Problem A. Centroid Tree

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Consider the following model to generate random trees with n vertices. Initially, there are n vertices and no edges in the graph. Then $n-1$ times a pair of vertices (u, v) is chosen randomly and uniformly among all correct pairs and edge between u and v is added. A pair (u, v) is correct, if there is no path between u and v in the graph.

For a tree T we can define rooted centroid-tree $C(T)$ which has the same number of vertices. Denote trees obtained by removing vertex u of degree d_u from T as $T_u(1), T_u(2), \dots T_u(d_u)$. The tree $C(T)$ is built in the following way.

1. If $|T| = 1$, then $C(T) = T$ (here $|T|$ stands for the number of vertices).
2. Otherwise, the root of $C(T)$ is the vertex u such that $\max\limits_{1 \le i \le d_u} |T_u(i)|$ is minimum possible among all vertices. If there are several such vertices, the one with minimum index is chosen.
3. Then edges between u and roots of $C(T_u(1)), C(T_u(2)), \dots C(T_u(d_u))$ are added, where $C(T_u(i))$ are built recursively.

You are given t trees with n vertices each. Each of them is either randomly generated tree or a centroid-tree of randomly generated tree. Your task is to determine which of two algorithms was used to generate the tree. Your answer for the test will be considered correct if at least $t - 1$ guesses are correct.

There are exactly 42 non-sample tests. Your solution will get verdict `OK` for sample even if you are wrong on both trees. Parameters n and t were chosen before trees were generated.

Input

The first line of the input contains two integers t and n ($60 \le n \le 1000$, $n \cdot t = 300\,000$) denoting the number of trees in the input and the number of vertices in each tree.

Then follow t blocks of $n - 1$ lines each. Each block describe edges (u_i, v_i) of the tree. Vertices are numbered 1 through n.

Output

Output t lines, each containing a singe word "`Centroid`" or "`Random`" denoting the algorithm that was used to generate the corresponding tree.

Example

standard input	standard output

2 7
1 2
2 3
3 4
4 5
5 6
6 7
1 2
1 3
2 4
2 5
3 6
3 7
	Random
	Centroid

Problem B. Completely Multiplicative Function

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 mebibytes

A function $f : \mathbb{N} \to \mathbb{R}$ is *completely multiplicative* if $f(xy) = f(x)f(y)$ for all $x, y \in \mathbb{N}$.

n-balance of a function f is the maximum value of $|f(1) + \ldots + f(k)|$ for $1 \le k \le n$.

Construct a completely multiplicative function f such that $|f(x)| = 1$ for all x, and its 10^6-balance does not exceed 20.

Input

The first line contains one integer n.

The only non-sample test has $n = 10^6$.

Output

Print n integers — values of $f(1)$, …, $f(n)$ of a completely multiplicative function f with n-balance less than or equal to 20.

Example

standard input	standard output
`10`	`1 -1 -1 1 1 1 -1 -1 1 -1`

Problem C. Even and Odd

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

There is a connected undirected graph with n vertices and m edges. There are no self-loops or multiple edges.

An unordered pair of distinct vertices $\{u, v\}$ is called *even* if every simple path (path visits no vertex twice) between u and v contains an even number of edges. Similarly, an unordered pair of distinct vertices is called *odd* if every simple path connecting them consists of an odd number of edges.

Find the number of odd and even vertex pairs in the given graph.

Input

The first line of the input contains two integers n and m ($1 \leqslant n \leqslant 200\,000$, $0 \leqslant m \leqslant 200\,000$) — the number of vertices and edges respectively. Each of the next m lines contains two endpoints of the corresponding edge a_i and b_i ($1 \leqslant a_i, b_i \leqslant n$). It is guaranteed that the graph is connected and contains no self-loops and multiple edges.

Output

Print two integers — the number of even and odd pairs respectively.

Examples

standard input	standard output

4 3
1 2
1 3
1 4
	3 3

4 4
1 2
2 3
3 4
4 2
	0 1

Note

In the first sample, there are three even pairs $\{2, 3\}$, $\{2, 4\}$, $\{3, 4\}$, and three odd pairs $\{1, 2\}$, $\{1, 3\}$, $\{1, 4\}$.

In the second sample, there is a single odd pair $\{1, 2\}$. One can verify that for any other pair of vertices it is possible to find both an even path and an odd path connecting them.

Problem D. Great Again

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 megabytes

The election in Berland is coming. The party United Berland is going to use its influence to win them again. The crucial condition for the party is to win the election in the capital to show the world that the protests of opposition in it are inspired by external enemies.

