: *problem clarity*, *originality*, and *difficulty*.

The rating for Alice's challenge is the triplet *a = (a[0], a[1], a[2])*, and the rating for Bob's challenge is the triplet *b = (b[0], b[1], b[2])*.

The task is to find their *comparison points* by comparing *a[0]* with *b[0]*, *a[1]* with *b[1]*, and *a[2]* with *b[2]*.

* If *a[i] > b[i]*, then Alice is awarded *1* point.
* If *a[i] < b[i]*, then Bob is awarded *1* point.
* If *a[i] = b[i]*, then neither person receives a point.

Comparison points is the total points a person earned.

Given *a* and *b*, determine their respective comparison points.

**Example:** *a = [1, 2, 3]*  
*b = [3, 2, 1]*

* For elements \*0\*, Bob is awarded a point because *a[0] .*
* *For the equal elements a[1] and b[1], no points are earned.*
* *Finally, for elements 2, a[2] > b[2] so Alice receives a point.*

*The return array is [1, 1] with Alice's score first and Bob's second.*

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

*compareTriplets has the following parameter(s):*

* *int a[3]: Alice's challenge rating*
* *int b[3]: Bob's challenge rating*

***Return:*** *int[2]*: Alice's score is in the first position, and Bob's score is in the second.

**Input Format:** The first line contains *3* space-separated integers, *a[0]*, *a[1]*, and *a[2]*, the respective values in triplet *a*.  
The second line contains *3* space-separated integers, *b[0]*, *b[1]*, and *b[2]*, the respective values in triplet *b*.

**Sample Input:** 17 28 30

99 16 8

**Sample Output:** 2 1

Ans:

import os

compareTriplets=lambda a,b:[sum(i>j for i,j in zip(a,b)),sum(i<j for i,j in zip(a,b))]

if \_\_name\_\_ == '\_\_main\_\_':

    fptr = open(os.environ['OUTPUT\_PATH'], 'w')

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

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

    result = compareTriplets(a, b)

    fptr.write(' '.join(map(str, result)))

    fptr.write('\n')

    fptr.close()

* In this challenge, you are required to calculate and print the sum of the elements in an array, keeping in mind that some of those integers may be quite large.

**Function Description:** Complete the *aVeryBigSum* function in the editor below. It must return the sum of all array elements.

aVeryBigSum has the following parameter(s):

* *int ar[n]*: an array of integers .

**Return:** *long*: the sum of all array elements

**Input Format:** The first line of the input consists of an integer n.  
The next line contains n space-separated integers contained in the array.

**Output Format:** Return the integer sum of the elements in the array.

**Sample Input:** 5

1000000001 1000000002 1000000003 1000000004 1000000005

**Output:** 5000000015

Ans:

import os

aVeryBigSum=lambda ar:sum(ar)

if \_\_name\_\_ == '\_\_main\_\_':

    fptr = open(os.environ['OUTPUT\_PATH'], 'w')

    ar\_count = int(input().strip())

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

    result = aVeryBigSum(ar)

    fptr.write(str(result) + '\n')

    fptr.close()

* Staircase detail: Its base and height are both equal to n. It is drawn using # symbols and spaces. *The last line is not preceded by any spaces.*

Write a program that prints a staircase of size n.

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

staircase has the following parameter(s):

* *int n*: an integer

**Print:** Print a staircase as described above.

**Input Format:** A single integer, n, denoting the size of the staircase.

**Output Format:** Print a staircase of size n using # symbols and spaces.

**Note**: The last line must have 0 spaces in it.

**Sample Input:** 6

**Sample Output:**  #

##

###

####

#####

######

Ans:

import os

def staircase(n):

    for i in range(n):

        print(" "\*(n-1-i),"#"\*(i+1),sep="")

if \_\_name\_\_ == '\_\_main\_\_':

    staircase(int(input().strip()))

* You are in charge of the cake for a child's birthday. You have decided the cake will have one candle for each year of their total age. They will only be able to blow out the tallest of the candles. Count how many candles are tallest.

**Example:** candles=[4,4,1,3]

The maximum height candles are 4 units high. There are 2 of them, so return 2.

