TASK	TIMSKO	PROFESOR	SRETAN	LJUTNJA	TABOVI	ŽABE
input	standard input (*stdin*)					
output	standard output (*stdout*)					
time limit	1 second	1 second	1 second	1 second	1 second	1 second
memory limit	32 MB	32 MB	32 MB	32 MB	32 MB	64 MB
total points	**30**	**40**	**70**	**100**	**120**	**140**
	500					

Every year, the University of Zagreb organizes a student team competition in informatics. Each team consists of **three** students.

Traditionally, the best competitors from the university are girls, and they outnumber boys significantly. This year, boys have raised their voice and a rule was made that each team must consist of **exactly** one boy and two girls.

To make competitors' lives a little more difficult, the dean of the university has decided to send **K** of the competitors on an internship in a distant country. Those competitors will not be able to compete.

Given the number of female competitors **M**, the number of male competitors **N**, and the number of competitors which have to be sent on an internship **K**, the dean has to create the maximum number of teams which will be able to attend the competition.

For example, if **M** is 6, **N** is 3 and **K** is 2, the dean can send one girl and one boy on an internship, which leaves him with 5 girls and 2 boys. He can then create two teams from them (one girl is left without a team).

INPUT

The first and only line of input contains three integers separated by single spaces: **M** ($0 \leq \mathbf{M} \leq 100$), the number of girls, **N** ($0 \leq \mathbf{N} \leq 100$), the number of boys, and **K** ($0 \leq \mathbf{K} \leq \mathbf{M+N}$), the number of competitors which have to be sent on an internship.

OUTPUT

The first and only line of output must contain only one number: the maximum number of teams which can be formed.

SAMPLE TESTS

input	**input**	**input**
6 3 2	2 1 1	6 10 3
output	**output**	**output**
2	0	3

In a long classroom, **N** desks are arranged in a single row, with two students sitting at each desk. Students are cranky because they are about to have an art class, and their professor is planning to examine them.

Each student has studied art, but only to a certain level. The old professor can tell by the looks on their faces just how much they have studied. The professor, being an artist, uses a different coloured pencil for each grade. Unfortunately, today he brought only one pencil.

In order to make the examination seem fair, he wants to choose two desks and question one student from **each desk** positioned between the two desks he has chosen (including the chosen desks). It is important that **all examined students deserve the same grades**, so he can write them down using his only pencil.

The professor wants to know the maximum number of students he can examine this way, as well as which grade the students will get.

INPUT

The first line of input contains a single integer **N** ($1 \leq \mathbf{N} \leq 100\,000$).

Each of the following **N** rows contains two integers: $\mathbf{A_i}$ and $\mathbf{B_i}$, grades deserved by students sitting at desk **i** ($1 \leq \mathbf{A_i}, \mathbf{B_i} \leq 5$).

OUTPUT

The first and only line of output must contain two numbers separated by a single space: the maximum number of students the professor can examine and the grade those students will get.

If there are multiple solutions possible, output the one with the smallest grade.

SCORING

Test cases worth 70% of total points have $\mathbf{N} \leq 100$.

SAMPLE TESTS

input	input	input
1 1 5 output 1 1	3 3 5 4 5 1 3 output 2 5	4 2 1 3 2 5 3 2 5 output 2 2

Digits 4 and 7 are lucky, while all others are unlucky. An integer is lucky if it contains only lucky digits in decimal notation. We would like to know the **K**-th lucky positive integer.

INPUT

The first and only line of input contains a positive integer **K** ($1 \leq \mathbf{K} \leq 10^9$).

OUTPUT

The first and only line of output must contain the **K**-th lucky positive integer.

SAMPLE TESTS

input	input	input
1	2	3
output	output	output
4	7	44

Children in a kindergarten have received a large sack containing **M** candies. It has been decided that the candies are to be distributed among **N** children.

Each child has stated the number of candies that it wants. If a child isn't given the amount of candy it wants, it will get angry. In fact it'll get angrier for each candy it is deprived of. Some speculate that it's anger will be equal to the square of the number of candy it is deprived of. For instance, if Mirko states that he wants 32 candies but receives only 29, he would be missing 3 candies, so his anger would be equal to 9.

Unfortunately, there is an insufficient amount of candy to satisfy all children. Therefore, the candies should be distributed in such a way that the sum of the children's anger is minimal.

INPUT

The first line contains two integers, **M** ($1 \leq \mathbf{M} \leq 2 \cdot 10^9$) and **N** ($1 \leq \mathbf{N} \leq 100\,000$).

