Problem A. Alice and Bob (and string): Double Menace

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

From the creators of "Alice and Bob (and string)" and "Alice and Bob (and string) Strikes Back"!

Alice and Bob are playing a game. Initially they have a string s and its substring t. Each player's turn consists of adding an arbitrary letter c_l to the left of t *and* an arbitrary letter c_r to the right of T in such a way that t is still a substring of s. The player who can't make a valid move loses.

Alice moves first. Before she makes the first move, she has the right to choose the initial string t. Of course, Alice wants to cheat and will choose such a string t that will guarantee her victory (assuming both players act optimally), but she doesn't want Bob to suspect anything. Therefore, Alice decided to choose the k-th lexicographically smallest string among all possible winning initial strings t. Help Alice!

Input

The first line of input contains string s of lowercase English letters ($1 \le |s| \le 10^5$).

The second line contains integer k ($1 \le k \le 10^{10}$).

Output

If there are less than k suitable options for the string t, print "`no solution`". Otherwise, print the k-th lexicographically smallest one. If the answer is an empty string, print "`-`" instead.

Examples

standard input	standard output
abac 3	b
rndstr 1	-
abc 10	no solution

Note

Winning strings for $s =$ `abac` are {`-`, `a`, `b`, `ba`}.

Problem B. Bag of Bags

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

A mathematician goes to the shop every day and brings a bag from it. The bags are nice and practical, so mathematician wants to keep them for future usage. He also wants to keep his bags in order: big bags with big bags and small bags with small bags.

The bag brought on the i-th day (we'll just call it bag i) occupies volume a_i in folded state and volume b_i in unfolded state (naturally, $a_i < b_i$). The bag i fits into the bag j if $a_i < b_j$. Mathematician thinks that bags i and j are *equal* (and should be kept together) if the bag i fits into the bag j and vice versa.

Unfortunately, sometimes it happens that there are three bags i, j, k such that bags i and j are equal, and bags j and k are equal, but bags i and k are not! It scares mathematician very much because it contradicts with what he knows about the equality relation. If adding a new bag to his collection gives rise to a contradictory triple as described above, he throws the new bag out instead, otherwise he keeps it (and never throws it away afterwards).

Your task is to determine for each bag whether it was kept or thrown away.

Input

The first line contains an integer n — the number of bags ($1 \le n \le 3 \cdot 10^5$).

The next n lines describe the bags. The i-th of these lines contains two integers a_i and b_i — sizes of the bag i in folded and unfolded states respectively ($1 \le a_i < b_i \le 10^9$).

Output

Print n lines. The i-th line should contain the word "`KEPT`" if the mathematician keeps the bag i, and "`THROWN AWAY`" otherwise.

Example

standard input	standard output
10 1 4 3 5 6 8 7 9 1 2 6 7 4 7 5 7 5 8 9 10	KEPT KEPT KEPT KEPT THROWN AWAY THROWN AWAY THROWN AWAY THROWN AWAY KEPT KEPT

Problem C. Circle Union

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 mebibytes

An arrangement of several circles in the plane is *interesting* if there exists a point that lies inside or on the boundary of each circle. The *covered region* of an arrangement consists of all points that lie inside or on the boundary of at least one of the circle.

Consider n circles of radii $r_1, \ldots, r_n$ respectively. Find the largest possible area of the region covered by these circles in an interesting arrangement.

Input

The first line contains a single integer n ($1 \leq n \leq 10^4$).

The second line contains n integers $r_1, \ldots, r_n$ ($1 \leq r_i \leq 10^3$).

Output

Print a single real number — the largest possible covered area. Your answer will be considered correct if its absolute or relative error doesn't exceed 10^{-6}.

Example

standard input	standard output
3 10 9 8	726.4578311468

Problem D. Different Summands Counting

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Consider all ordered partitions of a positive integer n into m positive summands: $n = a_1 + a_2 + \ldots + a_m$. Let $f(a_1, a_2, \ldots, a_m)$ be the number of different integers among $a_1, a_2, \ldots, a_m$. Find the sum of $f(a_1, a_2, \ldots, a_m)$ over all ordered partitions of the number n, and print it modulo $998\,244\,353$.

