BitWise LONG

A. Make Good

2 seconds, 256 megabytes

Let's call an array $a_1, a_2, \dots, a_m$ of nonnegative integer numbers **good** if $a_1 + a_2 + \dots + a_m = 2\cdot(a_1 \oplus a_2 \oplus \dots \oplus a_m)$, where $\oplus$ denotes the bitwise XOR operation.

For example, array $[1, 2, 3, 6]$ is good, as $1 + 2 + 3 + 6 = 12 = 2\cdot 6 = 2\cdot (1\oplus 2 \oplus 3 \oplus 6)$. At the same time, array $[1, 2, 1, 3]$ isn't good, as $1 + 2 + 1 + 3 = 7 \neq 2\cdot 1 = 2\cdot(1\oplus 2 \oplus 1 \oplus 3)$.

You are given an array of length n: $a_1, a_2, \dots, a_n$. Append at most 3 elements to it to make it good. Appended elements don't have to be different. It can be shown that the solution always exists under the given constraints. If there are different solutions, you are allowed to output any of them. Note that **you don't have to minimize the number of added elements!**. So, if an array is good already you are allowed to not append elements.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10\,000$). The description of the test cases follows.

The first line of each test case contains a single integer n $(1\le n \le 10^5)$ — the size of the array.

The second line of each test case contains n integers $a_1, a_2, \dots, a_n$ ($0\le a_i \le 10^9$) — the elements of the array.

It is guaranteed that the sum of n over all test cases does not exceed 10^5.

Output

For each test case, output two lines.

In the first line, output a single integer s $(0\le s\le 3)$ — the number of elements you want to append.

In the second line, output s integers $b_1, \dots, b_s$ $(0\le b_i \le 10^{18})$ — the elements you want to append to the array.

If there are different solutions, you are allowed to output any of them.

input

```
3
4
1 2 3 6
1
8
2
1 1
```

output

```
0

2
4 4
3
2 6 2
```

In the first test case of the example, the sum of all numbers is 12, and their $\oplus$ is 6, so the condition is already satisfied.

In the second test case of the example, after adding $4, 4$, the array becomes $[8, 4, 4]$. The sum of numbers in it is 16, $\oplus$ of numbers in it is 8.

B. Dima and a Bad XOR

1 second, 256 megabytes

Student Dima from Kremland has a matrix a of size $n\times m$ filled with non-negative integers.

He wants to select exactly one integer from each row of the matrix so that the `bitwise exclusive OR` of the selected integers is strictly greater than zero. Help him!

Formally, he wants to choose an integers sequence $c_1, c_2, \ldots, c_n$ ($1 \leq c_j \leq m$) so that the inequality $a_{1, c_1} \oplus a_{2, c_2} \oplus \ldots \oplus a_{n, c_n} > 0$ holds, where $a_{i, j}$ is the matrix element from the i-th row and the j-th column.

Here $x \oplus y$ denotes the bitwise XOR operation of integers x and y.

Input

The first line contains two integers n and m $(1 \leq n, m \leq 500)$ — the number of rows and the number of columns in the matrix a.

Each of the next n lines contains m integers: the j-th integer in the i-th line is the j-th element of the i-th row of the matrix a, i.e. $a_{i, j}$ ($0 \leq a_{i, j} \leq 1023$).

Output

If there is no way to choose one integer from each row so that their bitwise exclusive OR is strictly greater than zero, print "NIE".

Otherwise print "TAK" in the first line, in the next line print n integers $c_1, c_2, \ldots c_n$ $(1 \leq c_j \leq m)$, so that the inequality $a_{1, c_1} \oplus a_{2, c_2} \oplus \ldots \oplus a_{n, c_n} > 0$ holds.

If there is more than one possible answer, you may output any.

input

```
3 2
0 0
0 0
0 0
```

output

```
NIE
```

input

```
2 3
7 7 7
7 7 10
```

output

```
TAK
1 3
```

In the first example, all the numbers in the matrix are 0, so it is impossible to select one number in each row of the table so that their bitwise exclusive OR is strictly greater than zero.

In the second example, the selected numbers are 7 (the first number in the first line) and 10 (the third number in the second line), $7 \oplus 10 = 13$, 13 is more than 0, so the answer is found.

C. The Number Of Good Substrings

4 seconds, 256 megabytes

You are given a binary string s (recall that a string is binary if each character is either 0 or 1).

