C. Number of Ways

time limit per test: 2 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

You've got array *a*[1], *a*[2], ..., *a*[*n*], consisting of *n* integers. Count the number of ways to split all the elements of the array into three contiguous parts so that the sum of elements in each part is the same.

More formally, you need to find the number of such pairs of indices *i*, *j* (2 ≤ *i* ≤ *j* ≤ *n* - 1), that https://espresso.codeforces.com/669a2f09a3b9e143f54b1f1d9fd6b7dddf403680.png.

**Input**

The first line contains integer *n* (1 ≤ *n* ≤ 5·105), showing how many numbers are in the array. The second line contains *n* integers *a*[1], *a*[2], ..., *a*[*n*] (|*a*[*i*]| ≤  109) — the elements of array *a*.

**Output**

Print a single integer — the number of ways to split the array into three parts with the same sum.

A. Turtle and Piggy Are Playing a Game

time limit per test: 1 second

memory limit per test: 256 megabytes

input: standard input

output: standard output

Turtle and Piggy are playing a number game.

First, Turtle will choose an integer x𝑥, such that l≤x≤r𝑙≤𝑥≤𝑟, where l,r𝑙,𝑟 are given. It's also guaranteed that 2l≤r2𝑙≤𝑟.

Then, Piggy will keep doing the following operation until x𝑥 becomes 11:

* Choose an integer p𝑝 such that p≥2𝑝≥2 and p∣x𝑝∣𝑥 (i.e. x𝑥 is a multiple of p𝑝).
* Set x𝑥 to xp𝑥𝑝, and the score will increase by 11.

The score is initially 00. Both Turtle and Piggy want to maximize the score. Please help them to calculate the maximum score.

**Input**

Each test contains multiple test cases. The first line contains the number of test cases t𝑡 (1≤t≤1041≤𝑡≤104). The description of the test cases follows.

The first line of each test case contains two integers l,r𝑙,𝑟 (1≤l≤r≤109,2l≤r1≤𝑙≤𝑟≤109,2𝑙≤𝑟) — The range where Turtle can choose the integer from.

**Output**

For each test case, output a single integer — the maximum score.

A. Adding Digits

time limit per test: 2 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

Vasya has got two number: *a* and *b*. However, Vasya finds number *a* too short. So, he decided to repeat the operation of lengthening number *a* *n* times.

One operation of lengthening a number means adding exactly one digit to the number (in the decimal notation) to the right provided that the resulting number is divisible by Vasya's number *b*. If it is impossible to obtain the number which is divisible by *b*, then the lengthening operation cannot be performed.

Your task is to help Vasya and print the number he can get after applying the lengthening operation to number *a* *n* times.

**Input**

The first line contains three integers: *a*, *b*, *n* (1 ≤ *a*, *b*, *n* ≤ 105).

**Output**

In a single line print the integer without leading zeros, which Vasya can get when he applies the lengthening operations to number *a* *n* time. If no such number exists, then print number -1. If there are multiple possible answers, print any of them.

A. Sereja and Bottles

time limit per test: 2 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

Sereja and his friends went to a picnic. The guys had *n* soda bottles just for it. Sereja forgot the bottle opener as usual, so the guys had to come up with another way to open bottles.

Sereja knows that the *i*-th bottle is from brand *ai*, besides, you can use it to open **other** bottles of brand *bi*. You can use one bottle to open multiple other bottles. Sereja can open bottle with opened bottle or closed bottle.

Knowing this, Sereja wants to find out the number of bottles they've got that they won't be able to open in any way. Help him and find this number.

**Input**

The first line contains integer *n* (1 ≤ *n* ≤ 100) — the number of bottles. The next *n* lines contain the bottles' description. The *i*-th line contains two integers *ai*, *bi* (1 ≤ *ai*, *bi* ≤ 1000) — the description of the *i*-th bottle.

**Output**

In a single line print a single integer — the answer to the problem.

C. Exams

time limit per test: 1 second

memory limit per test: 256 megabytes

Input: standard input

output: standard output

Student Valera is an undergraduate student at the University. His end of term exams are approaching and he is to pass exactly *n* exams. Valera is a smart guy, so he will be able to pass any exam he takes on his first try. Besides, he can take several exams on one day, and in any order.

According to the schedule, a student can take the exam for the *i*-th subject on the day number *ai*. However, Valera has made an arrangement with each teacher and the teacher of the *i*-th subject allowed him to take an exam before the schedule time on day *bi* (*bi* < *ai*). Thus, Valera can take an exam for the *i*-th subject either on day *ai*, or on day *bi*. All the teachers put the record of the exam in the student's record book on the day of the actual exam and write down the date of the mark as number *ai*.Valera believes that it would be rather strange if the entries in the record book did not go in the order of non-decreasing date. Therefore Valera asks you to help him. Find the minimum possible value of the day when Valera can take the final exam if he takes exams so that all the records in his record book go in the order of non-decreasing date.