Two ordered partitions $a_1 + a_2 + \ldots + a_m = n$ and $b_1 + b_2 + \ldots + b_m = n$ are considered different if there is an index $i \in \{1, 2, \ldots, m\}$ such that $a_i \neq b_i$.

Input

The only line of input contains two integers n and m ($1 \le n \le 10^{18}$, $1 \le m \le 500$, $m \le n$).

Output

Print the answer modulo $998\,244\,353$.

Examples

standard input	standard output
10 2	17
20 4	3413

Problem E. Emerging Tree

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

Consider a set $V = \{1, \ldots, n\}$ of n vertices, and a sequence of directed edges $e_1, \ldots, e_{n-1}$. Let $G_0, \ldots, G_{n-1}$ be a sequence of graphs such that G_0 is empty, and G_i is obtained by introducing the edge e_i into G_{i-1} for each $i = 1, \ldots, n-1$. It is guaranteed that G_{n-1} is a rooted tree with all edges directed away from the root.

Your task is to find a *suitable* permutation $p_1, \ldots, p_n$ of the set $\{1, \ldots, n\}$. Let $S_i(v) = \{p_u \mid u$ can be reached from v in $G_i\}$. A permutation $p_1, \ldots, p_n$ is called *suitable* if for any $i \in \{0, \ldots, n-1\}$ and for any $v \in V$ we have that $S_i(v)$ consists of consecutive numbers (that is, $S_i(v) = \{l, l+1, \ldots, r\}$ for some numbers l and r).

Input

The first line contains a single integer n ($2 \le n \le 10^6$).

The next $n-1$ lines describe the edges $e_1, \ldots, e_{n-1}$. The i-th of these lines contains two integers u_i and v_i — indices of the source and the target vertices of the edge e_i ($1 \le u_i, v_i \le n$).

It is guaranteed that adding all $n-1$ edges results in a rooted tree with edges directed away from the root.

Output

If there is no suitable permutation, print the only word "`No`" in the only line.

Otherwise, print "`Yes`" on the first line. On the second line print n integers $p_1, \ldots, p_n$ describing any suitable permutation.

Examples

standard input	standard output

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

`7`
`1 2`
`1 3`
`1 4`
`2 5`
`3 6`
`4 7`
	`No`

Problem F. Fast Travel Coloring

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

You are given a complete undirected graph with $7n$ vertices (here n is a positive integer). Your task is to paint its edges in n colors in such a way that for each pair of vertices and each color there is a path of at most two edges of this color connecting this pair of vertices. More formally, for each pair of vertices u, v and each color c at least one of the two options should hold:

- the edge between u and v has color c;
- there is a vertex w that both edges (u, w) and (w, v) have color c.

Input

The only line of input contains a positive integer n ($7 \le 7n \le 1000$).

Output

Let us number the colors from 1 to n. Let $c_{i,j}$ be 0 if $i = j$, and the color of the edge (i, j) in your coloring otherwise (in particular, in this case $c_{i,j} = c_{j,i}$). Print $c_{i,j}$ in $7n$ lines containing $7n$ numbers each.

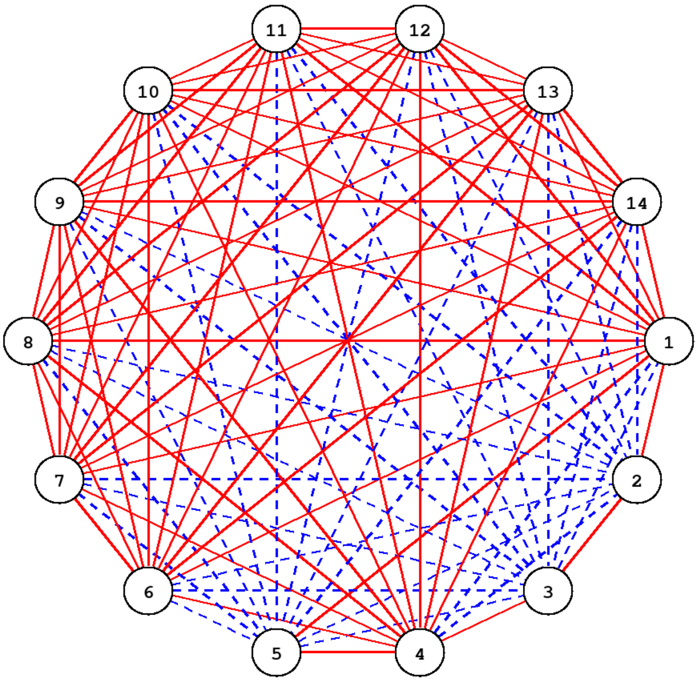
It is guaranteed that a solution exists.