The following **N** lines contain integers (one per line) which represent the wishes of the children. Those numbers are all strictly less than $2 \cdot 10^9$, and their sum always exceeds **M**.

OUTPUT

The first and only line of output must contain the minimum sum of the children's anger.

Note: The test cases will ensure that the result fits in a 64-bit unsigned integer: *int64* in Pascal, *long long* in C/C++, *long* in Java.

SCORING

Test cases worth 40% of total points have **M** not greater than 200 000.

Test cases worth 70% of total points have no child state that it wants more than 100 000 candies.

Test cases worth 80% of total points have at least one of the above stated constraints will be met.

SAMPLE TESTS

input	**input**
5 3 1 3 2 **output** 1	10 4 4 5 2 3 **output** 4

Zvonkec is yet another programmer employed in a small company. Every day he has to refactor one file of source code. Much to his dismay, the source is usually far from being clear and tidy. He is especially bothered by uneven indentation, i.e. the number of tabulators (tabs) indenting each line. Fortunately, his editor has a command to select a group of consecutive lines and add or delete a character from the start of each one. Help Zvonkec tidy up the code as quickly as possible.

You are given the number of lines **N**, a sequence specifying the current number of tabs at the start of each line, and a sequence specifying the required number of tabs at the start of each line.

Zvonkec can execute any number of commands consisting of:

- selecting any number of consecutive lines
- adding or deleting a **single** tab to/from the front of each of the selected lines

The two actions above comprise **a single command**, regardless of the number of selected lines.

It should be noted that it is forbidden to delete more tabs from a line than are actually present at the start of a line, as the editor would start deleting characters other than tabs.

You are asked to calculate the **minimum number of commands** required to tidy up the code.

INPUT

The first line of input contains a positive integer **N** ($\mathbf{N} \leq 1000$).

The second line contains a sequence of **N** integers $\mathbf{P_i}$ ($0 \leq \mathbf{P_i} \leq 80$), specifying the number of tabs at the start of i-th line before any editing.

The third line contains a sequence of **N** integers $\mathbf{K_i}$ ($0 \leq \mathbf{K_i} \leq 80$), specifying the number of tabs that Zvonkec would like at the start of i-th line.

OUTPUT

The first and only line of output must contain the required number, as specified in the problem statement.

SCORING

Test cases worth 70% of total points have **N** not greater than 100.

SAMPLE TESTS

input	**input**	**input**
3 3 4 5 6 7 8 **output** 3	4 1 2 3 4 3 1 1 0 **output** 6	4 5 4 5 5 1 5 0 1 **output** 10

The Frog Regent has arranged his **N** frog servants in a circle, with each frog facing the back of the next one. Each frog is assigned a unique integer identifier (ID) from the set of 1 to **N**. The frog arrangement is specified as a sequence of IDs. The sequence **always starts** with the frog with the **ID 1**. It is followed by the ID of the frog in front of it, then the ID of the next one, and so on until the ID of the last frog - the one *behind* the frog with ID 1.

A frog is considered to have made a single leap if it has jumped over the frog in front of it, swapping places with it in the process. For example, if the frogs are sequenced as "1 5 4 3 2 6" and the frog with ID 2 makes two leaps, the resulting sequence will be "1 2 5 4 3 6" (the frog has shifted two places forward). When the Frog Regent proclaims the number **B**, the frog with ID **B** makes **B** leaps.

The Frog Regent wishes, using some number of proclamations, to rearrange the frogs from the starting sequence to his favourite resulting sequence. Given the starting and resulting frog sequences, write a program that will compute a sequence of proclamations needed for the Regent to rearrange the frogs into the resulting sequence. Naturally, the starting and resulting sequences will not be equal.

INPUT

The first line of input contains a positive integer **N**, the number of frogs ($3 \leq \mathbf{N} \leq 100$).

The second line of input contains a permutation of the first **N** positive integers, the starting frog sequence.

The third line of input contains another permutation of the first **N** positive integers, the resulting frog sequence.

OUTPUT

Output any sequence of integers (one integer per line) that the Frog Regent can proclaim in order to rearrange the frogs into the resulting sequence.

The number of proclamations must not exceed **100 000**.

Note: The test data will ensure that a solution will always exist.

SCORING

Test cases worth 40% of total points have **N** not greater than 8.

SAMPLE TESTS

input	**input**
6 1 5 4 3 2 6 1 2 5 4 3 6	5 1 5 3 2 4 1 5 4 2 3
output	**output**
2	5 3 5 2