Manthan, Codefest 19 (open for everyone, rated, Div. 1 + Div. 2)

A. XORinacci

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

Cengiz recently learned Fibonacci numbers and now he is studying different algorithms to find them. After getting bored of reading them, he came with his own new type of numbers that he named *XORinacci* numbers. He defined them as follows:

- $f(0) = a$;
- $f(1) = b$;
- $f(n) = f(n-1) \oplus f(n-2)$ when $n > 1$, where $\oplus$ denotes the bitwise XOR operation.

You are given three integers a, b, and n, calculate $f(n)$.

You have to answer for T independent test cases.

Input

The input contains one or more independent test cases.

The first line of input contains a single integer T ($1 \le T \le 10^3$), the number of test cases.

Each of the T following lines contains three space-separated integers a, b, and n ($0 \le a, b, n \le 10^9$) respectively.

Output

For each test case, output $f(n)$.

Example

input
```
3
3 4 2
4 5 0
325 265 1231232
```
output
```
7
4
76
```

Note

In the first example, $f(2) = f(0) \oplus f(1) = 3 \oplus 4 = 7$.

B. Uniqueness

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

You are given an array $a_1, a_2, \ldots, a_n$. You can remove **at most one** subsegment from it. The remaining elements should be pairwise distinct.

In other words, **at most one** time you can choose two integers l and r ($1 \le l \le r \le n$) and delete integers $a_l, a_{l+1}, \ldots, a_r$ from the array. Remaining elements should be pairwise distinct.

Find the minimum size of the subsegment you need to remove to make all remaining elements distinct.

Input

The first line of the input contains a single integer n ($1 \le n \le 2000$) — the number of elements in the given array.

The next line contains n spaced integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the elements of the array.

Output

Print a single integer — the minimum size of the subsegment you need to remove to make all elements of the array pairwise distinct. If no subsegment needs to be removed, print 0.

Examples

input
```
3
1 2 3
```
output
```
0
```

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

input
```
5
1 4 1 4 9
```
output
```
2
```

Note

In the first example all the elements are already distinct, therefore no subsegment needs to be removed.

In the second example you can remove the subsegment from index 2 to 3.

In the third example you can remove the subsegments from index 1 to 2, or from index 2 to 3, or from index 3 to 4.

C. Magic Grid

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

Let us define a *magic* grid to be a square matrix of integers of size $n \times n$, satisfying the following conditions.

- All integers from 0 to $(n^2 - 1)$ inclusive appear in the matrix **exactly once**.
- Bitwise XOR of all elements in a row or a column must be the same for each row and column.

You are given an integer n which is a **multiple of** 4. Construct a *magic* grid of size $n \times n$.

Input

The only line of input contains an integer n ($4 \le n \le 1000$). It is guaranteed that n is a multiple of 4.

Output

Print a *magic* grid, i.e. n lines, the i-th of which contains n space-separated integers, representing the i-th row of the grid.

If there are multiple answers, print any. We can show that an answer always exists.

Examples

input

```
4
```

output

```
8 9 1 13
3 12 7 5
0 2 4 11
6 10 15 14
```

input

```
8
```

output

```
19 55 11 39 32 36 4 52
51 7 35 31 12 48 28 20
43 23 59 15 0 8 16 44
3 47 27 63 24 40 60 56
34 38 6 54 17 53 9 37
14 50 30 22 49 5 33 29
2 10 18 46 41 21 57 13
26 42 62 58 1 45 25 61
```

Note

In the first example, XOR of each row and each column is 13.

In the second example, XOR of each row and each column is 60.

D. Restore Permutation

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

An array of integers $p_1, p_2, \ldots, p_n$ is called a permutation if it contains each number from 1 to n exactly once. For example, the following arrays are permutations: $[3, 1, 2]$, $[1]$, $[1, 2, 3, 4, 5]$ and $[4, 3, 1, 2]$. The following arrays are not permutations: $[2]$, $[1, 1]$, $[2, 3, 4]$.

There is a hidden permutation of length n.

For each index i, you are given s_i, which equals to the sum of all p_j such that $j < i$ and $p_j < p_i$. In other words, s_i is the sum of elements before the i-th element that are smaller than the i-th element.

Your task is to restore the permutation.

Input

The first line contains a single integer n ($1 \le n \le 2 \cdot 10^5$) — the size of the permutation.

The second line contains n integers $s_1, s_2, \ldots, s_n$ ($0 \le s_i \le \frac{n(n-1)}{2}$).

It is guaranteed that the array s corresponds to a valid permutation of length n.

Output

Print n integers $p_1, p_2, \ldots, p_n$ — the elements of the restored permutation. We can show that the answer is always unique.

Examples

input

```
3
0 0 0
```

output

```
3 2 1
```

input

```
2
0 1
```

output

```
1 2
```

input

```
5
0 1 1 1 10
```

output

```
1 4 3 2 5
```

Note

In the first example for each i there is no index j satisfying both conditions, hence s_i are always 0.

