1) add (<https://codefights.com/arcade/intro/level-1/jwr339Kq6e3LQTsfa/description>)

Write a function that returns the sum of two numbers.

**Example**

For param1 = 1 and param2 = 2, the output should be  
add(param1, param2) = 3.

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] integer param1**

Guaranteed constraints:  
-100 ≤ param1 ≤ 1000.

* **[input] integer param2**

Guaranteed constraints:  
-100 ≤ param2 ≤ 1000.

* **[output] integer**

The sum of the two inputs.

2) CenturyFromYear (<https://codefights.com/arcade/intro/level-1/egbueTZRRL5Mm4TXN>)

Given a year, return the century it is in. The first century spans from the year 1 up to and including the year 100, the second - from the year 101 up to and including the year 200, etc.

**Example**

* For year = 1905, the output should be  
  centuryFromYear(year) = 20;
* For year = 1700, the output should be  
  centuryFromYear(year) = 17.

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] integer year**

A positive integer, designating the year.

*Guaranteed constraints:*  
1 ≤ year ≤ 2005.

* **[output] integer**

The number of the century the year is in.

3) CheckPalindrome (<https://codefights.com/arcade/intro/level-1/s5PbmwxfECC52PWyQ>)

Given the string, check if it is a [palindrome](keyword://palindrome).

**Example**

* For inputString = "aabaa", the output should be  
  checkPalindrome(inputString) = true;
* For inputString = "abac", the output should be  
  checkPalindrome(inputString) = false;
* For inputString = "a", the output should be  
  checkPalindrome(inputString) = true.

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] string inputString**

A non-empty string consisting of lowercase characters.

Guaranteed constraints:  
1 ≤ inputString.length ≤ 105.

* **[output] boolean**

true if inputString is a palindrome, false otherwise.

4) adjacentElementsProduct (<https://codefights.com/arcade/intro/level-2/xzKiBHjhoinnpdh6m>)

Given an array of integers, find the pair of adjacent elements that has the largest product and return that product.

**Example**

For inputArray = [3, 6, -2, -5, 7, 3], the output should be  
adjacentElementsProduct(inputArray) = 21.

7 and 3 produce the largest product.

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] array.integer inputArray**

An array of integers containing at least two elements.

*Guaranteed constraints:*  
2 ≤ inputArray.length ≤ 10,  
-1000 ≤ inputArray[i] ≤ 1000.

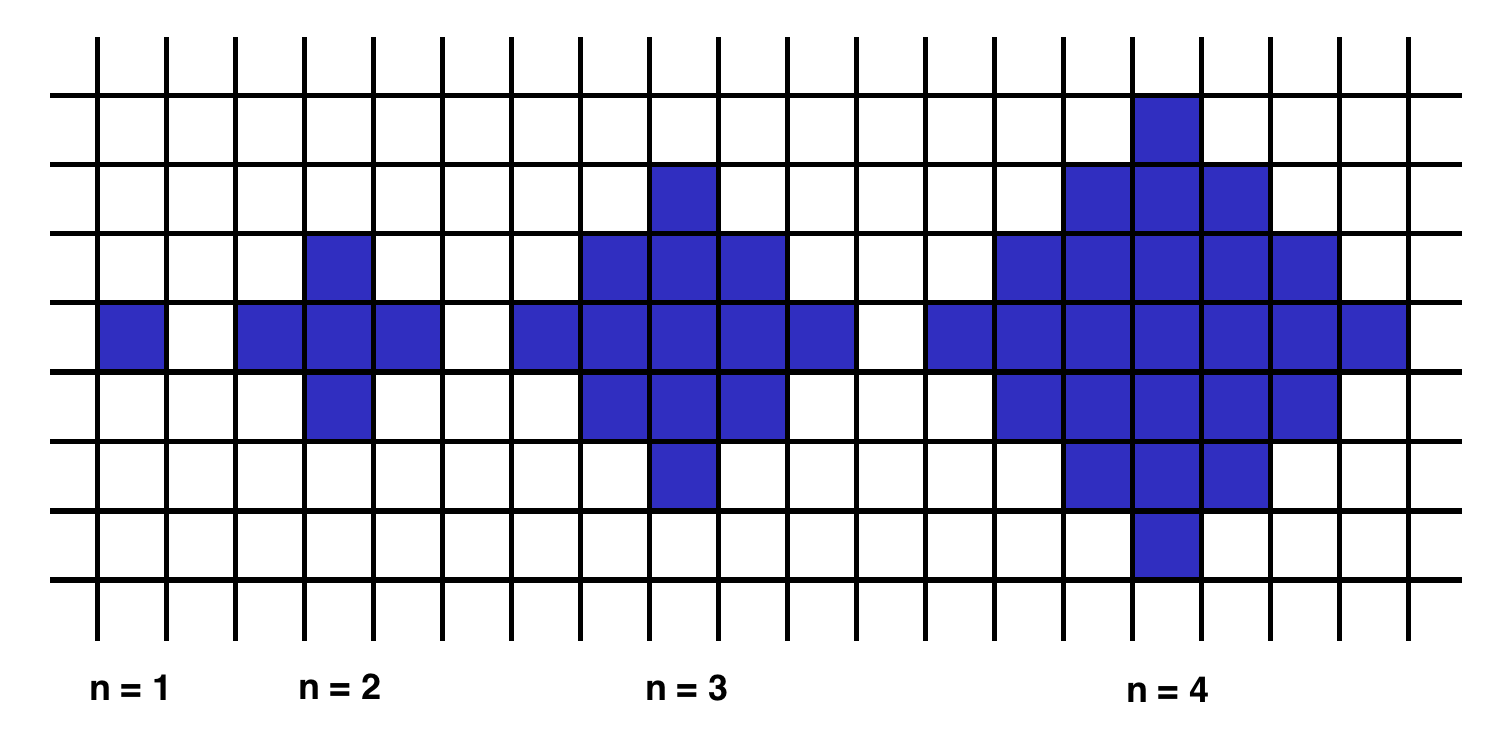
* **[output] integer**

The largest product of adjacent elements.

5) shapeArea (<https://codefights.com/arcade/intro/level-2/yuGuHvcCaFCKk56rJ>)

Below we will define an n-interesting polygon. Your task is to find the area of a polygon for a given n.

A 1-interesting polygon is just a square with a side of length 1. An n-interesting polygon is obtained by taking the n - 1-interesting polygon and appending 1-interesting polygons to its rim, side by side. You can see the 1-, 2-, 3- and 4-interesting polygons in the picture below.



**Example**

* For n = 2, the output should be  
  shapeArea(n) = 5;
* For n = 3, the output should be  
  shapeArea(n) = 13.

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] integer n**

*Guaranteed constraints:*  
1 ≤ n < 104.

* **[output] integer**

The area of the n-interesting polygon.

6) makeArrayConsecutive2 (<https://codefights.com/arcade/intro/level-2/bq2XnSr5kbHqpHGJC>)

Ratiorg got statues of *different* sizes as a present from CodeMaster for his birthday, each statue having an non-negative integer size. Since he likes to make things perfect, he wants to arrange them from smallest to largest so that each statue will be bigger than the previous one exactly by 1. He may need some additional statues to be able to accomplish that. Help him figure out the minimum number of additional statues needed.

