A. Greg's Workout

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

Greg is a beginner bodybuilder. Today the gym coach gave him the training plan. All it had was n integers $a_1, a_2, ..., a_n$. These numbers mean that Greg needs to do exactly n exercises today. Besides, Greg should repeat the i-th in order exercise a_i times.

Greg now only does three types of exercises: "chest" exercises, "biceps" exercises and "back" exercises. Besides, his training is cyclic, that is, the first exercise he does is a "chest" one, the second one is "biceps", the third one is "back", the fourth one is "chest", the fifth one is "biceps", and so on to the n-th exercise.

Now Greg wonders, which muscle will get the most exercise during his training. We know that the exercise Greg repeats the maximum number of times, trains the corresponding muscle the most. Help Greg, determine which muscle will get the most training.

Input

The first line contains integer n $(1 \le n \le 20)$. The second line contains n integers $a_1, a_2, ..., a_n$ $(1 \le a_i \le 25)$ — the number of times Greg repeats the exercises.

Output

Print word "`chest`" (without the quotes), if the chest gets the most exercise, "`biceps`" (without the quotes), if the biceps gets the most exercise and print "`back`" (without the quotes) if the back gets the most exercise.

It is guaranteed that the input is such that the answer to the problem is **unambiguous**.

Sample test(s)

input
```
2
2 8
```
output
```
biceps
```

input
```
3
5 1 10
```
output
```
back
```

input
```
7
3 3 2 7 9 6 8
```
output
```
chest
```

Note

In the first sample Greg does 2 chest, 8 biceps and zero back exercises, so the biceps gets the most exercises.

In the second sample Greg does 5 chest, 1 biceps and 10 back exercises, so the back gets the most exercises.

In the third sample Greg does 18 chest, 12 biceps and 8 back exercises, so the chest gets the most exercise.

B. Code Parsing

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

Little Vitaly loves different algorithms. Today he has invented a new algorithm just for you. Vitaly's algorithm works with string s, consisting of characters "x" and "y", and uses two following operations at runtime:

1. Find two consecutive characters in the string, such that the first of them equals "y", and the second one equals "x" and swap them. If there are several suitable pairs of characters, we choose the pair of characters that is located closer to the beginning of the string.
2. Find in the string two consecutive characters, such that the first of them equals "x" and the second one equals "y". Remove these characters from the string. If there are several suitable pairs of characters, we choose the pair of characters that is located closer to the beginning of the string.

The input for the new algorithm is string s, and the algorithm works as follows:

1. If you can apply at least one of the described operations to the string, go to step 2 of the algorithm. Otherwise, stop executing the algorithm and print the current string.
2. If you can apply operation 1, then apply it. Otherwise, apply operation 2. After you apply the operation, go to step 1 of the algorithm.

Now Vitaly wonders, what is going to be printed as the result of the algorithm's work, if the input receives string s.

Input

The first line contains a non-empty string s.

It is guaranteed that the string only consists of characters "x" and "y". It is guaranteed that the string consists of at most 10^6 characters. It is guaranteed that as the result of the algorithm's execution won't be an empty string.

Output

In the only line print the string that is printed as the result of the algorithm's work, if the input of the algorithm input receives string s.

Sample test(s)

```
input
x
output
x
```

```
input
yxyxy
output
y
```

```
input
xxxxxy
output
xxxx
```

Note

In the first test the algorithm will end after the first step of the algorithm, as it is impossible to apply any operation. Thus, the string won't change.

In the second test the transformation will be like this:

1. string "yxyxy" transforms into string "xyyxy";
2. string "xyyxy" transforms into string "xyxyy";
3. string "xyxyy" transforms into string "xxyyy";
4. string "xxyyy" transforms into string "xyy";
5. string "xyy" transforms into string "y".

As a result, we've got string "y".

In the third test case only one transformation will take place: string "xxxxxy" transforms into string "xxxx". Thus, the answer will be string "xxxx".