Let $f(t)$ be the decimal representation of integer t written in binary form (possibly with leading zeroes). For example $f(011) = 3, f(00101) = 5, f(00001) = 1, f(10) = 2, f(000) = 0$ and $f(000100) = 4$.

The substring $s_{l}, s_{l+1}, \dots , s_{r}$ is good if $r - l + 1 = f(s_l \dots s_r)$.

For example string $s = 1011$ has 5 good substrings: $s_1 \dots s_1 = 1$, $s_3 \dots s_3 = 1$, $s_4 \dots s_4 = 1$, $s_1 \dots s_2 = 10$ and $s_2 \dots s_4 = 011$.

Your task is to calculate the number of good substrings of string s.

You have to answer t independent queries.

Input

The first line contains one integer t ($1 \le t \le 1000$) — the number of queries.

The only line of each query contains string s ($1 \le |s| \le 2 \cdot 10^5$), consisting of only digits 0 and 1.

It is guaranteed that $\sum_{i=1}^{t} |s_i| \le 2 \cdot 10^5$.

Output

For each query print one integer — the number of good substrings of string s.

input

```
4
0110
0101
00001000
0001000
```

output

```
4
3
4
3
```

D. Bookshelves

1 second, 256 megabytes

Mr Keks is a typical white-collar in Byteland.

He has a bookshelf in his office with some books on it, each book has an integer positive price.

Mr Keks defines the value of a shelf as the sum of books prices on it.

Miraculously, Mr Keks was promoted and now he is moving into a new office.

He learned that in the new office he will have not a single bookshelf, but exactly k bookshelves. He decided that the beauty of the k shelves is the bitwise AND of the values of all the shelves.

He also decided that he won't spend time on reordering the books, so he will place several first books on the first shelf, several next books on the next shelf and so on. Of course, he will place at least one book on each shelf. This way he will put all his books on k shelves in such a way that the beauty of the shelves is as large as possible. Compute this maximum possible beauty.

Input

The first line contains two integers n and k ($1 \le k \le n \le 50$) — the number of books and the number of shelves in the new office.

The second line contains n integers $a_1, a_2, \ldots a_n$, ($0 < a_i < 2^{50}$) — the prices of the books in the order they stand on the old shelf.

Output

Print the maximum possible beauty of k shelves in the new office.

input

```
10 4
9 14 28 1 7 13 15 29 2 31
```

output

```
24
```

input

```
7 3
3 14 15 92 65 35 89
```

output

```
64
```

In the first example you can split the books as follows:

$$(9 + 14 + 28 + 1 + 7)\&(13 + 15)\&(29 + 2)\&(31) = 24.$$

In the second example you can split the books as follows:

$$(3 + 14 + 15 + 92)\&(65)\&(35 + 89) = 64.$$

E. Permutations

2 seconds, 256 megabytes

You are given a permutation p of numbers $1, 2, \ldots, n$. Let's define $f(p)$ as the following sum:

$$f(p) = \sum_{i=1}^{n} \sum_{j=i}^{n} min(p_i, p_{i+1}, \ldots p_j)$$

Find the lexicographically m-th permutation of length n in the set of permutations having the maximum possible value of $f(p)$.

Input

The single line of input contains two integers n and m ($1 \le m \le cnt_n$), where cnt_n is the number of permutations of length n with maximum possible value of $f(p)$.

The problem consists of two subproblems. The subproblems have different constraints on the input. You will get some score for the correct submission of the subproblem. The description of the subproblems follows.

- In subproblem B1 (3 points), the constraint $1 \le n \le 8$ will hold.
- In subproblem B2 (4 points), the constraint $1 \le n \le 50$ will hold.

Output

Output n number forming the required permutation.

input

```
2 2
```

output

```
2 1
```

input

```
3 2
```

output

```
1 3 2
```

In the first example, both permutations of numbers {1, 2} yield maximum possible $f(p)$ which is equal to 4. Among them, $(2, 1)$ comes second in lexicographical order.

F. Maximum Subsequence

1 second, 256 megabytes

You are given an array a consisting of n integers, and additionally an integer m. You have to choose some sequence of indices $b_1, b_2, \ldots, b_k$ ($1 \le b_1 < b_2 < \ldots < b_k \le n$) in such a way that the value of $\sum_{i=1}^{k} a_{b_i} \bmod m$ is maximized. Chosen sequence can be empty.