**Example**

For statues = [6, 2, 3, 8], the output should be  
makeArrayConsecutive2(statues) = 3.

Ratiorg needs statues of sizes 4, 5 and 7.

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] array.integer statues**

An array of *distinct* non-negative integers.

*Guaranteed constraints:*  
1 ≤ statues.length ≤ 10,  
0 ≤ statues[i] ≤ 20.

* **[output] integer**

The minimal number of statues that need to be added to existing statues such that it contains every integer size from an interval [L, R] (for some L, R) and no other sizes.

7) almostIncreasingSequence (<https://codefights.com/arcade/intro/level-2/2mxbGwLzvkTCKAJMG>)

Given a sequence of integers as an array, determine whether it is possible to obtain a strictly increasing sequence by removing no more than one element from the array.

**Example**

* For sequence = [1, 3, 2, 1], the output should be  
  almostIncreasingSequence(sequence) = false;

There is no one element in this array that can be removed in order to get a strictly increasing sequence.

* For sequence = [1, 3, 2], the output should be  
  almostIncreasingSequence(sequence) = true.

You can remove 3 from the array to get the strictly increasing sequence [1, 2]. Alternately, you can remove 2 to get the strictly increasing sequence [1, 3].

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] array.integer sequence**

*Guaranteed constraints:*  
2 ≤ sequence.length ≤ 105,  
-105 ≤ sequence[i] ≤ 105.

* **[output] boolean**

Return true if it is possible to remove one element from the array in order to get a strictly increasing sequence, otherwise return false.

8) matrixElementsSum (<https://codefights.com/arcade/intro/level-2/xskq4ZxLyqQMCLshr>)

After they became famous, the CodeBots all decided to move to a new building and live together. The building is represented by a rectangular matrix of rooms. Each cell in the matrix contains an integer that represents the price of the room. Some rooms are *free*(their cost is 0), but that's probably because they are haunted, so all the bots are afraid of them. That is why any room that is *free* or is located **anywhere below** a *free* room in the same column is not considered suitable for the bots to live in.

Help the bots calculate the total price of all the rooms that are suitable for them.

**Example**

* For

matrix = [[0, 1, 1, 2],

[0, 5, 0, 0],

[2, 0, 3, 3]]

the output should be  
matrixElementsSum(matrix) = 9.

Here's the rooms matrix with unsuitable rooms marked with 'x':

[[x, 1, 1, 2],

[x, 5, x, x],

[x, x, x, x]]

Thus, the answer is 1 + 5 + 1 + 2 = 9.

* For

matrix = [[1, 1, 1, 0],

[0, 5, 0, 1],

[2, 1, 3, 10]]

the output should be  
matrixElementsSum(matrix) = 9.

Here's the rooms matrix with unsuitable rooms marked with 'x':

[[1, 1, 1, x],

[x, 5, x, x],

[x, 1, x, x]]

Note that the free room in the first row make the full column unsuitable for bots.

Thus, the answer is 1 + 1 + 1 + 5 + 1 = 9.

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] array.array.integer matrix**

A 2-dimensional array of integers representing a rectangular matrix of the building.

*Guaranteed constraints:*  
1 ≤ matrix.length ≤ 5,  
1 ≤ matrix[i].length ≤ 5,  
0 ≤ matrix[i][j] ≤ 10.

* **[output] integer**

The total price of all the rooms that are suitable for the CodeBots to live in.

9) allLongestString (<https://codefights.com/arcade/intro/level-3/fzsCQGYbxaEcTr2bL>)

Given an array of strings, return another array containing all of its longest strings.

**Example**

For inputArray = ["aba", "aa", "ad", "vcd", "aba"], the output should be  
allLongestStrings(inputArray) = ["aba", "vcd", "aba"].

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] array.string inputArray**

A non-empty array.

*Guaranteed constraints:*  
1 ≤ inputArray.length ≤ 10,  
1 ≤ inputArray[i].length ≤ 10.

* **[output] array.string**

Array of the longest strings, stored in the same order as in the inputArray.

10) commonCharacterCount (<https://codefights.com/arcade/intro/level-3/JKKuHJknZNj4YGL32>)

Given two strings, find the number of common characters between them.

**Example**

For s1 = "aabcc" and s2 = "adcaa", the output should be  
commonCharacterCount(s1, s2) = 3.

Strings have 3 common characters - 2 "a"s and 1 "c".

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] string s1**

A string consisting of lowercase latin letters a-z.

*Guaranteed constraints:*  
1 ≤ s1.length ≤ 15.

* **[input] string s2**

A string consisting of lowercase latin letters a-z.

*Guaranteed constraints:*  
1 ≤ s2.length ≤ 15.

* **[output] integer**

11) isLucky (<https://codefights.com/arcade/intro/level-3/3AdBC97QNuhF6RwsQ>)

Ticket numbers usually consist of an even number of digits. A ticket number is considered *lucky* if the sum of the first half of the digits is equal to the sum of the second half.

Given a ticket number n, determine if it's *lucky* or not.

**Example**

* For n = 1230, the output should be  
  isLucky(n) = true;
* For n = 239017, the output should be  
  isLucky(n) = false.

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] integer n**

A ticket number represented as a positive integer with an even number of digits.

*Guaranteed constraints:*  
10 ≤ n < 106.

* **[output] boolean**

true if n is a lucky ticket number, false otherwise.

12) sortByHeight (<https://codefights.com/arcade/intro/level-3/D6qmdBL2NYz49XHwM>)

Some people are standing in a row in a park. There are trees between them which cannot be moved. Your task is to rearrange the people by their heights in a non-descending order without moving the trees.

**Example**

For a = [-1, 150, 190, 170, -1, -1, 160, 180], the output should be  
sortByHeight(a) = [-1, 150, 160, 170, -1, -1, 180, 190].

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] array.integer a**

If a[i] = -1, then the ith position is occupied by a tree. Otherwise a[i] is the height of a person standing in the ithposition.

*Guaranteed constraints:*  
5 ≤ a.length ≤ 15,  
-1 ≤ a[i] ≤ 200.

* **[output] array.integer**

Sorted array a with all the trees untouched.

13) reverseParentheses (<https://codefights.com/arcade/intro/level-3/3o6QFqgYSontKsyk4>)

You have a string s that consists of English letters, punctuation marks, whitespace characters, and brackets. It is guaranteed that the parentheses in s form a [regular bracket sequence](keyword://regular-bracket-sequence).

Your task is to reverse the strings contained in each pair of matching parentheses, starting from the innermost pair. The results string should not contain any parentheses.

**Example**

For string s = "a(bc)de", the output should be  
reverseParentheses(s) = "acbde".

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] string s**

A string consisting of English letters, punctuation marks, whitespace characters and brackets. It is guaranteed that parentheses form a regular bracket sequence.

Constraints:  
5 ≤ s.length ≤ 55.

* **[output] string**

14) alternatingSums (<https://codefights.com/arcade/intro/level-4/cC5QuL9fqvZjXJsW9>)

Several people are standing in a row and need to be divided into two teams. The first person goes into *team 1*, the second goes into *team 2*, the third goes into *team 1* again, the fourth into *team 2*, and so on.