The capital of Berland consists of only one long road with n people living alongside it. United Berland has a lot of informers, so they know for each citizen whether he is going to attend the election, and if yes, who is he going to vote for: the ruling party or the opposition.

United Berland has a vast soft power, so they can lobby the desired distribution of districts. Every district should be a consecutive segment of the road of length between l and r inclusive. Each citizen must be assigned to exactly one district. The votes are counted in each district separately, and the parties receive one point for each district, where it receives strictly more votes than the other party. If the parties got equal result in this district, no one gets its vote. United Berland is going to create the distribution that maximizes the difference of its points and points of the opposition, and you are asked to compute this value.

Input

The first line of the input contains three positive integers n, l, r ($1 \le n \le 300\,000$, $1 \le l \le r \le n$) — the number of citizens in the capital, the lower and the upper bounds on the possible length of a district.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($a_i \in \{-1, 0, 1\}$), denoting the votes of the citizens. 1 means vote for the ruling party, -1 means vote for opposition, 0 means that this citizen is not going to come to the elections.

Output

If there is no way to divide the road into districts of lengths between l and r, print "`Impossible`" (without quotes).

Otherwise, print one integer — the maximum possible difference between the scores of United Berland and the opposition in a valid distribution of citizens among voting districts.

Examples

standard input	standard output

`5 1 5`
`1 -1 0 -1 1`
	`1`

`5 2 3`
`-1 1 -1 1 -1`
	`-1`

`6 1 1`
`1 -1 -1 -1 -1 -1`
	`-4`

`5 3 3`
`1 1 1 1 1`
	`Impossible`

Note

In the first sample, the optimal division of districts is $\{1\}, \{2, 3, 4\}, \{5\}$.

In the second sample, the optimal division is $\{1, 2\}, \{3, 4, 5\}$.

In the third sample, there is only one possible division.

There is no way to divide 5 in segments of length 3, so in the fourth sample the answer is "`Impossible`".

Problem E. Jumping is Fun

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 mebibytes

Today is a day off in a programming camp you attend, and you decided to go the trampolining center. There are n trampolines in a row with positions $1, \ldots, n$. The distance between two successive trampolines is 1 meter. The size of the trampoline can be neglected. The i-th trampoline has power $a_i > 0$. This means that you can directly jump from the i-th trampoline to any other which is not further than a_i meters. Formally, you can directly jump from trampoline i to j if $|i - j| \le a_i$.

Staying in the same place is really boring. However, if you start jumping in random directions it may bother other jumpers. You decide to select two trampolines, x and y, and jump from one to another and back. While jumping from one trampoline to another you may visit any other trampolines as intermediate. They can be located everywhere, not necessary between x and y. Of course, when you move from one selected trampoline to another, you always use minimum possible number of jumps.

You feel frustrated if you arrive to y too soon after starting from x or vice versa. You want to select two trampolines in such a way that the minimum of these times is maximized.

Formally, we define the *distance* between two trampolines x and y as the minimum number of jumps required to travel either from x to y or from y to x. Note that distance is well-defined for each pair of trampolines as all a_i are positive.

You know the power of each trampoline. Find the pair with maximum distance.

Input

The first line of the input contains a single integer n ($2 \le n \le 200\,000$). In the next line there are n integers a_i ($1 \le a_i < n$) — powers of the trampolines.

Output

Print one integer — the maximum possible distance between a pair of trampolines.

Examples

standard input	standard output
8 7 1 1 1 1 1 1 7	3
10 2 2 1 2 2 1 2 2 1 2	6

Note

In the first sample, we can pick $x = 3$ and $y = 6$.

In the second sample, we can pick $x = 1$ and $y = 10$.

Problem F. Online LCS

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	512 mebibytes

There are two strings s_0 and s_1, which are initially empty. Let $lcs(a, b)$ be the length of the longest string t such that t is a substring of both a and b. Note that $lcs(a, b)$ can be zero.

You are to process q queries. The i-th query is described by numbers a_i and c_i, each of them either 0 or 1. To process the i-th query, you should add the character c_i to the end of the string s_{a_i}, and find the value of $lcs(s_0, s_1)$ for the updated strings s_0 and s_1. All operations must be performed *online*, that is, the answer for the i-th query must be found before processing the $(i + 1)$-th.

Input

The first line of the input contains a single integer q ($1 \le q \le 5 \cdot 10^6$) — the number of queries.

The second line contains a string $w = w_1 w_2 \dots w_q$ that describes encoded queries. Each character of w is a digit between 0 and 3.

