Problem A. Ideal Sets Strike Back

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

Of all the arts, the most essential still is the art of perfect balance.
„History of Novogireevka“, volume 1.

In the markets of the city of Novogireevka, all the sellers weigh their goods using a two-pan balance and set of masses each of which weighs an integer number of grams. The total weight of all the masses combined equals N grams. A set of masses is called a *perfect* set if any load with a weight equal to a integer number of grams from 1 to N can be balanced by a certain quantity of masses from this set, and in addition, if there is only one way to do this. The masses may be put on *both* sides of the balance, but it is not allowed to put two equal masses at both sides at the same time.

Two methods of weighing that differ only by replacing some of the masses with other masses of the same weight are considered to be identical. For example, for $N = 4$, there are two perfect sets: $(1, 1, 1, 1)$ and $(1, 3)$.

Find all the perfect sets of masses with a total weight of N grams which have the smallest number of masses.

Input

The first line contains one positive integer N — the total weight of all the masses in the set ($1 \le N \le 10^5$).

Output

Print two numbers on the first line: M — the number of perfect sets which have the smallest number of masses, and K — the number of masses in each of these sets. On the next M lines, print all the perfect sets of K masses with a total mass of N grams. The masses in a set must be listed in the order of non-descending weight. Output the sets in lexicographical order.

Examples

standard input	standard output

`4`	`1 2`
	`1 3`

`7`	`2 3`
	`1 1 5`
	`1 3 3`

Problem B. Solid Stacks

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

Consider a stack of K boxes. The weight of each box is a positive integer, and the sum of all those weights is N. Additionally we call stack *solid* if the weight of each box in the stack is not less than the total weight of all boxes above it.

For example, for $N = 7$, there are two *solid* stacks of $K = 3$ boxes: $7 = 1 + 2 + 4$ and $7 = 1 + 1 + 5$, where the first number denotes the weight of the upper box, the second one is the weight of the middle box, and the last one denotes the weight of the box which is placed on the ground.

How many different solid stacks can be built for given N and K?

Input

Input consists of one line with two integers: the total weight of stack N and the number of boxes in the stack K ($1 \le N \le 10^6$, $1 \le K \le 20$).

Output

Print the number of different solid stacks modulo $10^9 + 9$.

Examples

standard input	standard output
`1 1`	`1`
`1 2`	`0`
`7 3`	`2`

Problem C. Super-Binary

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

Recently, paleontologists have found the remains of a dinosaur *Linhenykus monodactylus* who has only one finger on each forelimb. Knowing that the decimal system emerged because humans have ten fingers, it is natural to suggest that these dinosaurs would have used the binary system. In fact, they used *super-binary* system in which there are three "digits": -1, 0 and 1. A *super-binary* representation of a number N is $N = 2^k a_k + \ldots + 2^2 a_2 + 2a_1 + a_0$, where each of the a_i equals to -1, 0 or 1 and $a_i \cdot a_{i+1} = 0$ for all $0 \leq i \leq k-1$.

For example, number 3 in this system could be written as **1 0 -1**, since $3 = 2^2 \cdot \mathbf{1} + 2 \cdot \mathbf{0} + (\mathbf{-1})$).

For a given integer in decimal form, find its super-binary representation.

Input

Input contains one integer N ($1 \leq N \leq 10^{18}$).

Output

Print one of possible super-binary representations of N — space-separated integers $a_k, \ldots, a_1, a_0$ where a_k is the leftmost non-zero digit in the representation, and a_0 is the rigthmost digit (the one multiplied by 1).

Examples

standard input	standard output
`1`	`1`
`3`	`1 0 -1`

Problem D. Universal Number (Division 1 Only!)

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

In the Italian city of Michel-Angelo lived a famous sculptor. “All perfect things in the world can be measured by *universal numbers*,” he said. According to him, a *universal* number for a given N is the smallest positive integer $U(N)$ such that any natural number from 1 to N can be obtained by removing some digits (possibly none) from its decimal representation. The order of the remaining digits must be preserved.

For example, for $N = 10$, $U(10)$ is equal to 1 023 456 789.

Your task is to construct the universal number for a given positive integer N.

Input

Input file contains a positive integer N ($1 \le N < 10^{100\,000}$).

Output

Print the answer: the appropriate universal number $U(N)$.

Examples

standard input	standard output
1	1
2	12
10	1023456789

Problem E. Socks

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	256 mebibytes

A luxury boutique sells one model of socks in different sizes. All of them are stacked for the display in a row. The boutique owner orders the employee to order all the socks by size in non-decreasing order. However, it is nigh impossible to put socks in the middle of the showcase. So, the only available operations are the following: get any pair of socks from the showcase and put it either at the beginning of the row or at the end of the row.

Find the mininum number of those operations for the employee to complete the task.

Input

The first line of input contains one integer N ($1 \le N \le 2 \cdot 10^5$) — the number of pair of socks placed on the boutique's showcase. The next line contains N integers ($1 \le A_i \le 10^9$) — the size for each pair of socks, listed in the original order.

Output

Print one integer: the minimal possible number of operations.