You are given an array of positive integers - the weights of the people. Return an array of two integers, where the first element is the total weight of *team 1*, and the second element is the total weight of *team 2*after the division is complete.

**Example**

For a = [50, 60, 60, 45, 70], the output should be  
alternatingSums(a) = [180, 105].

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] array.integer a**

*Guaranteed constraints:*  
1 ≤ a.length ≤ 10,  
45 ≤ a[i] ≤ 100.

* **[output] array.integer**

15) addBorder (<https://codefights.com/arcade/intro/level-4/ZCD7NQnED724bJtjN>)

Given a rectangular matrix of characters, add a border of asterisks(\*) to it.

**Example**

For

picture = ["abc",

"ded"]

the output should be

addBorder(picture) = ["\*\*\*\*\*",

"\*abc\*",

"\*ded\*",

"\*\*\*\*\*"]

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] array.string picture**

A non-empty array of non-empty equal-length strings.

*Guaranteed constraints:*  
1 ≤ picture.length ≤ 5,  
1 ≤ picture[i].length ≤ 5.

* **[output] array.string**

The same matrix of characters, framed with a border of asterisks of width 1.

16) areSimilar? (<https://codefights.com/arcade/intro/level-4/xYXfzQmnhBvEKJwXP>)

Two arrays are called similar if one can be obtained from another by swapping at most one pair of elements in one of the arrays.

Given two arrays a and b, check whether they are similar.

**Example**

* For a = [1, 2, 3] and b = [1, 2, 3], the output should be  
  areSimilar(a, b) = true.

The arrays are equal, no need to swap any elements.

* For a = [1, 2, 3] and b = [2, 1, 3], the output should be  
  areSimilar(a, b) = true.

We can obtain b from a by swapping 2 and 1 in b.

* For a = [1, 2, 2] and b = [2, 1, 1], the output should be  
  areSimilar(a, b) = false.

Any swap of any two elements either in a or in b won't make aand b equal.

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] array.integer a**

Array of integers.

Guaranteed constraints:  
3 ≤ a.length ≤ 105,  
1 ≤ a[i] ≤ 1000.

* **[input] array.integer b**

Array of integers of the same length as a.

Guaranteed constraints:  
b.length = a.length,  
1 ≤ b[i] ≤ 1000.

* **[output] boolean**

true if a and b are similar, false otherwise.

17) arrayChange (<https://codefights.com/arcade/intro/level-4/xvkRbxYkdHdHNCKjg>)

You are given an array of integers. On each move you are allowed to increase exactly one of its element by one. Find the minimal number of moves required to obtain a strictly increasing sequence from the input.

**Example**

For inputArray = [1, 1, 1], the output should be  
arrayChange(inputArray) = 3.

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] array.integer inputArray**

*Guaranteed constraints:*  
3 ≤ inputArray.length ≤ 105,  
-105 ≤ inputArray[i] ≤ 105.

* **[output] integer**

The minimal number of moves needed to obtain a strictly increasing sequence from inputArray.  
It's guaranteed that for the given test cases the answer always fits signed 32-bit integer type.

18) palindromeRearranging (<https://codefights.com/arcade/intro/level-4/Xfeo7r9SBSpo3Wico>)

Given a string, find out if its characters can be rearranged to form a [palindrome](keyword://palindrome).

**Example**

For inputString = "aabb", the output should be  
palindromeRearranging(inputString) = true.

We can rearrange "aabb" to make "abba", which is a palindrome.

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] string inputString**

A string consisting of lowercase English letters.

*Guaranteed constraints:*  
1 ≤ inputString.length ≤ 50.

* **[output] boolean**

true if the characters of the inputString can be rearranged to form a palindrome, false otherwise.

19) areEquallyStrong (<https://codefights.com/arcade/intro/level-5/g6dc9KJyxmFjB98dL>)

Call two arms *equally strong* if the heaviest weights they each are able to lift are equal.

Call two people *equally strong* if their strongest arms are equally strong (the strongest arm can be both the right and the left), and so are their weakest arms.

Given your and your friend's arms' lifting capabilities find out if you two are equally strong.

**Example**

* For yourLeft = 10, yourRight = 15, friendsLeft = 15 and friendsRight = 10, the output should be  
  areEquallyStrong(yourLeft, yourRight, friendsLeft, friendsRight) = true;
* For yourLeft = 15, yourRight = 10, friendsLeft = 15 and friendsRight = 10, the output should be  
  areEquallyStrong(yourLeft, yourRight, friendsLeft, friendsRight) = true;
* For yourLeft = 15, yourRight = 10, friendsLeft = 15 and friendsRight = 9, the output should be  
  areEquallyStrong(yourLeft, yourRight, friendsLeft, friendsRight) = false.

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] integer yourLeft**

A non-negative integer representing the heaviest weight you can lift with your left arm.

*Guaranteed constraints:*  
0 ≤ yourLeft ≤ 20.

* **[input] integer yourRight**

A non-negative integer representing the heaviest weight you can lift with your right arm.

*Guaranteed constraints:*  
0 ≤ yourRight ≤ 20.

* **[input] integer friendsLeft**

A non-negative integer representing the heaviest weight your friend can lift with his or her left arm.

*Guaranteed constraints:*  
0 ≤ friendsLeft ≤ 20.

* **[input] integer friendsRight**

A non-negative integer representing the heaviest weight your friend can lift with his or her right arm.

*Guaranteed constraints:*  
0 ≤ friendsRight ≤ 20.

* **[output] boolean**

true if you and your friend are equally strong, false otherwise.

20) arrayMaximalAdjacentDifference (<https://codefights.com/arcade/intro/level-5/EEJxjQ7oo7C5wAGjE>)

Given an array of integers, find the maximal absolute difference between any two of its adjacent elements.

**Example**

For inputArray = [2, 4, 1, 0], the output should be  
arrayMaximalAdjacentDifference(inputArray) = 3.

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] array.integer inputArray**

*Guaranteed constraints:*  
3 ≤ inputArray.length ≤ 10,  
-15 ≤ inputArray[i] ≤ 15.

* **[output] integer**

The maximal absolute difference.

21) isIPv4Address (<https://codefights.com/arcade/intro/level-5/veW5xJednTy4qcjso>)

An IP address is a numerical label assigned to each device (e.g., computer, printer) participating in a computer network that uses the Internet Protocol for communication. There are two versions of the Internet protocol, and thus two versions of addresses. One of them is the *IPv4 address*.

*IPv4 addresses* are represented in dot-decimal notation, which consists of four decimal numbers, each ranging from 0 to 255 inclusive, separated by dots, e.g., 172.16.254.1.

Given a string, find out if it satisfies the *IPv4 address*naming rules.

**Example**

* For inputString = "172.16.254.1", the output should be  
  isIPv4Address(inputString) = true;
* For inputString = "172.316.254.1", the output should be  
  isIPv4Address(inputString) = false.

316 is not in range [0, 255].

* For inputString = ".254.255.0", the output should be  
  isIPv4Address(inputString) = false.

There is no first number.

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] string inputString**

*Guaranteed constraints:*  
1 ≤ inputString.length ≤ 30.

* **[output] boolean**

true if inputString satisfies the IPv4 address naming rules, false otherwise.

