A. Array Without Local Maximums

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Ivan unexpectedly saw a present from one of his previous birthdays. It is array of n numbers from 1 to 200. Array is old and some numbers are hard to read. Ivan remembers that for all elements at least one of its neighbours ls not less than it, more formally:

$a_1 \le a_2$,

$a_n \le a_{n-1}$ and

$a_i \le max(a_{i-1}, \; a_{i+1})$ for all i from 2 to $n-1$.

Ivan does not remember the array and asks to find the number of ways to restore it. Restored elements also should be integers from 1 to 200. Since the number of ways can be big, print it modulo 998244353.

Input

First line of input contains one integer n ($2 \le n \le 10^5$) — size of the array.

Second line of input contains n integers a_i — elements of array. Either $a_i = -1$ or $1 \le a_i \le 200$. $a_i = -1$ means that i-th element can't be read.

Output

Print number of ways to restore the array modulo 998244353.

Examples

input

```
3
1 -1 2
```

output

```
1
```

input

```
2
-1 -1
```

output

```
200
```

Note

In the first example, only possible value of a_2 is 2.

In the second example, $a_1 = a_2$ so there are 200 different values because all restored elements should be integers between 1 and 200.

B. Multihedgehog

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

Someone give a strange birthday present to Ivan. It is hedgehog — connected undirected graph in which one vertex has degree at least 3 (we will call it center) and all other vertices has degree 1. Ivan thought that hedgehog is too boring and decided to make himself k-multihedgehog.

Let us define k-multihedgehog as follows:

- 1-multihedgehog is hedgehog: it has one vertex of degree at least 3 and some vertices of degree 1.
- For all $k \ge 2$, k-multihedgehog is $(k-1)$-multihedgehog in which the following changes has been made for each vertex v with degree 1: let u be its only neighbor; remove vertex v, create a new hedgehog with center at vertex w and connect vertices u and w with an edge. New hedgehogs can differ from each other and the initial gift.

Thereby k-multihedgehog is a tree. Ivan made k-multihedgehog but he is not sure that he did not make any mistakes. That is why he asked you to check if his tree is indeed k-multihedgehog.

Input

First line of input contains 2 integers n, k ($1 \le n \le 10^5$, $1 \le k \le 10^9$) — number of vertices and hedgehog parameter.

Next $n-1$ lines contains two integers u v ($1 \le u, \ v \le n; \ u \ne v$) — indices of vertices connected by edge.

It is guaranteed that given graph is a tree.

Output

Print "Yes" (without quotes), if given graph is k-multihedgehog, and "No" (without quotes) otherwise.

Examples

input

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

output

```
Yes
```

input

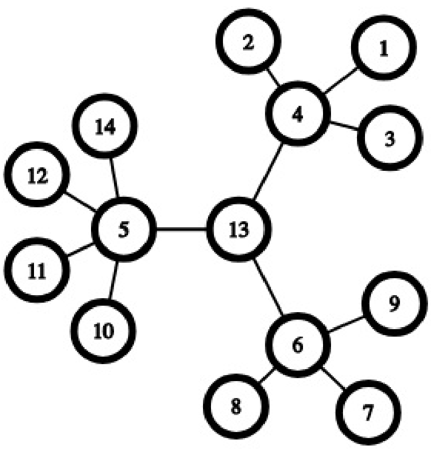
```
3 1
1 3
2 3
```

output

```
No
```

Note

2-multihedgehog from the first example looks like this:



Its center is vertex 13. Hedgehogs created on last step are: [4 (center), 1, 2, 3], [6 (center), 7, 8, 9], [5 (center), 10, 11, 12, 13].

Tree from second example is not a hedgehog because degree of center should be at least 3.

C. Knights

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

Ivan places knights on infinite chessboard. Initially there are n knights. If there is free cell which is under attack of at least 4 knights then he places new knight in this cell. Ivan repeats this until there are no such free cells. One can prove that this process is finite. One can also prove that position in the end does not depend on the order in which new knights are placed.

Ivan asked you to find initial placement of exactly n knights such that in the end there will be at least $\lfloor \frac{n^2}{10} \rfloor$ knights.

Input

The only line of input contains one integer n ($1 \le n \le 10^3$) — number of knights in the initial placement.