Examples

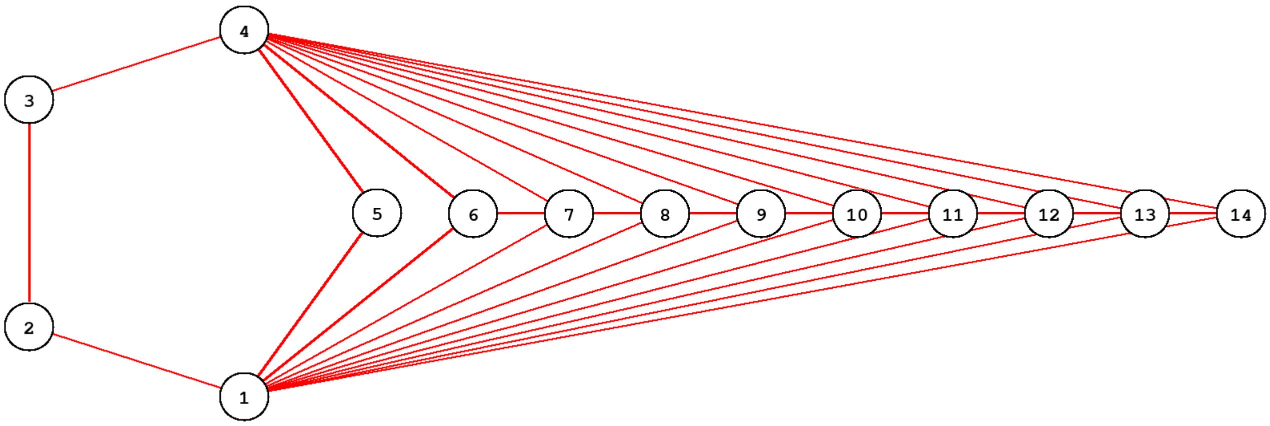
standard input	standard output
1	0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0
2	0 1 2 2 1 1 1 1 1 1 1 1 1 1 1 0 1 2 2 2 2 2 2 2 2 2 2 2 2 1 0 1 2 2 2 2 2 2 2 2 2 2 2 2 1 0 1 1 1 1 1 1 1 1 1 1 1 2 2 1 0 2 2 2 2 2 2 2 2 2 1 2 2 1 2 0 1 1 1 1 1 1 1 1 1 2 2 1 2 1 0 1 1 1 1 1 1 1 1 2 2 1 2 1 1 0 1 1 1 1 1 1 1 2 2 1 2 1 1 1 0 1 1 1 1 1 1 2 2 1 2 1 1 1 1 0 1 1 1 1 1 2 2 1 2 1 1 1 1 1 0 1 1 1 1 2 2 1 2 1 1 1 1 1 1 0 1 1 1 2 2 1 2 1 1 1 1 1 1 1 0 1 1 2 2 1 2 1 1 1 1 1 1 1 1 0