**Input**

The first line contains a single positive integer *n* (1 ≤ *n* ≤ 5000) — the number of exams Valera will take.

Each of the next *n* lines contains two positive space-separated integers *ai* and *bi* (1 ≤ *bi* < *ai* ≤ 109) — the date of the exam in the schedule and the early date of passing the *i*-th exam,correspondingly.

**Output**

Print a single integer — the minimum possible number of the day when Valera can take the last exam if he takes all the exams so that all the records in his record book go in the order of non-decreasing date.

A. Boredom

time limit per test: 1 second

memory limit per test: 256 megabytes

Input: standard input

output: standard output

Alex doesn't like boredom. That's why whenever he gets bored, he comes up with games. One long winter evening he came up with a game and decided to play it.

Given a sequence *a* consisting of *n* integers. The player can make several steps. In a single step he can choose an element of the sequence (let's denote it *ak*) and delete it, at that all elements equal to *ak* + 1 and *ak* - 1 also must be deleted from the sequence. That step brings *ak* points to the player.Alex is a perfectionist, so he decided to get as many points as possible. Help him.

**Input**

The first line contains integer *n* (1 ≤ *n* ≤ 105) that shows how many numbers are in Alex's sequence.

The second line contains *n* integers *a*1, *a*2, ..., *an* (1 ≤ *ai* ≤ 105).

**Output**

Print a single integer — the maximum number of points that Alex can earn.

C. Table Decorations

time limit per test: 1 second

memory limit per test: 256 megabytes

Input: standard input

output: standard output

You have *r* red, *g* green and *b* blue balloons. To decorate a single table for the banquet you need exactly three balloons. Three balloons attached to some table shouldn't have the same color. What maximum number *t* of tables can be decorated if we know number of balloons of each color?

Your task is to write a program that for given values *r*, *g* and *b* will find the maximum number *t* of tables, that can be decorated in the required manner.

**Input**

The single line contains three integers *r*, *g* and *b* (0 ≤ *r*, *g*, *b* ≤ 2·109) — the number of red, green and blue baloons respectively. The numbers are separated by exactly one space.

**Output**

Print a single integer *t* — the maximum number of tables that can be decorated in the required manner.

C. Beautiful Sets of Points

time limit per test: 1 second

memory limit per test: 256 megabytes

Input: standard input

output: standard output

Manao has invented a new mathematical term — a beautiful set of points. He calls a set of points on a plane *beautiful* if it meets the following conditions:

1. The coordinates of each point in the set are integers.
2. For any two points from the set, the distance between them is a non-integer.

Consider all points (*x*, *y*) which satisfy the inequations: 0 ≤ *x* ≤ *n*; 0 ≤ *y* ≤ *m*; *x* + *y* > 0. Choose their subset of maximum size such that it is also a beautiful set of points.

**Input**

The single line contains two space-separated integers *n* and *m* (1 ≤ *n*, *m* ≤ 100).

**Output**

In the first line print a single integer — the size *k* of the found beautiful set. In each of the next *k* lines print a pair of space-separated integers — the *x*- and *y*- coordinates, respectively, of a point from the set.If there are several optimal solutions, you may print any of them.

C. Valera and Tubes

time limit per test: 1 second

memory limit per test: 256 megabytes

Input: standard input

output: standard output

Valera has got a rectangle table consisting of *n* rows and *m* columns. Valera numbered the table rows starting from one, from top to bottom and the columns – starting from one, from left to right. We will represent cell that is on the intersection of row *x* and column *y* by a pair of integers (*x*, *y*).

Valera wants to place exactly *k* tubes on his rectangle table. A tube is such sequence of table cells (*x*1, *y*1), (*x*2, *y*2), ..., (*xr*, *yr*), that:

* *r* ≥ 2;
* for any integer *i* (1 ≤ *i* ≤ *r* - 1) the following equation |*xi* - *xi*+ 1| + |*yi* - *yi*+ 1| = 1 holds;
* each table cell, which belongs to the tube, must occur exactly once in the sequence.

Valera thinks that the tubes are arranged in a fancy manner if the following conditions are fulfilled:

* no pair of tubes has common cells;
* each cell of the table belongs to some tube.

Help Valera to arrange *k* tubes on his rectangle table in a fancy manner.

**Input**

The first line contains three space-separated integers *n*, *m*, *k* (2 ≤ *n*, *m* ≤ 300; 2 ≤ 2*k* ≤ *n*·*m*) — the number of rows, the number of columns and the number of tubes, correspondingly.

**Output**

Print *k* lines. In the *i*-th line print the description of the *i*-th tube: first print integer *ri* (the number of tube cells), then print 2*ri* integers *xi*1, *yi*1, *xi*2, *yi*2, ..., *xiri*, *yiri* (the sequence of table cells).

If there are multiple solutions, you can print any of them. It is guaranteed that at least one solution exists.