Output

Print n lines. Each line should contain 2 numbers x_i and y_i ($-10^9 \le x_i, \; y_i \le 10^9$) — coordinates of i-th knight. For all $i \ne j$, $(x_i, \; y_i) \ne (x_j, \; y_j)$ should hold. In other words, all knights should be in different cells.

It is guaranteed that the solution exists.

Examples

input

```
4
```

output

```
1 1
3 1
1 5
4 4
```

input

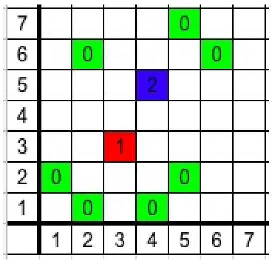
```
7
```

output

```
2 1
1 2
4 1
5 2
2 6
5 7
6 6
```

Note

Let's look at second example:



Green zeroes are initial knights. Cell $(3, \; 3)$ is under attack of 4 knights in cells $(1, \; 2)$, $(2, \; 1)$, $(4, \; 1)$ and $(5, \; 2)$, therefore Ivan will place a knight in this cell. Cell $(4, \; 5)$ is initially attacked by only 3 knights in cells $(2, \; 6)$, $(5, \; 7)$ and $(6, \; 6)$. But new knight in cell $(3, \; 3)$ also attacks cell $(4, \; 5)$, now it is attacked by 4 knights and Ivan will place another knight in this cell. There are no more free cells which are attacked by 4 or more knights, so the process stops. There are 9 knights in the end, which is not less than $\lfloor \frac{7^2}{10} \rfloor = 4$.

D. Computer Game

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Ivan plays some computer game. There are n quests in the game. Each quest can be upgraded once, this increases the reward for its completion. Each quest has 3 parameters a_i, b_i, p_i: reward for completing quest before upgrade, reward for completing quest after upgrade ($a_i < b_i$) and probability of successful completing the quest.

Each second Ivan can try to complete one quest and he will succeed with probability p_i. In case of success Ivan will get the reward and opportunity to upgrade any one quest (not necessary the one he just completed). In case of failure he gets nothing. Quests **do not vanish** after completing.

Ivan has t seconds. He wants to maximize expected value of his total gain after t seconds. Help him to calculate this value.

Input

First line contains 2 integers n ($1 \le n \le 10^5$) and t ($1 \le t \le 10^{10}$) — number of quests and total time.

Following n lines contain description of quests. Each description is 3 numbers a_i b_i p_i ($1 \le a_i < b_i \le 10^8$, $0 < p_i < 1$) — reward for completing quest before upgrade, reward for completing quest after upgrade and probability of successful completing of quest. a_i and b_i are integers. All probabilities are given with at most 9 decimal places.

Output

Print the expected value.

Your answer will be accepted if absolute or relative error does not exceed 10^{-6}. Formally, let your answer be a, and the jury's answer be b. Your answer is considered correct if $\frac{|a-b|}{max(b,\ 1)} \le 10^{-6}$.

Examples

input

```
3 2
3 1000 0.5
1 2 0.48
3 20 0.3
```

output

```
252.2500000000000
```

input

```
2 2
1 1000 0.1
2 3 0.2
```

output

```
20.7200000000000
```

E. Random Forest Rank

time limit per test: 3 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Let's define rank of undirected graph as rank of its adjacency matrix in $\mathbb{R}^{n \times n}$.

Given a tree. Each edge of this tree will be deleted with probability $1/2$, all these deletions are independent. Let E be the expected rank of resulting forest. Find $E \cdot 2^{n-1}$ modulo 998244353 (it is easy to show that $E \cdot 2^{n-1}$ is an integer).

Input

First line of input contains n ($1 \le n \le 5 \cdot 10^5$) — number of vertices.

Next $n-1$ lines contains two integers u v ($1 \le u,\ v \le n;\ u \ne v$) — indices of vertices connected by edge.

It is guaranteed that given graph is a tree.

Output

Print one integer — answer to the problem.

Examples

input

```
3
1 2
2 3
```

output

```
6
```

input

```
4
1 2
1 3
1 4
```

output

```
14
```

input

```
4
1 2
2 3
3 4
```

output

```
18
```