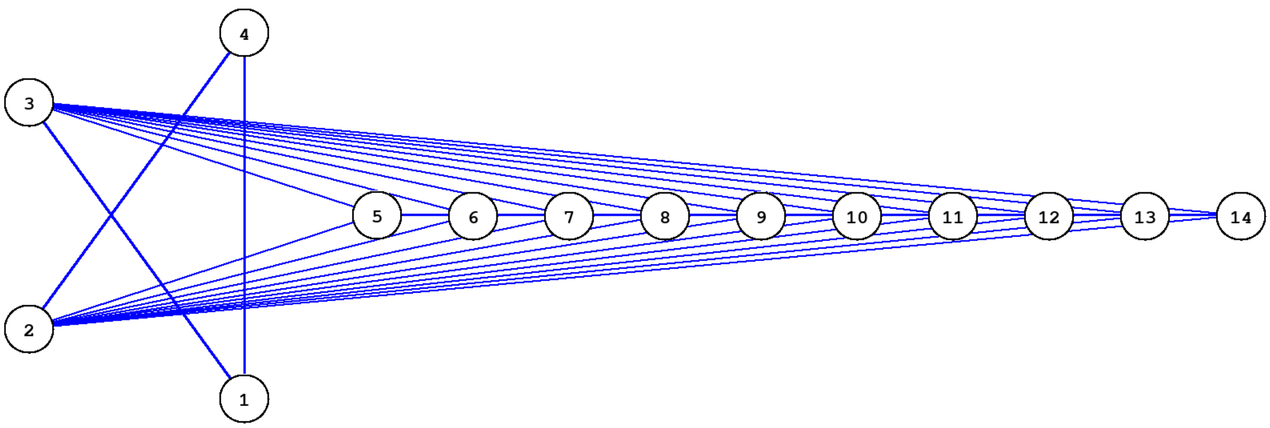
Note

The second sample test corresponds to the following coloring:



Here are two separate subgraphs for both colors:





Problem G. Gnutella Chessmaster

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

Alexander has recently achieved ridiculously high rating on Chessforces competition website. Alexander's coach challenged him with a difficult problem so that Alexander could truly prove his mettle.

Consider an $n \times n$ chessboard. A *bishop* is a chess piece that attacks all positions sharing a diagonal with it. A *non-attacking configuration* is an arrangement of several bishops on the chessboard such that no two bishops occupy the same position, and no bishop attacks any other.

Alexander has to count the number of non-attacking bishop configurations with exactly k bishops for each k from 1 to $2n-1$. Since the answers can be large, each number has to be computed modulo a completely random number $998\,244\,353$.

Input

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

Output

Print $2n-1$ integers. The k-th of these integers should be the number (modulo $998\,244\,353$) of non-attacking configurations of exactly k bishops on an $n \times n$ chessboard.

Examples

standard input	standard output
2	4 4 0
3	9 26 26 8 0
10	100 4380 110960 1809464 20014112 154215760 837543200 214861037 625796024 941559921 770927213 837612209 756883449 146369278 295974400 17275136 246784 1024 0

Problem H. Halve & Merge

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

You have an array $a = (a_1, \ldots, a_n)$ that initially contains a permutation of numbers 1 through n. You have to process queries of two types:

- "1 p" ($1 \le p \le n$): find a_p in the current array a;
- "2 p" ($1 \le p \le n-1$): replace a by the result of the function *merge* applied to arrays $(a_1, \ldots, a_p)$ and $(a_{p+1}, \ldots, a_n)$.

Function *merge* can be written in the following way.

```
func merge(var a as array, var b as array)
     var c as array
     while (a and b have elements)
          if (a[0] > b[0])
               add b[0] to the end of c
               remove b[0] from b
          else
               add a[0] to the end of c
               remove a[0] from a
     while (a has elements)
          add a[0] to the end of c
          remove a[0] from a
     while (b has elements)
          add b[0] to the end of c
          remove b[0] from b
     return c
```

Input

The first line contains two integers n and m — the length of the array and the number of queries ($2 \le n, m \le 2 \cdot 10^5$).

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

Each of the next m lines contains two integers t_i and p_i — the description of the i-th query ($t_i \in \{1, 2\}$, p satisfies the constraints given in the format description above).

Output

For each query of type 1, print the answer on a separate line.

Examples

standard input	standard output
4 3 4 3 2 1 2 1 2 1 1 2	1
5 7 4 3 5 2 1 2 4 2 1 1 3 1 1 2 4 1 4 1 5	3 1 3 5

Problem I. Interpolate

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

A *Zhegalkin polynomial* is a boolean function $f(x_1, \ldots, x_n)$ which is represented as follows:

$$f(x_1, \ldots, x_n) = \bigoplus_{S \subseteq \{1,2,\ldots,n\}} a_S \cdot \bigwedge_{i \in S} x_i,$$

where $\oplus$ and $\wedge$ stand for XOR and AND boolean operations respectively, XOR is taken over all subsets of variables, and $a_S \in \{0, 1\}$ for each subset S of $\{1, 2, \ldots, n\}$.