22) avoidObstacles (<https://codefights.com/arcade/intro/level-5/XC9Q2DhRRKQrfLhb5>)

You are given an array of integers representing coordinates of obstacles situated on a straight line.

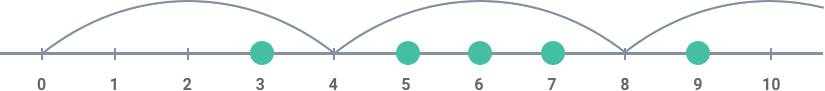
Assume that you are jumping from the point with coordinate 0 to the right. You are allowed only to make jumps of the same length represented by some integer.

Find the minimal length of the jump enough to avoid all the obstacles.

**Example**

For inputArray = [5, 3, 6, 7, 9], the output should be  
avoidObstacles(inputArray) = 4.

Check out the image below for better understanding:



**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] array.integer inputArray**

Non-empty array of positive integers.

*Guaranteed constraints:*  
2 ≤ inputArray.length ≤ 10,  
1 ≤ inputArray[i] ≤ 40.

* **[output] integer**

The desired length.

23) boxBlur (<https://codefights.com/arcade/intro/level-5/5xPitc3yT3dqS7XkP>)

Last night you partied a little too hard. Now there's a black and white photo of you that's about to go viral! You can't let this ruin your reputation, so you want to apply the [box bluralgorithm](https://en.wikipedia.org/wiki/Box_blur) to the photo to hide its content.

The pixels in the input image are represented as integers. The algorithm distorts the input image in the following way: Every pixel x in the output image has a value equal to the average value of the pixel values from the 3 × 3 square that has its center at x, including x itself. All the pixels on the border of x are then removed.

Return the blurred image as an integer, with the fractions rounded down.

**Example**

For image = [[1, 1, 1],

[1, 7, 1],

[1, 1, 1]]

the output should be boxBlur(image) = [[1]].

To get the value of the middle pixel in the input 3 × 3 square: (1 + 1 + 1 + 1 + 7 + 1 + 1 + 1 + 1) = 15 / 9 = 1.66666 = 1. The border pixels are cropped from the final result.

For image = [[7, 4, 0, 1],

[5, 6, 2, 2],

[6, 10, 7, 8],

[1, 4, 2, 0]]

the output should be boxBlur(image) = [[5, 4],

[4, 4]]

There are four 3 × 3 squares in the input image, so there should be four integers in the blurred output.

To get the first value: (7 + 4 + 0 + 5 + 6 + 2 + 6 + 10 + 7) = 47 / 9 = 5.2222 = 5. The other three integers are obtained the same way, then the surrounding integers are cropped from the final result.

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] array.array.integer image**

An image, stored as a rectangular matrix of non-negative integers.

*Guaranteed constraints:*  
3 ≤ image.length ≤ 10,  
3 ≤ image[0].length ≤ 10,  
0 ≤ image[i][j] ≤ 255.

* **[output] array.array.integer**

A blurred image represented as integers, obtained through the process in the description.

24) minesweeper (<https://codefights.com/arcade/intro/level-5/ZMR5n7vJbexnLrgaM>)

In the popular **Minesweeper**game you have a board with some mines and those cells that don't contain a mine have a number in it that indicates the total number of mines in the neighboring cells. Starting off with some arrangement of mines we want to create a **Minesweeper**game setup.

**Example**

For

matrix = [[true, false, false],

[false, true, false],

[false, false, false]]

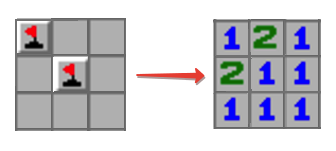
the output should be

minesweeper(matrix) = [[1, 2, 1],

[2, 1, 1],

[1, 1, 1]]

Check out the image below for better understanding:



**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] array.array.boolean matrix**

A non-empty rectangular matrix consisting of boolean values - true if the corresponding cell contains a mine, false otherwise.

*Guaranteed constraints:*  
2 ≤ matrix.length ≤ 5,  
2 ≤ matrix[0].length ≤ 5.

* **[output] array.array.integer**

Rectangular matrix of the same size as matrix each cell of which contains an integer equal to the number of mines in the neighboring cells. Two cells are called neighboring if they share at least one corner.

25) arrayReplace (<https://codefights.com/arcade/intro/level-6/mCkmbxdMsMTjBc3Bm>)

Given an array of integers, replace all the occurrences of elemToReplace with substitutionElem.

**Example**

For inputArray = [1, 2, 1], elemToReplace = 1and substitutionElem = 3, the output should be  
arrayReplace(inputArray, elemToReplace, substitutionElem) = [3, 2, 3].

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] array.integer inputArray**

*Guaranteed constraints:*  
2 ≤ inputArray.length ≤ 10,  
0 ≤ inputArray[i] ≤ 10.

* **[input] integer elemToReplace**

*Guaranteed constraints:*  
0 ≤ elemToReplace ≤ 10.

* **[input] integer substitutionElem**

*Guaranteed constraints:*  
0 ≤ substitutionElem ≤ 10.

* **[output] array.integer**

26) evenDigitsOnly (<https://codefights.com/arcade/intro/level-6/6cmcmszJQr6GQzRwW>)

Check if all digits of the given integer are even.

**Example**

* For n = 248622, the output should be  
  evenDigitsOnly(n) = true;
* For n = 642386, the output should be  
  evenDigitsOnly(n) = false.

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] integer n**

*Guaranteed constraints:*  
1 ≤ n ≤ 109.

* **[output] boolean**

true if all digits of n are even, false otherwise.

27) alphabeticShift (<https://codefights.com/arcade/intro/level-6/PWLT8GBrv9xXy4Dui>)

Given a string, replace each its character by the next one in the English alphabet (z would be replaced by a).

**Example**

For inputString = "crazy", the output should be  
alphabeticShift(inputString) = "dsbaz".

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] string inputString**

Non-empty string consisting of lowercase English characters.

*Guaranteed constraints:*  
1 ≤ inputString.length ≤ 1000.

* **[output] string**

The result string after replacing all of its characters.

28) alphabeticShift (<https://codefights.com/arcade/intro/level-6/PWLT8GBrv9xXy4Dui>)

Given a string, replace each its character by the next one in the English alphabet (z would be replaced by a).

**Example**

For inputString = "crazy", the output should be  
alphabeticShift(inputString) = "dsbaz".

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] string inputString**

Non-empty string consisting of lowercase English characters.

*Guaranteed constraints:*  
1 ≤ inputString.length ≤ 1000.

* **[output] string**

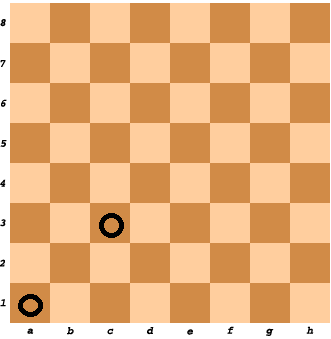
The result string after replacing all of its characters.

29) chessBoardColor (<https://codefights.com/arcade/intro/level-6/t97bpjfrMDZH8GJhi>)

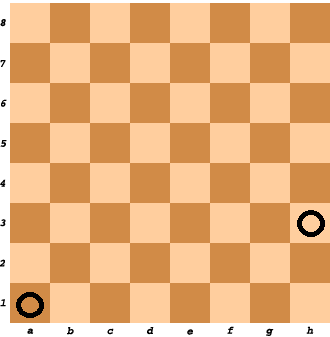
Given two cells on the standard chess board, determine whether they have the same color or not.