Print the maximum possible value of $\sum_{i=1}^{k} a_{b_i} \bmod m$.

Input

The first line contains two integers n and m ($1 \le n \le 35$, $1 \le m \le 10^9$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$).

Output

Print the maximum possible value of $\sum_{i=1}^{k} a_{b_i} \bmod m$.

input

```
4 4
5 2 4 1
```

output

```
3
```

input

```
3 20
199 41 299
```

output

```
19
```

In the first example you can choose a sequence $b = \{1, 2\}$, so the sum $\sum_{i=1}^{k} a_{b_i}$ is equal to 7 (and that's 3 after taking it modulo 4).

In the second example you can choose a sequence $b = \{3\}$.

G. Dr. Evil Underscores

1 second, 256 megabytes

Today, as a friendship gift, Bakry gave Badawy n integers $a_1, a_2, \dots, a_n$ and challenged him to choose an integer X such that the value $\max\limits_{1 \le i \le n} (a_i \oplus X)$ is minimum possible, where $\oplus$ denotes the bitwise XOR operation.

As always, Badawy is too lazy, so you decided to help him and find the minimum possible value of $\max\limits_{1 \le i \le n} (a_i \oplus X)$.

Input

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

The second line contains n integers $a_1, a_2, \dots, a_n$ ($0 \le a_i \le 2^{30} - 1$).

Output

Print one integer — the minimum possible value of $\max\limits_{1 \le i \le n} (a_i \oplus X)$.

input

```
3
1 2 3
```

output

```
2
```

input

```
2
1 5
```

output

```
4
```

In the first sample, we can choose $X = 3$.

In the second sample, we can choose $X = 5$.

H. Vasiliy's Multiset

4 seconds, 256 megabytes

Author has gone out of the stories about Vasiliy, so here is just a formal task description.

You are given q queries and a multiset A, initially containing only integer 0. There are three types of queries:

1. "+ x" — add integer x to multiset A.
2. "- x" — erase one occurrence of integer x from multiset A. It's guaranteed that at least one x is present in the multiset A before this query.
3. "? x" — you are given integer x and need to compute the value $\max\limits_{y \in A}(x \oplus y)$, i.e. the maximum value of bitwise exclusive OR (also know as XOR) of integer x and some integer y from the multiset A.

Multiset is a set, where equal elements are allowed.

Input

The first line of the input contains a single integer q ($1 \le q \le 200\,000$) — the number of queries Vasiliy has to perform.

Each of the following q lines of the input contains one of three characters '+', '-' or '?' and an integer x_i ($1 \le x_i \le 10^9$). It's guaranteed that there is at least one query of the third type.

Note, that the integer 0 will always be present in the set A.

Output

For each query of the type '?' print one integer — the maximum value of bitwise exclusive OR (XOR) of integer x_i and some integer from the multiset A.

input

```
10
+ 8
+ 9
+ 11
+ 6
+ 1
? 3
- 8
? 3
? 8
? 11
```

output

```
11
10
14
13
```

After first five operations multiset A contains integers 0, 8, 9, 11, 6 and 1.

The answer for the sixth query is integer $11 = 3 \oplus 8$ — maximum among integers $3 \oplus 0 = 3$, $3 \oplus 9 = 10$, $3 \oplus 11 = 8$, $3 \oplus 6 = 5$ and $3 \oplus 1 = 2$.

I. Mahmoud and Ehab and the xor-MST

2 seconds, 256 megabytes

Ehab is interested in the bitwise-xor operation and the special graphs. Mahmoud gave him a problem that combines both. He has a complete graph consisting of n vertices numbered from 0 to $n - 1$. For all $0 \le u < v < n$, vertex u and vertex v are connected with an undirected edge that has weight $u \oplus v$ (where $\oplus$ is the bitwise-xor operation). Can you find the weight of the minimum spanning tree of that graph?

You can read about complete graphs in https://en.wikipedia.org/wiki/Complete_graph

You can read about the minimum spanning tree in https://en.wikipedia.org/wiki/Minimum_spanning_tree

The weight of the minimum spanning tree is the sum of the weights on the edges included in it.

Input

The only line contains an integer n $(2 \le n \le 10^{12})$, the number of vertices in the graph.

Output

The only line contains an integer x, the weight of the graph's minimum spanning tree.

