* 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

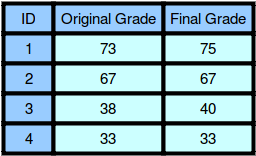
33

**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()