**Example**

* For cell1 = "A1" and cell2 = "C3", the output should be  
  chessBoardCellColor(cell1, cell2) = true.



* For cell1 = "A1" and cell2 = "H3", the output should be  
  chessBoardCellColor(cell1, cell2) = false.



**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] string cell1**
* **[input] string cell2**
* **[output] boolean**

true if both cells have the same color, false otherwise.

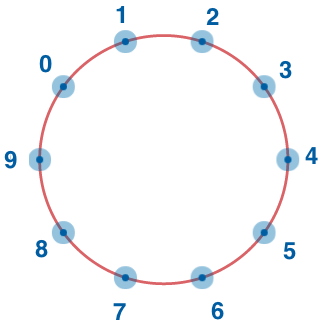
30) circleofNumbers (<https://codefights.com/arcade/intro/level-7/vExYvcGnFsEYSt8nQ>)

Consider integer numbers from 0 to n - 1 written down along the circle in such a way that the distance between any two neighbouring numbers is equal (note that 0 and n - 1 are neighbouring, too).

Given n and firstNumber, find the number which is written in the radially opposite position to firstNumber.

**Example**

For n = 10 and firstNumber = 2, the output should be  
circleOfNumbers(n, firstNumber) = 7.



**Input/Output**

**[execution time limit] 3 seconds (kt)**

**[input] integer n**

A positive **even** integer.

*Guaranteed constraints:*  
4 ≤ n ≤ 20.

**[input] integer firstNumber**

*Guaranteed constraints:*  
0 ≤ firstNumber ≤ n - 1.

**[output] integer**

31) depositProfit (<https://codefights.com/arcade/intro/level-7/8PxjMSncp9ApA4DAb>)

You have deposited a specific amount of dollars into your bank account. Each year your balance increases at the same growth rate. Find out how long it would take for your balance to pass a specific threshold with the assumption that you don't make any additional deposits.

**Example**

For deposit = 100, rate = 20 and threshold = 170, the output should be  
depositProfit(deposit, rate, threshold) = 3.

Each year the amount of money on your account increases by 20%. It means that throughout the years your balance would be:

* year 0: 100;
* year 1: 120;
* year 2: 144;
* year 3: 172,8.

Thus, it will take 3 years for your balance to pass the threshold, which is the answer.

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] integer deposit**

The initial deposit as a positive integer.

*Guaranteed constraints:*  
1 ≤ deposit ≤ 100.

* **[input] integer rate**

The rate of increase. Each year the balance increases by the rate*percent* of the current sum.

*Guaranteed constraints:*  
1 ≤ rate ≤ 100.

* **[input] integer threshold**

The target balance.

*Guaranteed constraints:*  
deposit < threshold ≤ 200.

* **[output] integer**

The number of years it would take to hit the threshold.

32) absoluteValuesSumMinimization (<https://codefights.com/arcade/intro/level-7/ZFnQkq9RmMiyE6qtq>)

Given a sorted array of integers a, find an integer x from a such that the value of

abs(a[0] - x) + abs(a[1] - x) + ... + abs(a[a.length - 1] - x)

is the *smallest possible* (here absdenotes the absolute value).

If there are several possible answers, output the *smallest* one.

**Example**

For a = [2, 4, 7], the output should be  
absoluteValuesSumMinimization(a) = 4.

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] array.integer a**

A non-empty array of integers, sorted in ascending order.

*Guaranteed constraints:*  
1 ≤ a.length ≤ 200,  
-106 ≤ a[i] ≤ 106.

* **[output] integer**

33) stringsRearrangement (<https://codefights.com/arcade/intro/level-7/PTWhv2oWqd6p4AHB9>)

Given an array of equal-length strings, check if it is possible to rearrange the strings in such a way that after the rearrangement the strings at consecutive positions would differ by exactly one character.

**Example**

* For inputArray = ["aba", "bbb", "bab"], the output should be  
  stringsRearrangement(inputArray) = false;

All rearrangements don't satisfy the description condition.

* For inputArray = ["ab", "bb", "aa"], the output should be  
  stringsRearrangement(inputArray) = true.

Strings can be rearranged in the following way: "aa", "ab", "bb".

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] array.string inputArray**

A non-empty array of strings of lowercase letters.

*Guaranteed constraints:*  
2 ≤ inputArray.length ≤ 10,  
1 ≤ inputArray[i].length ≤ 15.

* **[output] boolean**

34) extractEachKth (<https://codefights.com/arcade/intro/level-8/3AgqcKrxbwFhd3Z3R>)

Given array of integers, remove each kthelement from it.

**Example**

For inputArray = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] and k = 3, the output should be  
extractEachKth(inputArray, k) = [1, 2, 4, 5, 7, 8, 10].

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] array.integer inputArray**

*Guaranteed constraints:*  
5 ≤ inputArray.length ≤ 15,  
-20 ≤ inputArray[i] ≤ 20.

* **[input] integer k**

*Guaranteed constraints:*  
1 ≤ k ≤ 10.

* **[output] array.integer**

inputArray without elements k - 1, 2k - 1, 3k - 1 etc.

35) firstDigit (<https://codefights.com/arcade/intro/level-8/rRGGbTtwZe2mA8Wov>)

Find the leftmost digit that occurs in a given string.

**Example**

* For inputString = "var\_1\_\_Int", the output should be  
  firstDigit(inputString) = '1';
* For inputString = "q2q-q", the output should be  
  firstDigit(inputString) = '2';
* For inputString = "0ss", the output should be  
  firstDigit(inputString) = '0'.

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] string inputString**

A string containing at least one digit.

*Guaranteed constraints:*  
3 ≤ inputString.length ≤ 10.

* **[output] char**

36) differentSymbolsNaive (<https://codefights.com/arcade/intro/level-8/8N7p3MqzGQg5vFJfZ>)

Given a string, find the number of different characters in it.

**Example**

For s = &quot;cabca&quot;, the output should be  
differentSymbolsNaive(s) = 3.

There are 3 different characters a, b and c.

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] string s**

A string of lowercase English letters.

*Guaranteed constraints:*  
3 ≤ s.length ≤ 1000.

* **[output] integer**

37) arrayMaxConsecutiveSum (<https://codefights.com/arcade/intro/level-8/Rqvw3daffNE7sT7d5>)

Given array of integers, find the maximal possible sum of some of its k consecutive elements.

**Example**

For inputArray = [2, 3, 5, 1, 6] and k = 2, the output should be  
arrayMaxConsecutiveSum(inputArray, k) = 8.  
All possible sums of 2 consecutive elements are:

* 2 + 3 = 5;
* 3 + 5 = 8;
* 5 + 1 = 6;
* 1 + 6 = 7.  
  Thus, the answer is 8.

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] array.integer inputArray**

Array of positive integers.

*Guaranteed constraints:*  
3 ≤ inputArray.length ≤ 105,  
1 ≤ inputArray[i] ≤ 1000.

* **[input] integer k**

An integer (not greater than the length of inputArray).