Examples

standard input	standard output
3 1 3 2	1
4 2 1 3 2	2
2 2 1	1

Problem F. The Big Test

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	256 mebibytes

On the control panel of a nuclear power plant, there are N switches. Each of these switches has two states: “on” and “off”. After toggling i-th switch, a new mode of operation is set. This mode is defined by the number i and the positions of all switches.

The testing crew regularly checks the functioning of control systems. In particular, the testers must observe the operation of the station in all possible modes.

Help the testing crew to perform this work as fast as possible, e.g., toggling the minimum possible number of switches.

Input

Input contains one integer N ($1 \le N \le 18$).

Output

On the first line of output, print one integer M: the minimum possible number of switches which have to be toggled. On the next line, print M integers a_i ($1 \le a_i \le N$): the numbers of switches to toggle, in appropriate order.

Examples

standard input	standard output

`1`	`2`
	`1 1`

`2`	`8`
	`2 1 2 2 1 2 1 1`

Problem G. Good Morning, Colleague!

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

In the ICL head office, tables are arranged so that there is a single way from the front door to each workplace without self-intersections. The office can be represented as a tree with workplaces in some vertices, and the length of each edge is equal to 1 meter. When employees come to work in the morning, each of them should come to greet all who came before him and wish a good day, and only then go to his place.

However, the office manager does not really like this tradition. He believes that employees are spending too much worktime on personal greetings, so he has decided to prepare a memo to the boss. In Annex 15 to the memo, the manager is planning to give some information about the total distance each employee travels in the morning from the front door to his workplace. Fortunately, the order of staff arrival is already known, as is the fact that each of them chooses the shortest way for morning greetings and manages to take his place in the office until the next employee arrival.

Input

The first line of input contains an integer N ($2 \le N \le 10^5$) — the number of vertices in the tree describing the office. Each of the next $N-1$ lines contains two different integers a and b ($1 \le a, b \le N$) — the numbers of vertices connected by an edge.

The next line contains an integer M ($1 \le M \le N-1$) — the number of employees working in the office. The next line contains M pairwise distinct integers v_i ($2 \le v_i \le N$) — the numbers of vertices with workplaces. They are listed in the order of staff arrival. The front door used by all employees to enter office is placed at vertex 1.

Output

Print M integers: the distances in meters traveled by each employee in order of their numbering.

Examples

standard input	standard output

7
1 2
1 3
3 4
3 5
3 6
6 7
6
6 4 2 7 3 5
	2 4 7 7 9 10

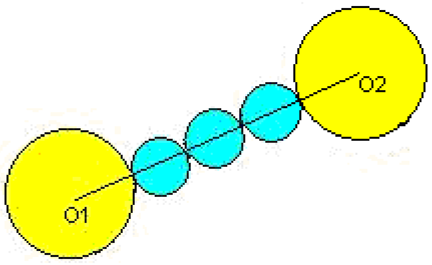
Problem H. Fortress (Division 1 Only!)

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

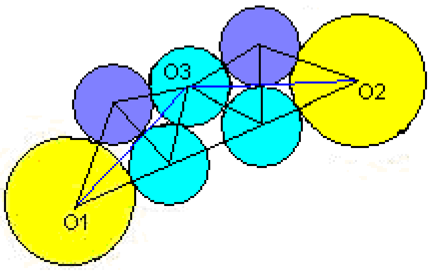
In middle ages, Rus' fortresses were encircled by wooden fences. But, unfortunately, sometimes barabians burned the fences during their savage sieges. Only the thickest logs in the fence remained after fires. Once, citizens of a fortress decided to rebuild the fence.

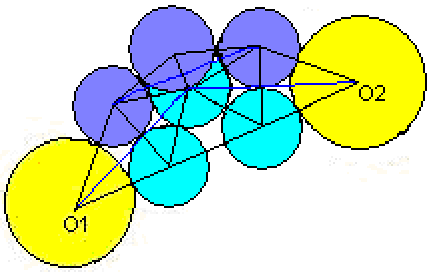
From a bird's view, the fortress looks like several disks. These are the old logs remained after the fires, which are placed vertically. The radii of these logs are equal to R. By connecting the centers of these disks in the order they are given, one can obtain a convex polygon. The builders will then rebuild the fence as follows:

- The builders use new logs which are also seen as disks. All of these disks have radii equal to R_0, and it is known that $R_0 \leq R$.
- For each segment that connects consecutive disk centers, the builders put a log edge-to-edge with both existing logs on the ends of the segment. Then another two log edge-to-edge with the new two logs, and so on until it is possible to maintain the symmetry with respect to the perpendicular bisector of the segment. All new log centers should lie on the segment.
- If the addition of new logs as described above ended with an edge-to-edge connection, builders celebrated their success and went to build another edge.



- However, if a gap remained in the middle of the segment, another log or two are put on the external side of the fortress, once again maintaining symmetry with respect to the perpendicular bisector.
- In the latter case, the fence polygon (with log centers as its vertices) could have turned out to be non-convex. To fix this, the builders select any two adjacent edges of the fence which make it non-convex, and reflect them symmetrically with respect to the segment connecting two distant ends of these two edges. This process continues until the fence is a convex polygon. Note that the length of the fence does not change during these reflections.