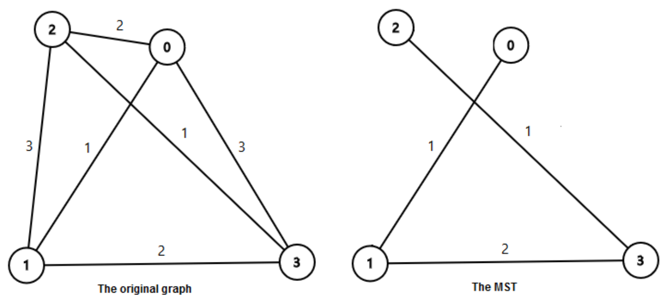
input

```
4
```

output

```
4
```

In the first sample:



The weight of the minimum spanning tree is 1+2+1=4.

J. Ehab and the Expected XOR Problem

1 second, 256 megabytes

Given two integers n and x, construct an array that satisfies the following conditions:

- for any element a_i in the array, $1 \le a_i < 2^n$;
- there is no **non-empty** subsegment with bitwise XOR equal to 0 or x,
- its length l should be maximized.

A sequence b is a subsegment of a sequence a if b can be obtained from a by deletion of several (possibly, zero or all) elements from the beginning and several (possibly, zero or all) elements from the end.

Input

The only line contains two integers n and x ($1 \le n \le 18$, $1 \le x < 2^{18}$).

Output

The first line should contain the length of the array l.

If l is positive, the second line should contain l space-separated integers $a_1, a_2, \dots, a_l$ ($1 \le a_i < 2^n$) — the elements of the array a.

If there are multiple solutions, print any of them.

input
```
3 5
```
output
```
3
6 1 3
```

input
```
2 4
```
output
```
3
1 3 1
```

input
```
1 1
```
output
```
0
```

In the first example, the bitwise XOR of the subsegments are $\{6, 7, 4, 1, 2, 3\}$.

K. Shortest Cycle

1 second, 256 megabytes

You are given n integer numbers $a_1, a_2, \dots, a_n$. Consider graph on n nodes, in which nodes i, j ($i \ne j$) are connected if and only if, a_i AND $a_j \ne 0$, where AND denotes the bitwise AND operation.

Find the length of the shortest cycle in this graph or determine that it doesn't have cycles at all.

Input

The first line contains one integer n $(1 \le n \le 10^5)$ — number of numbers.

The second line contains n integer numbers $a_1, a_2, \dots, a_n$ ($0 \le a_i \le 10^{18}$).

Output

If the graph doesn't have any cycles, output -1. Else output the length of the shortest cycle.

input
```
4
3 6 28 9
```
output
```
4
```

input
```
5
5 12 9 16 48
```
output
```
3
```

input
```
4
1 2 4 8
```
output
```
-1
```

In the first example, the shortest cycle is $(9, 3, 6, 28)$.

In the second example, the shortest cycle is $(5, 12, 9)$.

The graph has no cycles in the third example.

L. The Brand New Function

2 seconds, 256 megabytes

Polycarpus has a sequence, consisting of n non-negative integers: $a_1, a_2, \dots, a_n$.

Let's define function $f(l, r)$ (l, r are integer, $1 \le l \le r \le n$) for sequence a as an operation of bitwise OR of all the sequence elements with indexes from l to r. Formally: $f(l, r) = a_l \mid a_{l+1} \mid \dots \mid a_r$.

Polycarpus took a piece of paper and wrote out the values of function $f(l, r)$ for all l, r (l, r are integer, $1 \le l \le r \le n$). Now he wants to know, how many **distinct** values he's got in the end.

Help Polycarpus, count the number of distinct values of function $f(l, r)$ for the given sequence a.

Expression $x \mid y$ means applying the operation of bitwise OR to numbers x and y. This operation exists in all modern programming languages, for example, in language *C++* and *Java* it is marked as "`|`", in *Pascal* — as "`or`".

Input

The first line contains integer n $(1 \le n \le 10^5)$ — the number of elements of sequence a. The second line contains n space-separated integers $a_1, a_2, \dots, a_n$ $(0 \le a_i \le 10^6)$ — the elements of sequence a.

Output

Print a single integer — the number of distinct values of function $f(l, r)$ for the given sequence a.

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

input
```
3
1 2 0
```

output

```
4
```

input

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

output

```
11
```

In the first test case Polycarpus will have 6 numbers written on the paper: $f(1, 1) = 1, f(1, 2) = 3, f(1, 3) = 3, f(2, 2) = 2, f(2, 3) = 2, f(3, 3) = 0$. There are exactly 4 distinct numbers among them: $0, 1, 2, 3$.