Let x be the answer to the i-th query (we consider x to be 0 before all queries). Parameters of the i-th query are then computed as $a_i = (x \oplus w_i) \bmod 2$ and $c_i = \lfloor \frac{x \oplus w_i}{2} \rfloor \bmod 2)$. Here $\oplus$ is bitwise XOR operation and mod is modulo operation.

Output

Let ans_i be $lcs(s_0, s_1)$ after performing i-th query. Then output $\sum_{i=1}^{q} ans_i$.

Example

standard input	standard output
5 02102	4

Note

The encoded queries are $(0, 0)$, $(0, 1)$, $(1, 0)$, $(1, 0)$, $(1, 1)$. The strings after all queries are $s_0 = 01$ and $s_1 = 001$. Answers for queries are 0, 0, 1, 1, 2.

Problem G. Brawling

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

There are n people standing in a row. Each person is facing either left or right. People don't really like each other, so when two adjacent people are facing towards each other (that is, the one to the left is facing right, and the one to the right is facing left), one of them can kick the other one out of the row (any of two outcomes is possible). If several clashes are possible, any of them can happen next. What is the minimum possible number of people remaining in the row after a sequence of clashes?

Input

The only line of the input contains a string of characters "L" and "R", denoting people facing left and right respectively. The string is non-empty and its length does not exceed 10^6 characters.

Output

Output a single integer — the minimum possible number of people after a sequence of clashes.

Example

standard input	standard output
LRRLRL	2

Note

As the result of clashes, the following sequence of configurations is possible:

$$\texttt{LRRLRL} \rightarrow \texttt{LRLRL} \rightarrow \texttt{LRLRL} \rightarrow \texttt{LRRL} \rightarrow \texttt{LRL} \rightarrow \texttt{LR}$$

One can verify that in any scenario at least two persons survive, so the answer is 2.

Problem H. I Spy

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 mebibytes

This is an interactive problem.

Flatcity can be considered an infinite plane. A house is located at every integer point of the plane. There are n spy transmitters located in some of the houses. Your task is to find the houses containing transmitters using a mobile radar.

The radar is unable to track exact locations of transmitters. To find them, you can place the radar near any house and set its power level to any integer R. The radar is then able to determine the number of transmitters within euclidean distance $\sqrt{R}$.

You have to find the transmitters before the spies get away. Formally, you can ask at most 15 000 queries.

Interaction Protocol

First, interactor prints a single number n — the number of transmitters you have to find ($1 \leqslant n \leqslant 100$). It is guaranteed that all transmitters are at distinct locations, and their coordinates don't exceed 10^6 by absolute value. Then the following process is repeated.

Each time you want to make a query, you should print query parameters in the following format:

`?` x y R

where x and y denote the coordinates of the radar, and R is the square of its coverage radius (all numbers are integer, $|x|, |y| \leqslant 10^6$, $0 \leqslant R \leqslant 10^{18}$). After you print the parameters and flush the output, the interactor responds with a single integer c ($0 \leqslant c \leqslant n$) — the number of transmitters within the coverage radius of your position (inside the covered circle or on its border).

When you are ready to reveal the transmitters positions, you should print a single line in the following format:

`!` x_1 y_1 x_2 y_2 $\ldots$

Here x_i and y_i are the coordinates of the i-th point. All numbers should be integer, all tokens in this line should be separated by spaces. The order of points in the answer does not matter. After your program prints the answer, it should immediately terminate. Your answer is considered correct if your program made at most 15 000 `?`-queries and determined all transmitter positions correctly.

Example

standard input	standard output
2	
	? 0 0 4
1	
	? 0 0 5
2	
	? 0 0 3
0	
	? 1 0 1
1	
	? 2 0 0
1	
	? -1 0 4
1	
	? -1 0 2
0	
	? -1 1 1
1	
	! 2 0 -1 1

Problem I. Rage Minimum Query

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	512 mebibytes

Gleb proposed a problem for the contest:

Given an array of n integers, where n is up to 10^7, answer queries of kind "change i-th element to x" and report the global minimum after each query.

Gleb is a very experienced problemsetter. He knows that random tests check everything and there is no need to add any other. However, the contest is held tomorrow, and even random tests require some time to prepare them. That's why he asked you to generate tests by yourself.

You are given a linear pseudo-random generator specified by integers x_0, x_1, a, b and c. Here is the code of a procedure which yields random integers in C++:

```
uint32_t next() {
    uint32_t t = x0 * a + x1 * b + c;
    x0 = x1;
    x1 = t;
    return x0 >> 2;
}
```

Here `uint32_t` is an unsigned 32-bit integer type. Overflows are handled by taking the value modulo 2^{32}.