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

birthdayCakeCandles has the following parameter(s):

* *int candles[n]*: the candle heights

**Returns:** *int*: the number of candles that are tallest

**Input Format:** The first line contains a single integer, n, the size of candles[].  
The second line contains n space-separated integers, where each integer i describes the height of candles[i].

**Sample Input:** 4

3 2 1 3

**Sample Output:** 2

Ans:

import os

from collections import Counter

birthdayCakeCandles=lambda candles:next(v for k,v in Counter(candles).most\_common(1))

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()

* HackerLand University has the following grading policy:
* Every student receives a grade in the inclusive range from 0 to 100.
* Any grade less than 40 is a failing grade.

Sam is a professor at the university and likes to round each student's grade according to these rules:

* If the difference between the grade and the next multiple of 5 is less than 3, round grade up to the next multiple of 5.
* If the value of grade is less than 38, no rounding occurs as the result will still be a failing grade.

**Examples:**

* grade=84 round to 85 (85 - 84 is less than 3)
* grade=29 do not round (result is less than 40)
* grade=57 do not round (60 - 57 is 3 or higher)

Given the initial value of grade for each of Sam's n students, write code to automate the rounding process.

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

gradingStudents has the following parameter(s):

* *int grades[n]*: the grades before rounding

**Returns:** *int[n]*: the grades after rounding as appropriate

**Input Format:** The first line contains a single integer,n , the number of students.  
Each line i of the n subsequent lines contains a single integer, grades[i].

**Sample Input:** 4

73

67

38

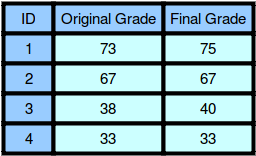
33

**Sample Output:** 75

67

40

33

**Explanation:** 

Ans:

import os

gradingStudents=lambda grades:[(g+5-g%5) if g>=38 and g%5>=3 else g for g in grades]

if \_\_name\_\_ == '\_\_main\_\_':

    fptr = open(os.environ['OUTPUT\_PATH'], 'w')

    grades\_count = int(input().strip())

    grades = []

    for \_ in range(grades\_count):

        grades\_item = int(input().strip())

        grades.append(grades\_item)

    result = gradingStudents(grades)

    fptr.write('\n'.join(map(str, result)))

    fptr.write('\n')

    fptr.close()

* Given an array of integers and a positive integer k, determine the number of (i,j) pairs where i<j and ar[i] + ar[j] is divisible by k.

**Example:** ar=[1,2,3,4,5,6]

k=5

Three pairs meet the criteria: [1,4],[2,3] and [4,6].

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

divisibleSumPairs has the following parameter(s):

* *int n:* the length of array ar
* *int ar[n]:* an array of integers
* *int k:* the integer divisor

**Returns:** - *int:* the number of pairs

**Input Format:** The first line contains 2 space-separated integers, n and k.  
The second line contains n space-separated integers, each a value of arr[i].

**Sample Input:** STDIN Function

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

6 3 n = 6, k = 3

1 3 2 6 1 2 ar = [1, 3, 2, 6, 1, 2]

**Sample Output**: 5

Ans:

import os

def divisibleSumPairs(n,k,ar):

    ar=sorted(ar)

    return sum(1 for i in range(n) for j in range(i+1,n) if (ar[i]+ar[j])%k==0)

if \_\_name\_\_ == '\_\_main\_\_':

    fptr = open(os.environ['OUTPUT\_PATH'], 'w')

    first\_multiple\_input = input().rstrip().split()

    n = int(first\_multiple\_input[0])

    k = int(first\_multiple\_input[1])

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

    result = divisibleSumPairs(n, k, ar)

    fptr.write(str(result) + '\n')

    fptr.close()

* Given an array of bird sightings where every element represents a bird type id, determine the id of the most frequently sighted type. If more than 1 type has been spotted that maximum amount, return the smallest of their ids.

**Example**: arr=[1,1,2,2,3]

There are two each of types 1 and 2, and one sighting of type 3. Pick the lower of the two types seen twice: type 1.

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

migratoryBirds has the following parameter(s):

* *int arr[n]*: the types of birds sighted

**Returns:** *int:* the lowest type id of the most frequently sighted birds

**Input Format:** The first line contains an integer, n, the size of arr.  
The second line describes arr as n space-separated integers, each a type number of the bird sighted.