M. Ehab and another another xor problem

1 second, 256 megabytes

This is an interactive problem!

Ehab plays a game with Laggy. Ehab has 2 hidden integers (a, b). Laggy can ask a pair of integers (c, d) and Ehab will reply with:

- 1 if $a \oplus c > b \oplus d$.
- 0 if $a \oplus c = b \oplus d$.
- -1 if $a \oplus c < b \oplus d$.

Operation $a \oplus b$ is the bitwise-xor operation of two numbers a and b.

Laggy should guess (a, b) with **at most 62 questions**. You'll play this game. You're Laggy and the interactor is Ehab.

It's guaranteed that $0 \le a, b < 2^{30}$.

Input

See the interaction section.

Output

To print the answer, print "! a b" (without quotes). **Don't forget to flush the output after printing the answer**.

Interaction

To ask a question, print "? c d" (without quotes). Both c and d must be non-negative integers less than 2^{30} . **Don't forget to flush the output after printing any question**.

After each question, you should read the answer as mentioned in the legend. If the interactor replies with -2, that means you asked more than 62 queries and your program should terminate.

To flush the output, you can use:-

- fflush(stdout) in C++.
- System.out.flush() in Java.
- stdout.flush() in Python.
- flush(output) in Pascal.
- See the documentation for other languages.

Hacking:

To hack someone, print the 2 space-separated integers a and b $(0 \le a, b < 2^{30})$.

input

```
1
-1
0
```

output

```
? 2 1
? 1 2
? 2 0
! 3 1
```

In the sample:

The hidden numbers are $a = 3$ and $b = 1$.

In the first query: $3 \oplus 2 = 1$ and $1 \oplus 1 = 0$, so the answer is 1.

In the second query: $3 \oplus 1 = 2$ and $1 \oplus 2 = 3$, so the answer is -1.

In the third query: $3 \oplus 2 = 1$ and $1 \oplus 0 = 1$, so the answer is 0.

Then, we printed the answer.

N. XOR on Segment

4 seconds, 256 megabytes

You've got an array a, consisting of n integers $a_1, a_2, ..., a_n$. You are allowed to perform two operations on this array:

1. Calculate the sum of current array elements on the segment $[l, r]$, that is, count value $a_l + a_{l+1} + ... + a_r$.
2. Apply the xor operation with a given number x to each array element on the segment $[l, r]$, that is, execute $a_l = a_l \oplus x, a_{l+1} = a_{l+1} \oplus x, \dots, a_r = a_r \oplus x$. This operation changes exactly $r - l + 1$ array elements.

Expression $x \oplus y$ means applying bitwise xor operation to numbers x and y. The given operation exists in all modern programming languages, for example in language *C++* and *Java* it is marked as "^", in *Pascal* — as "xor".

You've got a list of m operations of the indicated type. Your task is to perform all given operations, for each sum query you should print the result you get.

Input

The first line contains integer n $(1 \le n \le 10^5)$ — the size of the array. The second line contains space-separated integers $a_1, a_2, ..., a_n$ $(0 \le a_i \le 10^6)$ — the original array.

The third line contains integer m $(1 \le m \le 5 \cdot 10^4)$ — the number of operations with the array. The i-th of the following m lines first contains an integer t_i $(1 \le t_i \le 2)$ — the type of the i-th query. If $t_i = 1$, then this is the query of the sum, if $t_i = 2$, then this is the query to change array elements. If the i-th operation is of type 1, then next follow two integers l_i, r_i $(1 \le l_i \le r_i \le n)$. If the i-th operation is of type 2, then next follow three integers l_i, r_i, x_i $(1 \le l_i \le r_i \le n, 1 \le x_i \le 10^6)$. The numbers on the lines are separated by single spaces.

Output

For each query of type 1 print in a single line the sum of numbers on the given segment. Print the answers to the queries in the order in which the queries go in the input.

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.

input

```
5
4 10 3 13 7
8
1 2 4
2 1 3 3
1 2 4
1 3 3
2 2 5 5
1 1 5
2 1 2 10
1 2 3
```

output

```
26
22
0
34
11
```

input

```
6
4 7 4 0 7 3
5
2 2 3 8
1 1 5
2 3 5 1
2 4 5 6
1 2 3
```

output

```
38
28
```