Same in Java (note that, although `int` is signed in Java, overflow is well-defined modulo 2^{32} as well):

```
int next() {
    int t = x0 * a + x1 * b + c;
    x0 = x1;
    x1 = t;
    return x0 >>> 2;
}
```

Initially, all elements of the array contain value $2^{31} - 1$. For each query, two numbers i and x are generated with two subsequent calls to `next()`, that is, with a call "i `= next()`" followed by "x `= next()`". This stands for the query "`a[i % n] = x`".

Suppose the global minimum after i-th query is s_i. Then you have to find the value $\sum_{i=1}^{q} s_i \cdot 10099^i \bmod 2^{32}$.

Note, that the only purpose of the above generator is to provide you some approximation of the random generator, you are not expected to find and exploit any its properties. The parameters of the generator are also chosen to make generated values indeed pseudo-random to the best of our knowledge and belief.

Input

The first and only line of input contains space-separated integers n, q, x_0, x_1, a, b, c ($1 \le n \le 10^7$, $1 \le q \le 5 \cdot 10^7$, $0 \le x_0, x_1, c < 2^{31}$, $2 \le a, b < 2^{31}$, a and b are odd). Here n is the size of the array, q is the number of queries, and other values are the parameters of the generator.

Output

Print the single integer: the value of $\sum_{i=1}^{q} s_i \cdot 10099^i \bmod 2^{32}$.

Examples

standard input	standard output
`5 5 1 3 5 7 9`	`2071650652`

`10000 10000 8800 5553535 314159275`
`271828181 987654321`
	`2620556972`

Note

In the first sample, the queries are:

1. `a[0] = 8`
2. `a[2] = 516`
3. `a[0] = 30276`
4. `a[4] = 1773386`
5. `a[3] = 103872918`

Problem J. Regular Cake

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

It's Bob's birthday soon, and Alice decided to make a cake for him.

Alice's cake must look like a regular n-gon, because it's Bob's n-th birthday. However, Bob likes number m a lot, so the cake should be packed in a regular-m-gon-shaped box. Finally, to make atmosphere more romantic, Alice will put a candle just at the center of the cake. That's why the candle will slightly bend the box, and for the sake of symmetry it'd be great if this happened at the center of the m-gon.

In other words, a regular-n-gon-shaped cake must fit into a regular-m-gon-shaped box, and their centers should coincide.

In fact, Alice has already cooked the cake, and its sides have length 1. Now she wants to know what is the least possible side length of an appropriate box. Help her!

Input

The only line of input contains two integers n and m ($3 \leqslant n, m \leqslant 10^9$).

Output

Print the least possible side length of the box in shape of a regular m-gon. Your answer is considered correct if the absolute or relative error doesn't exceed 10^{-6}.

Examples

standard input	standard output
`4 8`	`0.5411961001`
`8 4`	`2.4142135624`
`5 5`	`1.0000000000`

Problem K. Piecemaking

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 mebibytes

The civil war in Berland continues for five years already. The United Nation decided to end the bloodshed.

Berland consists of n cities, connected by $n-1$ bidirectional roads, forming a tree. The Purple Army occupies several cities which are listed in the set A, and the Cian Army occupies cities listed in the set B. The UN Assembly took an unexpected decision to end the war: they decided to destroy some roads in Berland, so that no city from the set A is connected (directly or undirectly) with a city from the set B. This way the enemies wouldn't be able to reach each other, and the fighting would stop. Destroying a road of length x kilometers requires x dollars.

Find the minimum sum of money neccessary for peacemaking in Berland.

Input

The first line of the input contains a positive integer n ($2 \le n \le 200\,000$) — the number of cities in Berland.

Next $n-1$ lines contain information about roads in the form u_i, v_i, w_i ($1 \le u_i, v_i \le n$, $1 \le w_i \le 10^9$) — indices of cities connected by the i-th road, and the length of this road in kilometers.

The next line contains a positive integer m ($1 \le m \le n$) — the number of cities occupied by the Purple Army, and m distinct integers $a_1, a_2, \ldots, a_m$ ($1 \le a_i \le n$) — indices of these cities.

The next line contains a positive integer k ($1 \le k \le n$) and k distinct integers $b_1, b_2, \ldots, b_k$ ($1 \le b_i \le n$) — the cities, occupied by the Cian army, in the similar format.

It is guaranteed that no city is occupied by both armies.

Output

Print the minimum possible number of dollars required to make it impossible to reach any city in set B from any city in set A.

Example

standard input	standard output

6
1 2 5
2 4 4
2 5 1
1 3 2
3 6 7
1 4
2 5 6
	3