**Sample Input:** 6

1 4 4 4 5 3

**Sample Output:** 4

Ans:

from collections import Counter

import os

def migratoryBirds(arr):

    c=Counter(arr)

    f=next(v for k,v in c.most\_common(1))

    return min(k for k,v in c.items() if f==v)

if \_\_name\_\_ == '\_\_main\_\_':

    fptr = open(os.environ['OUTPUT\_PATH'], 'w')

    arr\_count = int(input().strip())

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

    result = migratoryBirds(arr)

    fptr.write(str(result) + '\n')

    fptr.close()

* Maria plays college basketball and wants to go pro. Each season she maintains a record of her play. She tabulates the number of times she breaks her season record for *most points* and *least points* in a game. Points scored in the first game establish her record for the season, and she begins counting from there.

**Example:** scores=[12,24,10,24]

Scores are in the same order as the games played. She tabulates her results as follows:

Count

Game Score Minimum Maximum Min Max

0 12 12 12 0 0

1 24 12 24 0 1

2 10 10 24 1 1

3 24 10 24 1 1

Given the scores for a season, determine the number of times Maria breaks her records for *most* and *least* points scored during the season.

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

breakingRecords has the following parameter(s):

* *int scores[n]:* points scored per game

**Returns:** *int[2]:* An array with the numbers of times she broke her records. Index 0 is for breaking *most points* records, and index 1 is for breaking *least points* records.

**Input Format:** The first line contains an integer n, the number of games.  
The second line contains n space-separated integers describing the respective values of score0, score1,…, scoren-1.

**Sample Input:** 9

10 5 20 20 4 5 2 25 1

**Sample Output:** 2 4

Ans:

import os

def breakingRecords(scores):

    min,max,minc,maxc=scores[0],scores[0],0,0

    for i in scores:

        if i>max:max=i;maxc+=1

        elif i<min:min=i;minc+=1

    return [maxc,minc]

if \_\_name\_\_ == '\_\_main\_\_':

    fptr = open(os.environ['OUTPUT\_PATH'], 'w')

    n = int(input().strip())

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

    result = breakingRecords(scores)

    fptr.write(' '.join(map(str, result)))

    fptr.write('\n')

    fptr.close()

* Two children, Lily and Ron, want to share a chocolate bar. Each of the squares has an integer on it.

Lily decides to share a contiguous segment of the bar selected such that:

* The length of the segment matches Ron's birth month, and,
* The sum of the integers on the squares is equal to his birth day.

Determine how many ways she can divide the chocolate.

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

birthday has the following parameter(s):

* *int s[n]:* the numbers on each of the squares of chocolate
* *int d:* Ron's birth day
* *int m:* Ron's birth month

**Returns:** *int:* the number of ways the bar can be divided

**Input Format:** The first line contains an integer n, the number of squares in the chocolate bar.  
The second line contains n space-separated integers s[i], the numbers on the chocolate squares where 0<=i<n.  
The third line contains two space-separated integers, d and m, Ron's birth day and his birth month.

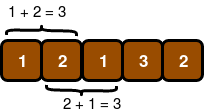
**Sample Input:** 5

1 2 1 3 2

3 2

**Sample Output:** 2

**Explanation:** Lily wants to give Ron m=2 squares summing to d=3. The following two segments meet the criteria:



Ans:

import os

birthday=lambda s,d,m:sum(1 for i in range(len(s)) if sum(s[i:i+m])==d)

if \_\_name\_\_ == '\_\_main\_\_':

    fptr = open(os.environ['OUTPUT\_PATH'], 'w')

    n = int(input().strip())

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

    first\_multiple\_input = input().rstrip().split()

    d = int(first\_multiple\_input[0])

    m = int(first\_multiple\_input[1])

    result = birthday(s, d, m)

    fptr.write(str(result) + '\n')

    fptr.close()

* Two cats and a mouse are at various positions on a line. You will be given their starting positions. Your task is to determine which cat will reach the mouse first, assuming the mouse does not move and the cats travel at equal speed. If the cats arrive at the same time, the mouse will be allowed to move and it will escape while they fight.