C. Almost Arithmetical Progression

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

Gena loves sequences of numbers. Recently, he has discovered a new type of sequences which he called an almost arithmetical progression. A sequence is an *almost arithmetical progression*, if its elements can be represented as:

- $a_1 = p$, where p is some integer;
- $a_i = a_{i-1} + (-1)^{i+1} \cdot q$ $(i > 1)$, where q is some integer.

Right now Gena has a piece of paper with sequence b, consisting of n integers. Help Gena, find there the longest subsequence of integers that is an almost arithmetical progression.

Sequence $s_1,\ s_2,\ \dots,\ s_k$ is a subsequence of sequence $b_1,\ b_2,\ \dots,\ b_n$, if there is such increasing sequence of indexes $i_1, i_2, \dots, i_k$ $(1\ \le\ i_1\ <\ i_2\ < \dots\ <\ i_k\ \le\ n)$, that $b_{i_j}\ =\ s_j$. In other words, sequence s can be obtained from b by crossing out some elements.

Input

The first line contains integer n $(1 \le n \le 4000)$. The next line contains n integers $b_1, b_2, \dots, b_n$ $(1 \le b_i \le 10^6)$.

Output

Print a single integer — the length of the required longest subsequence.

Sample test(s)

input

```
2
3 5
```

output

```
2
```

input

```
4
10 20 10 30
```

output

```
3
```

Note

In the first test the sequence actually is the suitable subsequence.

In the second test the following subsequence fits: $10, 20, 10$.

D. Mr. Bender and Square

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

Mr. Bender has a digital table of size $n \times n$, each cell can be switched on or off. He wants the field to have at least c switched on squares. When this condition is fulfilled, Mr Bender will be happy.

We'll consider the table rows numbered from top to bottom from 1 to n, and the columns — numbered from left to right from 1 to n. Initially there is exactly one switched on cell with coordinates (x, y) (x is the row number, y is the column number), and all other cells are switched off. Then each second we switch on the cells that are off but have the side-adjacent cells that are on.

For a cell with coordinates (x, y) the side-adjacent cells are cells with coordinates $(x - 1, y)$, $(x + 1, y)$, $(x, y - 1)$, $(x, y + 1)$.

In how many seconds will Mr. Bender get happy?

Input

The first line contains four space-separated integers n, x, y, c $(1 \le n, c \le 10^9;\ 1 \le x, y \le n;\ c \le n^2)$.

Output

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

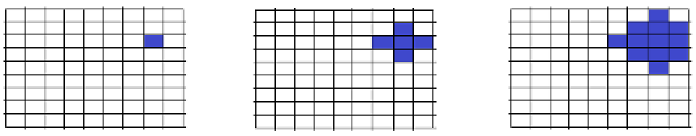
Sample test(s)

input
```
6 4 3 1
```
output
```
0
```

input
```
9 3 8 10
```
output
```
2
```

Note

Initially the first test has one painted cell, so the answer is 0. In the second test all events will go as is shown on the figure.



.

E. Furlo and Rublo and Game

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

Furlo and Rublo play a game. The table has n piles of coins lying on it, the i-th pile has a_i coins. Furlo and Rublo move in turns, Furlo moves first. In one move you are allowed to:

- choose some pile, let's denote the current number of coins in it as x;
- choose some integer y $(0 \le y < x; x^{1/4} \le y \le x^{1/2})$ and decrease the number of coins in this pile to y. In other words, after the described move the pile will have y coins left.

The player who can't make a move, loses.

Your task is to find out, who wins in the given game if both Furlo and Rublo play optimally well.

Input

The first line contains integer n $(1 \le n \le 77777)$ — the number of piles. The next line contains n integers $a_1, a_2, ..., a_n$ $(1 \le a_i \le 777777777777)$ — the sizes of piles. The numbers are separated by single spaces.

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

If both players play optimally well and Furlo wins, print "`Furlo`", otherwise print "`Rublo`". Print the answers without the quotes.

Sample test(s)

input

```
1
1
```

output

```
Rublo
```

input

```
2
1 2
```

output

```
Rublo
```

input

```
10
1 2 3 4 5 6 7 8 9 10
```

output

```
Furlo
```