In this task you are given m sets of variable values $(v_{i1}, \ldots, v_{in})$ and m boolean values $y_i \in \{0, 1\}$. You have to construct a Zhegalkin polynomial with at most 9000 non-zero terms satisfying $f(v_{i1}, \ldots, v_{in}) = y_i$ for each $i = 1, 2, \ldots, m$.

Input

The first line contains two integers n and m — the number of variables and the number of given variable values ($1 \le n, m \le 2000$).

Each of the following m lines contains a string of n characters 0 or 1 representing the i-th set of variable values, followed by the integer y_i.

It is guaranteed that all sets of variable values are distinct and $y_i = 1$ for at least one set.

Output

Your polynomial has to contain at most 9000 terms having $a_S = 1$. For each such term print its corresponding subset S of variables as a string of n characters 0 or 1 such that i-th character equals 1 if $i \in S$ and 0 otherwise. You can output the terms in any order but there should be no repeating subsets.

If there are multiple possible answers, output any of them. If the solution does not exist, output -1.

It is guaranteed that if the solution exists, then the solution with at most 9000 terms S having $a_S = 1$ exists as well.

Examples

standard input	standard output
2 3 01 1 10 1 11 1	00
3 2 000 0 111 1	100 010 001

Note

One of the possible answers to the first sample is $f(x_1, x_2) = 1$.

In the second sample $f(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3$ is one of the possible answers.

Problem J. Jaw-Dropping Set

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

A subset A of the set $\{1, 2, 3, \ldots, n\}$ is called *interesting* if for any pair of different integers $x, y \in A$ neither x divides y nor y divides x.

An interesting subset A is called *amazing* if it has the maximum cardinality among all interesting subsets.

Finally, an amazing subset A is called *jaw-dropping* if it has the minimum sum of elements among all amazing subsets.

Given n, find the sum of elements in any jaw-dropping subset of $\{1, 2, 3, \ldots, n\}$.

Input

The first line contains integer t ($1 \le t \le 10^5$) — the number of test cases.

Each of the next T lines contains an integer n_i ($1 \le n_i \le 10^9$).

Output

Print T lines with answers for each test case.

Example

standard input	standard output
7 1 2 3 4 5 6 7	1 1 5 5 10 10 17

Problem K. Kingdom Connectivity

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

You are an engineer under the king's command. The king asked you to build a castle. The project is almost finished. It is already known that the castle is to contain n towers and m walls, each wall connecting some pair of towers. The towers can be viewed as points in the plane, and walls as segments connecting towers. The plan satisfies a number of sensible assumptions:

- no wall connects a tower to itself;
- there can be at most one wall between any pair of towers;
- different walls do not intersect anywhere except at the towers;
- no two towers have the same position;
- no wall can pass through any towers other than its endpoints.

Your task is to select some walls and build gates in them. After that, both sides of every wall of the castle must be accessible from the exterior through gates. Different landscape imposes that you must spend different amounts of money to build gates through different wall. What is the minimum possible amount of money needed to accomplish your task?

Input

First line contains two numbers n, m — the number of towers and and the number of walls respectively ($1 \leq n, m \leq 10^5$).

Each of the next n lines contains two integers x_i, y_i, denoting that the i-th tower is to be built at the point (x_i, y_i). Coordinates do not exceed 10^6 by absolute value.

Each of the next m lines contains three integers u_i, v_i, c_i ($1 \leq u_i, v_i \leq n$, $1 \leq c_i \leq 10^6$), denoting that there will be a wall between towers u_i and v_i and the price of building a gate through this wall is c_i.

Output

First, print a single number: minimum amount of money needed to build all necessary gates. Then print a number k, the number of gates to be built. Then print k pairs of numbers denoting pairs of towers which are connected by walls with gates according to your plan.

Examples

standard input	standard output
3 3 0 0 0 1 1 0 1 2 1 1 3 2 2 3 3	1 1 1 2
4 5 1 0 2 1 1 2 0 1 1 2 1 2 3 2 3 4 3 4 1 4 1 3 5	4 2 3 4 1 2