A. Bit++

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

The classic programming language of Bitland is Bit++. This language is so peculiar and complicated.

The language is that peculiar as it has exactly one variable, called x. Also, there are two operations:

- Operation ++ increases the value of variable x by 1.
- Operation -- decreases the value of variable x by 1.

A statement in language Bit++ is a sequence, consisting of exactly one operation and one variable x. The statement is written without spaces, that is, it can only contain characters "+", "-", "X". Executing a statement means applying the operation it contains.

A programme in Bit++ is a sequence of statements, each of them needs to be executed. Executing a programme means executing all the statements it contains.

You're given a programme in language Bit++. The initial value of x is 0. Execute the programme and find its final value (the value of the variable when this programme is executed).

Input

The first line contains a single integer n $(1 \le n \le 150)$ — the number of statements in the programme.

Next n lines contain a statement each. Each statement contains exactly one operation (++ or --) and exactly one variable x (denoted as letter «X»). Thus, there are no empty statements. The operation and the variable can be written in any order.

Output

Print a single integer — the final value of x.

Examples

input

```
1
++X
```

output

```
1
```

input

```
2
X++
--X
```

output

```
0
```

B. Painting Eggs

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The Bitlandians are quite weird people. They have very peculiar customs.

As is customary, Uncle J. wants to have n eggs painted for Bitruz (an ancient Bitland festival). He has asked G. and A. to do the work.

The kids are excited because just as is customary, they're going to be paid for the job!

Overall uncle J. has got n eggs. G. named his price for painting each egg. Similarly, A. named his price for painting each egg. It turns out that for each egg the sum of the money both A. and G. want for the painting equals 1000.

Uncle J. wants to distribute the eggs between the children so as to give each egg to exactly one child. Also, Uncle J. wants the total money paid to A. to be different from the total money paid to G. by no more than 500.

Help Uncle J. Find the required distribution of eggs or otherwise say that distributing the eggs in the required manner is impossible.

Input

The first line contains integer n $(1 \le n \le 10^6)$ — the number of eggs.

Next n lines contain two integers a_i and g_i each $(0 \le a_i, g_i \le 1000;\ a_i + g_i = 1000)$: a_i is the price said by A. for the i-th egg and g_i is the price said by G. for the i-th egg.

Output

If it is impossible to assign the painting, print "`-1`" (without quotes).

Otherwise print a string, consisting of n letters "`G`" and "`A`". The i-th letter of this string should represent the child who will get the i-th egg in the required distribution. Letter "`A`" represents A. and letter "`G`" represents G. If we denote the money Uncle J. must pay A. for the painting as S_a, and the money Uncle J. must pay G. for the painting as S_g, then this inequality must hold: $|S_a - S_g| \le 500$.

If there are several solutions, you are allowed to print any of them.

Examples

input

```
2
1 999
999 1
```

output

```
AG
```

input

```
3
400 600
400 600
400 600
```

output

```
AGA
```

C. XOR and OR

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The Bitlandians are quite weird people. They do everything differently. They have a different alphabet so they have a different definition for a string.

A Bitlandish string is a string made only of characters "`0`" and "`1`".

BitHaval (the mayor of Bitland) loves to play with Bitlandish strings. He takes some Bitlandish string a, and applies several (possibly zero) operations to it. In one operation the mayor may take any two adjacent characters of a string, define one of them as x and the other one as y. Then he calculates two values p and q: $p = x\ xor\ y$, $q = x\ or\ y$. Then he replaces one of the two taken characters by p and the other one by q.

The xor operation means the bitwise excluding OR operation. The or operation is the bitwise OR operation.

So for example one operation can transform string `11` to string `10` or to string `01`. String `1` cannot be transformed into any other string.

You've got two Bitlandish strings a and b. Your task is to check if it is possible for BitHaval to transform string a to string b in several (possibly zero) described operations.

Input

The first line contains Bitlandish string a, the second line contains Bitlandish string b. The strings can have different lengths.

It is guaranteed that the given strings only consist of characters "0" and "1". The strings are not empty, their length doesn't exceed 10^6.

Output

Print "YES" if a can be transformed into b, otherwise print "`NO`". Please do not print the quotes.

Examples

input

```
11
10
```

output

```
YES
```

input

```
1
01
```

output

```
NO
```

input

```
000
101
```

output

```
NO
```

D. Yet Another Number Game

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Since most contestants do not read this part, I have to repeat that Bitlandians are quite weird. They have their own jobs, their own working method, their own lives, their own sausages and their own games!

Since you are so curious about Bitland, I'll give you the chance of peeking at one of these games.

BitLGM and BitAryo are playing yet another of their crazy-looking genius-needed Bitlandish games. They've got a sequence of n non-negative integers $a_1, a_2, ..., a_n$. The players make moves in turns. BitLGM moves first. Each player can and must do one of the two following actions in his turn:

- Take one of the integers (we'll denote it as a_i). Choose integer x $(1 \le x \le a_i)$. And then decrease a_i by x, that is, apply assignment: $a_i = a_i - x$.
- Choose integer x $(1 \le x \le \min_{i=1}^{n} a_i)$. And then decrease all a_i by x, that is, apply assignment: $a_i = a_i - x$, for all i.

The player who cannot make a move loses.

You're given the initial sequence $a_1, a_2, ..., a_n$. Determine who wins, if both players plays optimally well and if BitLGM and BitAryo start playing the described game in this sequence.

Input

The first line contains an integer n $(1 \le n \le 3)$.

The next line contains n integers $a_1, a_2, ..., a_n$ $(0 \le a_i < 300)$.

Output

Write the name of the winner (provided that both players play optimally well). Either "`BitLGM`" or "`BitAryo`" (without the quotes).

Examples

input
```
2
1 1
```
output
```
BitLGM
```

input
```
2
1 2
```
output
```
BitAryo
```

input
```
3
1 2 1
```
output
```
BitLGM
```

E. Sausage Maximization

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The Bitlandians are quite weird people. They have their own problems and their own solutions. They have their own thoughts and their own beliefs, they have their own values and their own merits. They have their own dishes and their own sausages!

In Bitland a sausage is an array of integers! A sausage's deliciousness is equal to the bitwise excluding OR (the xor operation) of all integers in that sausage.

One day, when Mr. Bitkoch (the local cook) was going to close his BitRestaurant, BitHaval and BitAryo, the most famous citizens of Bitland, entered the restaurant and each ordered a sausage.

But Mr. Bitkoch had only one sausage left. So he decided to cut a prefix (several, may be zero, first array elements) of the sausage and give it to BitHaval and a postfix (several, may be zero, last array elements) of the sausage and give it to BitAryo. Note that one or both pieces of the sausage can be empty. Of course, the cut pieces mustn't intersect (no array element can occur in both pieces).

The pleasure of BitHaval and BitAryo is equal to the bitwise XOR of their sausages' deliciousness. An empty sausage's deliciousness equals zero.

Find a way to cut a piece of sausage for BitHaval and BitAryo that maximizes the pleasure of these worthy citizens.

Input

The first line contains an integer n ($1 \le n \le 10^5$).

The next line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^{12}$) — Mr. Bitkoch's sausage.

Please, do not use the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use the `cin`, `cout` streams or the `%I64d` specifier.

Output

Print a single integer — the maximum pleasure BitHaval and BitAryo can get from the dinner.

Examples

input

```
2
1 2
```

output

```
3
```

input

```
3
1 2 3
```

output

```
3
```

input

```
2
1000 1000
```

output

```
1000
```