In the second example for $i = 2$ it happens that $j = 1$ satisfies the conditions, so $s_2 = p_1$.

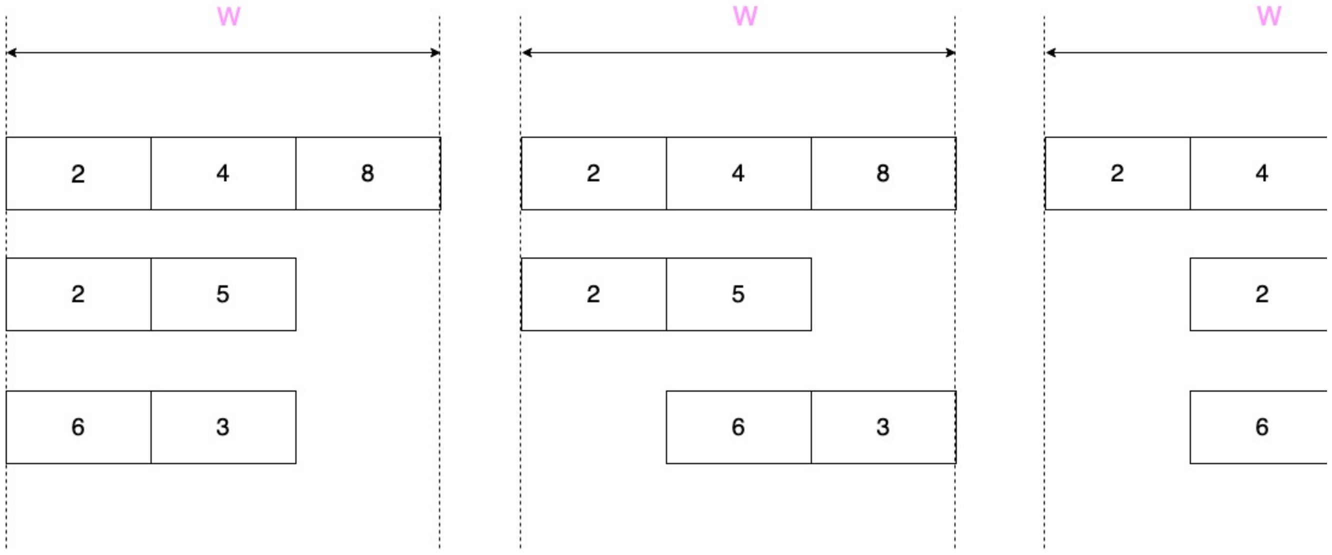
In the third example for $i = 2, 3, 4$ only $j = 1$ satisfies the conditions, so $s_2 = s_3 = s_4 = 1$. For $i = 5$ all $j = 1, 2, 3, 4$ are possible, so $s_5 = p_1 + p_2 + p_3 + p_4 = 10$.

E. Let Them Slide

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given n arrays that can have different sizes. You also have a table with w columns and n rows. The i-th array is placed horizontally in the i-th row. You can slide each array within its row as long as it occupies several consecutive cells and lies completely inside the table.

You need to find the maximum sum of the integers in the j-th column for each j from 1 to w independently.



Optimal placements for columns 1, 2 and 3 are shown on the pictures from left to right.

Note that you can exclude any array out of a column provided it remains in the window. In this case its value is considered to be zero.

Input

The first line contains two integers n ($1 \le n \le 10^6$) and w ($1 \le w \le 10^6$) — the number of arrays and the width of the table.

Each of the next n lines consists of an integer l_i ($1 \le l_i \le w$), the length of the i-th array, followed by l_i integers $a_{i1}, a_{i2}, \ldots, a_{il_i}$ ($-10^9 \le a_{ij} \le 10^9$) — the elements of the array.

The total length of the arrays does no exceed 10^6.

Output

Print w integers, the i-th of them should be the maximum sum for column i.

Examples

input

```
3 3
3 2 4 8
2 2 5
2 6 3
```

output

```
10 15 16
```

input

```
2 2
2 7 8
1 -8
```

output

```
7 8
```

Note

Illustration for the first example is in the statement.

F. Bits And Pieces

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

output: standard output

You are given an array a of n integers.

You need to find the maximum value of $a_i|(a_j\&a_k)$ over all triplets (i, j, k) such that $i < j < k$.

Here $\&$ denotes the bitwise AND operation, and $|$ denotes the bitwise OR operation.

Input

The first line of input contains the integer n ($3 \le n \le 10^6$), the size of the array a.

Next line contains n space separated integers a_1, a_2, ..., a_n ($0 \le a_i \le 2 \cdot 10^6$), representing the elements of the array a.

Output

Output a single integer, the maximum value of the expression given in the statement.

Examples

input

```
3
2 4 6
```

output

```
6
```

input

```
4
2 8 4 7
```

output

```
12
```

Note

In the first example, the only possible triplet is $(1, 2, 3)$. Hence, the answer is $2|(4\&6) = 6$.