*Guaranteed constraints:*  
1 ≤ k ≤ inputArray.length.

* **[output] integer**

The maximal possible sum.

38) growingPlant (<https://codefights.com/arcade/intro/level-9/xHvruDnQCx7mYom3T>)

Each day a plant is growing by upSpeed meters. Each night that plant's height decreases by downSpeed meters due to the lack of sun heat. Initially, plant is 0 meters tall. We plant the seed at the beginning of a day. We want to know when the height of the plant will reach a certain level.

**Example**

For upSpeed = 100, downSpeed = 10 and desiredHeight = 910, the output should be  
growingPlant(upSpeed, downSpeed, desiredHeight) = 10.

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] integer upSpeed**

A positive integer representing the daily growth.

*Guaranteed constraints:*  
3 ≤ upSpeed ≤ 100.

* **[input] integer downSpeed**

A positive integer representing the nightly decline.

*Guaranteed constraints:*  
2 ≤ downSpeed < upSpeed.

* **[input] integer desiredHeight**

A positive integer representing the threshold.

*Guaranteed constraints:*  
4 ≤ desiredHeight ≤ 1000.

* **[output] integer**

The number of days that it will take for the plant to reach/pass desiredHeight (including the last day in the total count).

39) knapsackLight (<https://codefights.com/arcade/intro/level-9/r9azLYp2BDZPyzaG2>)

You found two items in a treasure chest! The first item weighs weight1 and is worth value1, and the second item weighs weight2and is worth value2. What is the total maximum value of the items you can take with you, assuming that your max weight capacity is maxW and you can't come back for the items later?

**Note** that there are only two items and you can't bring more than one item of each type, i.e. you can't take two first items or two second items.

**Example**

* For value1 = 10, weight1 = 5, value2 = 6, weight2 = 4 and maxW = 8, the output should be  
  knapsackLight(value1, weight1, value2, weight2, maxW) = 10.

You can only carry the first item.

* For value1 = 10, weight1 = 5, value2 = 6, weight2 = 4 and maxW = 9, the output should be  
  knapsackLight(value1, weight1, value2, weight2, maxW) = 16.

You're strong enough to take both of the items with you.

* For value1 = 5, weight1 = 3, value2 = 7, weight2 = 4 and maxW = 6, the output should be  
  knapsackLight(value1, weight1, value2, weight2, maxW) = 7.

You can't take both items, but you can take any of them.

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] integer value1**

*Guaranteed constraints:*  
2 ≤ value1 ≤ 20.

* **[input] integer weight1**

*Guaranteed constraints:*  
2 ≤ weight1 ≤ 10.

* **[input] integer value2**

*Guaranteed constraints:*  
2 ≤ value2 ≤ 20.

* **[input] integer weight2**

*Guaranteed constraints:*  
2 ≤ weight2 ≤ 10.

* **[input] integer maxW**

*Guaranteed constraints:*  
1 ≤ maxW ≤ 20.

* **[output] integer**

40) longestDigitsPrefix (<https://codefights.com/arcade/intro/level-9/AACpNbZANCkhHWNs3>)

Given a string, output its longest [prefix](keyword://string-prefix) which contains only digits.

**Example**

For inputString="123aa1", the output should be  
longestDigitsPrefix(inputString) = "123".

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] string inputString**

*Guaranteed constraints:*  
3 ≤ inputString.length ≤ 35.

* **[output] string**

41) digitDegree (<https://codefights.com/arcade/intro/level-9/hoLtYWbjdrD2PF6yo>)

Let's define *digit degree* of some positive integer as the number of times we need to replace this number with the sum of its digits until we get to a one digit number.

Given an integer, find its digit degree.

**Example**

* For n = 5, the output should be  
  digitDegree(n) = 0;
* For n = 100, the output should be  
  digitDegree(n) = 1.  
  1 + 0 + 0 = 1.
* For n = 91, the output should be  
  digitDegree(n) = 2.  
  9 + 1 = 10 -> 1 + 0 = 1.

**Input/Output**

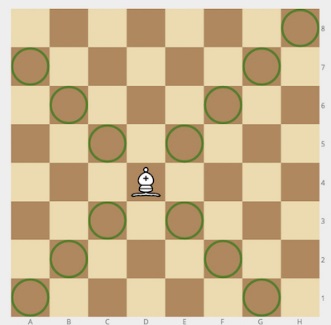
* **[execution time limit] 3 seconds (kt)**
* **[input] integer n**

*Guaranteed constraints:*  
5 ≤ n ≤ 109.

* **[output] integer**

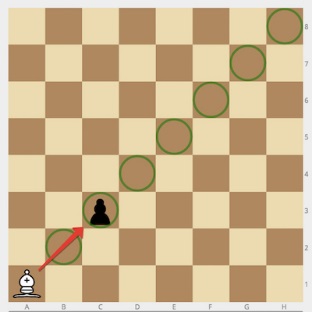
42) bishopAndPawn (<https://codefights.com/arcade/intro/level-9/6M57rMTFB9MeDeSWo>)

Given the positions of a white bishop and a black pawn on the standard chess board, determine whether the bishop can capture the pawn in one move.

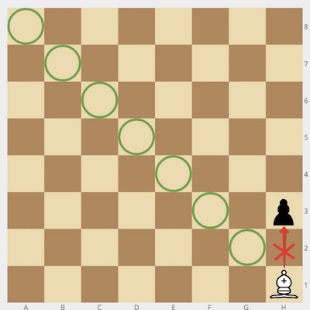
The bishop has no restrictions in distance for each move, but is limited to diagonal movement. Check out the example below to see how it can move:  


**Example**

* For bishop = "a1" and pawn = "c3", the output should be  
  bishopAndPawn(bishop, pawn) = true.



* For bishop = "h1" and pawn = "h3", the output should be  
  bishopAndPawn(bishop, pawn) = false.



**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] string bishop**

Coordinates of the white bishop in the [chess notation](keyword://chess-notation).

* **[input] string pawn**

Coordinates of the black pawn in the same notation.

* **[output] boolean**

true if the bishop can capture the pawn, false otherwise.

43) isBeautifulString (<https://codefights.com/arcade/intro/level-10/PHSQhLEw3K2CmhhXE>)

A string is said to be *beautiful* if boccurs in it no more times than a; coccurs in it no more times than b; etc.

Given a string, check whether it is *beautiful*.

**Example**

* For inputString = "bbbaacdafe", the output should be  
  isBeautifulString(inputString) = true;
* For inputString = "aabbb", the output should be  
  isBeautifulString(inputString) = false;
* For inputString = "bbc", the output should be  
  isBeautifulString(inputString) = false.

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] string inputString**

A string of lowercase letters.

*Guaranteed constraints:*  
3 ≤ inputString.length ≤ 50.

* **[output] boolean**

44) findEmailDomain (<https://codefights.com/arcade/intro/level-10/TXFLopNcCNbJLQivP>)

An email address such as "John.Smith@example.com" is made up of a local part ("John.Smith"), an "@" symbol, then a domain part ("example.com").

The domain name part of an email address may only consist of letters, digits, hyphens and dots. The local part, however, also allows a lot of different special characters. [Here](https://en.wikipedia.org/wiki/Email_address#Examples) you can look at several examples of correct and incorrect email addresses.