You are given q queries in the form of x, y, and z representing the respective positions for cats A and B, and for mouse C. Complete the function catAndMouse to return the appropriate answer to each query, which will be printed on a new line.

* If cat A catches the mouse first, print Cat A.
* If cat B catches the mouse first, print Cat B.
* If both cats reach the mouse at the same time, print Mouse C as the two cats fight and mouse escapes.

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

catAndMouse has the following parameter(s):

* *int x*: Cat A's position
* *int y*: Cat B's position
* *int z*: Mouse C's position

**Returns:** *string:* Either 'Cat A', 'Cat B', or 'Mouse C'

**Input Format:** The first line contains a single integer, q, denoting the number of queries.  
Each of the q subsequent lines contains three space-separated integers describing the respective values of  x(cat A's location),  y(cat B's location), and  z(mouse C's location).

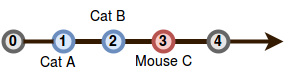
**Sample Input:** 2

1 2 3

1 3 2

**Sample Output:** Cat B

Mouse C

**Explanation:** The positions of the cats and mouse are shown below: 

Ans:

import os

catAndMouse=lambda catA,catB,mouseC:"Cat A" if abs(catA-mouseC)<abs(catB-mouseC) else "Cat B" if abs(catA-mouseC)>abs(catB-mouseC) else "Mouse C"

if \_\_name\_\_ == '\_\_main\_\_':

    fptr = open(os.environ['OUTPUT\_PATH'], 'w')

    q = int(input())

    for q\_itr in range(q):

        xyz = input().split()

        x = int(xyz[0])

        y = int(xyz[1])

        z = int(xyz[2])

        result = catAndMouse(x, y, z)

        fptr.write(result + '\n')

    fptr.close()

* There is a sequence of words in [CamelCase](https://en.wikipedia.org/wiki/CamelCase) as a string of letters, s, having the following properties:
* It is a concatenation of one or more words consisting of English letters.
* All letters in the first word are lowercase.
* For each of the subsequent words, the first letter is uppercase and rest of the letters are lowercase.

Given s, determine the number of words in s.

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

camelcase has the following parameter(s):

* string s: the string to analyze

**Returns:** int: the number of words in s.

**Input Format:** A single line containing string s.

**Sample Input:** saveChangesInTheEditor

**Sample Output:** 5

Ans:

import os

camelcase=lambda s:sum(1 for i in s if i.isupper())+1

if \_\_name\_\_ == '\_\_main\_\_':

    fptr = open(os.environ['OUTPUT\_PATH'], 'w')

    s = input()

    result = camelcase(s)

    fptr.write(str(result) + '\n')

    fptr.close()

* Louise joined a social networking site to stay in touch with her friends. The signup page required her to input a *name* and a *password*. However, the password must be *strong*. The website considers a password to be *strong* if it satisfies the following criteria:
* Its length is at least 6.
* It contains at least one digit.
* It contains at least one lowercase English character.
* It contains at least one uppercase English character.
* It contains at least one special character. The special characters are: !@#$%^&\*()-+

She typed a random string of length n in the password field but wasn't sure if it was strong. Given the string she typed, can you find the minimum number of characters she must add to make her password strong?

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

*minimumNumber* has the following parameters:

* *int n:* the length of the password
* *string password:* the password to test

**Returns:** *int:* the minimum number of characters to add

**Input Format:** The first line contains an integer n, the length of the password.

The second line contains the password string. Each character is either a lowercase/uppercase English alphabet, a digit, or a special character.

**Sample Input:** 3

Ab1

**Sample Output:** 3

Ans:

import os

import re

def minimumNumber(n,password):

    patterns=['[0-9]','[a-z]','[A-Z]','[!@#$%^&\*()+-]']

    missing\_types=sum(1 for pattern in patterns if not re.search(pattern,password))

    return max(missing\_types,6-n)

if \_\_name\_\_ == '\_\_main\_\_':

    fptr = open(os.environ['OUTPUT\_PATH'], 'w')

    n = int(input().strip())

    password = input()

    answer = minimumNumber(n, password)

    fptr.write(str(answer) + '\n')

    fptr.close()