In the second example, there are 4 possible triplets:

1. $(1, 2, 3)$, value of which is $2|(8\&4) = 2$.
2. $(1, 2, 4)$, value of which is $2|(8\&7) = 2$.
3. $(1, 3, 4)$, value of which is $2|(4\&7) = 6$.
4. $(2, 3, 4)$, value of which is $8|(4\&7) = 12$.

The maximum value hence is 12.

G. Polygons

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

You are given two integers n and k.

You need to construct k regular polygons having same circumcircle, with **distinct** number of sides l between 3 and n.

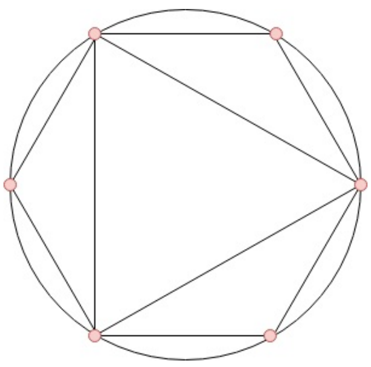


Illustration for the first example.

You can rotate them to minimize the total number of distinct points on the circle. Find the minimum number of such points.

Input

The only line of input contains two integers n and k ($3 \le n \le 10^6$, $1 \le k \le n-2$), the maximum number of sides of a polygon and the number of polygons to construct, respectively.

Output

Print a single integer — the minimum number of points required for k polygons.

Examples

input

```
6 2
```

output

```
6
```

input

```
200 50
```

output

```
708
```

Note

In the first example, we have $n = 6$ and $k = 2$. So, we have 4 polygons with number of sides 3, 4, 5 and 6 to choose from and if we choose the triangle and the hexagon, then we can arrange them as shown in the picture in the statement.

Hence, the minimum number of points required on the circle is 6, which is also the minimum overall possible sets.

H. Red Blue Tree

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

You are given a tree of n nodes. The tree is rooted at node 1, which is not considered as a leaf regardless of its degree.

Each leaf of the tree has one of the two colors: *red* or *blue*. Leaf node v initially has color s_v.

The color of each of the internal nodes (including the root) is determined as follows.

- Let b be the number of *blue* immediate children, and r be the number of *red* immediate children of a given vertex.
- Then the color of this vertex is *blue* if and only if $b - r \geq k$, otherwise *red*.

Integer k is a parameter that is same for all the nodes.

You need to handle the following types of queries:

- 1 v: print the color of node v;
- 2 v c: change the color of leaf v to c ($c = 0$ means red, $c = 1$ means blue);
- 3 h: update the current value of k to h.

Input

The first line of the input consists of two integers n and k ($2 \leq n \leq 10^5$, $-n \leq k \leq n$) — the number of nodes and the initial parameter k.

Each of the next $n - 1$ lines contains two integers u and v ($1 \leq u, v \leq n$), denoting that there is an edge between vertices u and v.

The next line consists of n space separated integers — the initial array s ($-1 \leq s_i \leq 1$). $s_i = 0$ means that the color of node i is red. $s_i = 1$ means that the color of node i is blue. $s_i = -1$ means that the node i is not a leaf.

The next line contains an integer q ($1 \leq q \leq 10^5$), the number of queries.

q lines follow, each containing a query in one of the following queries:

- 1 v ($1 \leq v \leq n$): print the color of node v;
- 2 v c ($1 \leq v \leq n$, $c = 0$ or $c = 1$): change the color of leaf v to c ($c = 0$ means red, $c = 1$ means blue). It is guaranteed that v is a leaf;
- 3 h ($-n \leq h \leq n$): update the current value of k to h.

Output

For each query of the first type, print 0 if the color of vertex v is red, and 1 otherwise.

Example

input

```
5 2
1 2
1 3
2 4
2 5
-1 -1 0 1 0
9
1 1
1 2
3 -2
1 1
1 2
3 1
2 5 1
1 1
1 2
```

output

```
0
0
1
1
0
1
```

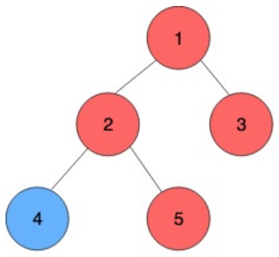
Note

Figures:

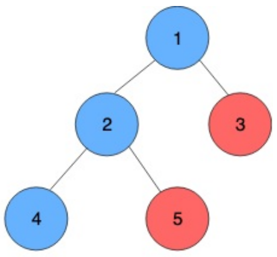
(i) The initial tree

(ii) The tree after the 3rd query

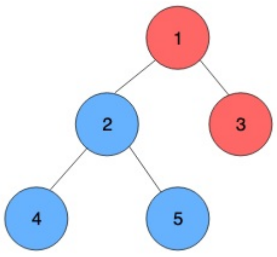
(iii) The tree after the 7th query



(i)



(ii)



(iii)