Given a valid email address, find its domain part.

**Example**

* For address = "prettyandsimple@example.com", the output should be  
  findEmailDomain(address) = "example.com";
* For address = "<>[]:,;@\"!#$%&\*+-/=?^\_{}| ~.a\"@example.org", the output should be  
  findEmailDomain(address) = "example.org".

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] string address**

*Guaranteed constraints:*  
10 ≤ address.length ≤ 50.

* **[output] string**

45) buildPalindrome (<https://codefights.com/arcade/intro/level-10/ppZ9zSufpjyzAsSEx>)

Given a string, find the shortest possible string which can be achieved by adding characters to the end of initial string to make it a [palindrome](keyword://palindrome).

**Example**

For st = "abcdc", the output should be  
buildPalindrome(st) = "abcdcba".

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] string st**

A string consisting of lowercase latin letters.

*Guaranteed constraints:*  
3 ≤ st.length ≤ 10.

* **[output] string**

46) electionsWinners (<https://codefights.com/arcade/intro/level-10/8RiRRM3yvbuAd3MNg>)

Elections are in progress!

Given an array of the numbers of votes given to each of the candidates so far, and an integer k equal to the number of voters who haven't cast their vote yet, find the number of candidates who still have a chance to win the election.

The winner of the election must secure strictly more votes than any other candidate. If two or more candidates receive the same (maximum) number of votes, assume there is no winner at all.

**Example**

For votes = [2, 3, 5, 2] and k = 3, the output should be  
electionsWinners(votes, k) = 2.

* The first candidate got 2 votes. Even if all of the remaining 3candidates vote for him, he will still have only 5 votes, i.e. the same number as the third candidate, so there will be no winner.
* The second candidate can win if all the remaining candidates vote for him (3 + 3 = 6 > 5).
* The third candidate can win even if none of the remaining candidates vote for him. For example, if each of the remaining voters cast their votes for each of his opponents, he will still be the winner (the votes array will thus be [3, 4, 5, 3]).
* The last candidate can't win no matter what (for the same reason as the first candidate).

Thus, only 2 candidates can win (the second and the third), which is the answer.

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] array.integer votes**

A non-empty array of non-negative integers. Its ith element denotes the number of votes cast for the ith candidate.

*Guaranteed constraints:*  
4 ≤ votes.length ≤ 105,  
0 ≤ votes[i] ≤ 104.

* **[input] integer k**

The number of voters who haven't cast their vote yet.

*Guaranteed constraints:*  
0 ≤ k ≤ 105.

* **[output] integer**

47) isMAC48Address (<https://codefights.com/arcade/intro/level-10/HJ2thsvjL25iCvvdm>)

A media access control address (MAC address) is a unique identifier assigned to network interfaces for communications on the physical network segment.

The standard (IEEE 802) format for printing MAC-48 addresses in human-friendly form is six groups of two hexadecimal digits (0 to 9or A to F), separated by hyphens (e.g. 01-23-45-67-89-AB).

Your task is to check by given string inputString whether it corresponds to MAC-48 address or not.

**Example**

* For inputString = "00-1B-63-84-45-E6", the output should be  
  isMAC48Address(inputString) = true;
* For inputString = "Z1-1B-63-84-45-E6", the output should be  
  isMAC48Address(inputString) = false;
* For inputString = "not a MAC-48 address", the output should be  
  isMAC48Address(inputString) = false.

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] string inputString**

*Guaranteed constraints:*  
15 ≤ inputString.length ≤ 20.

* **[output] boolean**

true if inputStringcorresponds to MAC-48 address naming rules, falseotherwise.

48) isDigit (<https://codefights.com/arcade/intro/level-11/Zr2XXEpkBPtYWqPJu>)

Determine if the given character is a digit or not.

**Example**

* For symbol = '0', the output should be  
  isDigit(symbol) = true;
* For symbol = '-', the output should be  
  isDigit(symbol) = false.

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] char symbol**

A character which is either a digit or not.

* **[output] boolean**

true if symbol is a digit, false otherwise.

49) lineEncoding (<https://codefights.com/arcade/intro/level-11/o2uq6eTuvk7atGadR>)

Given a string, return its encoding defined as follows:

* First, the string is divided into the least possible number of disjoint [substrings](keyword://substring) consisting of identical characters
  + for example, "aabbbc" is divided into ["aa", "bbb", "c"]
* Next, each *substring* with length greater than one is replaced with a concatenation of its length and the repeating character
  + for example, *substring* "bbb" is replaced by "3b"
* Finally, all the new strings are concatenated together in the same order and a new string is returned.

**Example**

For s = "aabbbc", the output should be  
lineEncoding(s) = "2a3bc".

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] string s**

String consisting of lowercase English letters.

*Guaranteed constraints:*  
4 ≤ s.length ≤ 15.

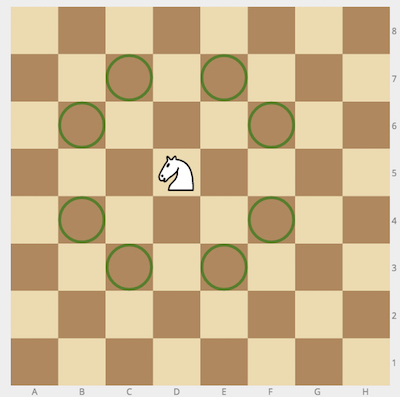
* **[output] string**

Encoded version of s.

50) chessKnight (<https://codefights.com/arcade/intro/level-11/pwRLrkrNpnsbgMndb>)

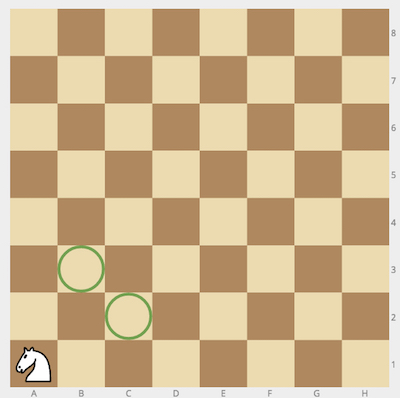
Given a position of a knight on the standard chessboard, find the number of different moves the knight can perform.

The knight can move to a square that is two squares horizontally and one square vertically, or two squares vertically and one square horizontally away from it. The complete move therefore looks like the letter L. Check out the image below to see all valid moves for a knight piece that is placed on one of the central squares.

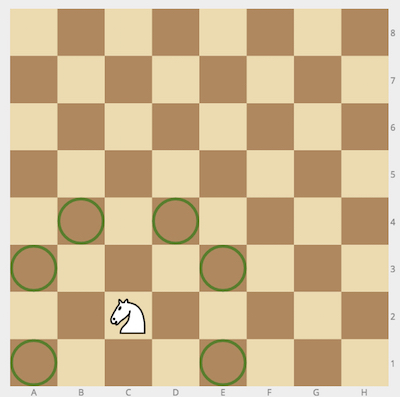


**Example**

* For cell = "a1", the output should be  
  chessKnight(cell) = 2.



* For cell = "c2", the output should be  
  chessKnight(cell) = 6.



**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] string cell**

String consisting of 2 letters - coordinates of the knight on an 8 × 8 chessboard in [chess notation](keyword://chess-notation).

* **[output] integer**

51) deleteDigit (<https://codefights.com/arcade/intro/level-11/vpfeqDwGZSzYNm2uX>)

Given some integer, find the maximal number you can obtain by deleting exactly one digit of the given number.

**Example**

* For n = 152, the output should be  
  deleteDigit(n) = 52;
* For n = 1001, the output should be  
  deleteDigit(n) = 101.

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] integer n**

*Guaranteed constraints:*  
10 ≤ n ≤ 106.

* **[output] integer**

52) longestWord (<https://codefights.com/arcade/intro/level-12/s9qvXv4yTaWg8g4ma>)

Define a *word* as a sequence of consecutive English letters. Find the longest *word* from the given string.

**Example**

For text = "Ready, steady, go!", the output should be  
longestWord(text) = "steady".

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] string text**

*Guaranteed constraints:*  
4 ≤ text.length ≤ 50.

* **[output] string**

The longest *word* from text. It's guaranteed that there is a unique output.

53) validTime (<https://codefights.com/arcade/intro/level-12/ywMyCTspqGXPWRZx5>)

Check if the given string is a correct time representation of the 24-hour clock.

**Example**

* For time = "13:58", the output should be  
  validTime(time) = true;
* For time = "25:51", the output should be  
  validTime(time) = false;
* For time = "02:76", the output should be  
  validTime(time) = false.

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] string time**

A string representing time in HH:MM format. It is guaranteed that the first two characters, as well as the last two characters, are digits.

* **[output] boolean**

true if the given representation is correct, false otherwise.

54) sumUpNumbers (<https://codefights.com/arcade/intro/level-12/YqZwMJguZBY7Hz84T>)

CodeMaster has just returned from shopping. He scanned the check of the items he bought and gave the resulting string to Ratiorg to figure out the total number of purchased items. Since Ratiorg is a bot he is definitely going to automate it, so he needs a program that sums up all the numbers which appear in the given input.

Help Ratiorg by writing a function that returns the sum of numbers that appear in the given inputString.

**Example**

For inputString = "2 apples, 12 oranges", the output should be  
sumUpNumbers(inputString) = 14.

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] string inputString**

*Guaranteed constraints:*  
6 ≤ inputString.length ≤ 60.

* **[output] integer**

55) differentSquares (<https://codefights.com/arcade/intro/level-12/fQpfgxiY6aGiGHLtv>)

Given a rectangular matrix containing only digits, calculate the number of different 2 × 2 squares in it.

**Example**

For

matrix = [[1, 2, 1],

[2, 2, 2],

[2, 2, 2],

[1, 2, 3],

[2, 2, 1]]

the output should be  
differentSquares(matrix) = 6.

Here are all 6 different 2 × 2 squares:

* 1 2  
  2 2
* 2 1  
  2 2
* 2 2  
  2 2
* 2 2  
  1 2
* 2 2  
  2 3
* 2 3  
  2 1

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] array.array.integer matrix**

*Guaranteed constraints:*  
1 ≤ matrix.length ≤ 100,  
1 ≤ matrix[i].length ≤ 100,  
0 ≤ matrix[i][j] ≤ 9.

* **[output] integer**

The number of different 2 × 2 squares in matrix.

56) digitsProduct (<https://codefights.com/arcade/intro/level-12/NJJhENpgheFRQbPRA>)

Given an integer product, find the *smallest* **positive** (i.e. greater than 0) integer the product of whose digits is equal to product. If there is no such integer, return -1 instead.

**Example**

* For product = 12, the output should be  
  digitsProduct(product) = 26;
* For product = 19, the output should be  
  digitsProduct(product) = -1.

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] integer product**

*Guaranteed constraints:*  
0 ≤ product ≤ 600.

* **[output] integer**

57) fileNaming (<https://codefights.com/arcade/intro/level-12/sqZ9qDTFHXBNrQeLC>)

You are given an array of desired filenames in the order of their creation. Since two files cannot have equal names, the one which comes later will have an addition to its name in a form of (k), where k is the smallest positive integer such that the obtained name is not used yet.

Return an array of names that will be given to the files.

**Example**

For names = ["doc", "doc", "image", "doc(1)", "doc"], the output should be  
fileNaming(names) = ["doc", "doc(1)", "image", "doc(1)(1)", "doc(2)"].

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] array.string names**

*Guaranteed constraints:*  
5 ≤ names.length ≤ 15,  
1 ≤ names[i].length ≤ 15.

* **[output] array.string**

58) messageFromBinaryCode (<https://codefights.com/arcade/intro/level-12/sCpwzJCyBy2tDSxKW>)

You are taking part in an Escape Room challenge designed specifically for programmers. In your efforts to find a clue, you've found a binary code written on the wall behind a vase, and realized that it must be an encrypted message. After some thought, your first guess is that each consecutive 8 bits of the code stand for the character with the corresponding [extended ASCII code](http://www.ascii-code.com/).

Assuming that your hunch is correct, decode the message.

**Example**

For code = "010010000110010101101100011011000110111100100001", the output should be  
messageFromBinaryCode(code) = "Hello!".

The first 8 characters of the code are 01001000, which is 72 in the binary numeral system. 72 stands for H in the *ASCII-table*, so the first letter is H.  
Other letters can be obtained in the same manner.

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] string code**

A string, the encrypted message consisting of characters '0'and '1'.

*Guaranteed constraints:*  
0 < code.length < 800.

* **[output] string**

The decrypted message.

59) spiralNumbers (<https://codefights.com/arcade/intro/level-12/uRWu6K8E7uLi3Qtvx>)

Construct a square matrix with a size N × N containing integers from 1 to N \* N in a spiral order, starting from top-left and in clockwise direction.

**Example**

For n = 3, the output should be

spiralNumbers(n) = [[1, 2, 3],

[8, 9, 4],

[7, 6, 5]]

**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] integer n**

Matrix size, a positive integer.

*Guaranteed constraints:*  
3 ≤ n ≤ 100.

* **[output] array.array.integer**

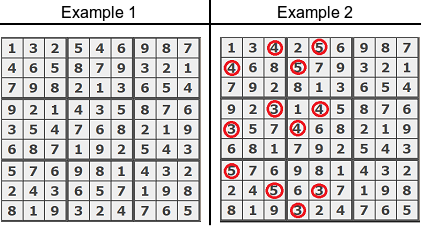
60) sudoku (<https://codefights.com/arcade/intro/level-12/tQgasP8b62JBeirMS>)

*Sudoku* is a number-placement puzzle. The objective is to fill a 9 × 9grid with digits so that each column, each row, and each of the nine 3 × 3 sub-grids that compose the grid contains all of the digits from 1to 9.

This algorithm should check if the given grid of numbers represents a correct solution to Sudoku.

**Example**

For the first example below, the output should be true. For the other grid, the output should be false: each of the nine 3 × 3 sub-grids should contain all of the digits from 1 to 9.



**Input/Output**

* **[execution time limit] 3 seconds (kt)**
* **[input] array.array.integer grid**

A matrix representing 9 × 9 grid already filled with numbers from 1 to 9.

* **[output] boolean**

true if the given grid represents a correct solution to Sudoku, false otherwise.