To calculate the exact amount of paint necessary to cover all the exterior of the fence, you need to calculate its external perimeter. It is guaranteed that the initial logs have no intersections.

Input

The first line of input holds a single number N ($1 \le N \le 100$) — the number of old logs before the rebuilding.

Each of the next N lines hold two integers X_i, Y_i ($0 \le X_i, Y_i \le 1000$) — the coordinates of centers of the old logs in clockwise order.

The last line holds two integers: R_0 ($1 \le R_0 \le 10$) — radius of each of the new logs and R ($R_0 \le R \le 10$) — radius of each of the old logs.

Output

Output the minimum possible number of new logs necessary to rebuild the fence and the external perimeter of the resulting fence with absolute or relative error no more than 10^{-6}.

Examples

standard input	standard output

4
0 0
0 6
6 6
6 0
1 2
	4
	50.265482

4
0 0
0 6
6 6
6 0
1 2
	4
	50.265482

Problem I. Bacteria

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

Famous biologist M. D. Scalpel is studying a bacterium found on the bottom of Mariana Trench.

The bacterium is always in one of the two states: either "at rest" or "agitated".

When the bacterium is agitated, it sends an electromagnetic impulse to the adjacent bacteria.

If any bacterium receives an impulse from two adjacent bacteria, it comes to rest. If the impulse is received from one adjacent bacterium, it becomes agitated. The state changes regardless of the initial state.

A bacterium that received no impulse comes to rest for the next second.

M. D. put N bacteria in line, some of the bacteria were agitated.

Each second, M. D. writes down the list of agitated bacteria in the line.

Help M. D. to calculate the status of bacteria in T seconds from the beginning of the experiment. M. D. has no time to wait for T seconds to pass.

Input

The first line holds two integers T and N ($0 \le T \le 10^{18}$, $1 \le N \le 10^5$) — the number of seconds before the target moment and the nubmer of bacteria in the line. The next line holds N characters "`0`" and "`1`" representing the initial state of bacteria. Here, "`1`" means that the respective bacterium is agitated, and "`0`" means that it is at rest.

Output

Output the state of bacteria in T seconds from the beginning of the experiment in the same format as the initial bacteria setup is given.

Examples

standard input	standard output
2 5 00100	10001
0 1 1	1
1 5 01110	11011

Problem J. BST

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	256 mebibytes

A *binary search tree* is a binary tree that satisfies the following conditions:

1. both left and right subtrees of each node are binary search trees,
2. for any node X, each node in the left subtree of X has value less than the value in the node X,
3. for any node X, each node in the right subtree of X has value greater than the value in the node X.

You can apply the following operations to a BST:

- `ADD` X — finds a place for the key value X and adds it as a new node. If the initial tree is empty, a tree with a single node with key X is created. If the node X already exists in the tree, the operation returns `FALSE` and does nothing. Otherwise, the operation returns `TRUE`.
- `FIND` X — searches for a node with the key value X and returns `TRUE` in case of success, `FALSE` otherwise.
- `REMOVE` X — searches for a node with the key value X and returns `TRUE` in case of success, `FALSE` otherwise. Then removes the node from the BST. The deletion goes as follows. If the node has no children, the node is just removed from the tree. If the node has one child, the node is removed and replaced with its child. If the node has two children, the least (leftmost) value of the right subtree replaces the value in the node being removed. The least-valued node of the right subtree is then removed.

The height of the tree root equals to 1. The height of any node in the tree is less by 1 than the heights of the roots of its subtrees.

The *cost* of each operation is the maximum height of a node that is processed during the operation:

- For `ADD` — the height of the node with value X after the execution.
- For `FIND` — the height of the node with value X, or the height of the node such that the fact that this node has no child means that a node with key X does not exist in the BST. The cost of `FIND` on an empty tree is 1.
- For `REMOVE` — the height of the node which will be removed, or the height of the node such that the fact that this node has no child means that a node with key X does not exist in the BST. The cost of `REMOVE` on an empty tree is 1.

Your task is to apply all operations given in the input file and output their return values and costs. Initially, the BST is empty.

Input

The first line of input file holds a single integer N ($1 \le N \le 10^5$) — the nubmer of operations on the tree. Each of the next N lines holds one operation from the set {`ADD`, `FIND`, `REMOVE`} separated by space from its integer argument X ($-10^9 \le X \le 10^9$).

Output

For each operation in the input file, output a single line containing the answer (`TRUE` or `FALSE`) and the cost of that operation, separated by a space.

Examples

standard input	standard output
3 ADD 6 FIND 6 REMOVE 6	TRUE 1 TRUE 1 TRUE 1
11 REMOVE 7 FIND 8 ADD 9 ADD 10 ADD 8 FIND 9 FIND 8 FIND 10 FIND 11 REMOVE 9 FIND 8	FALSE 1 FALSE 1 TRUE 1 TRUE 2 TRUE 2 TRUE 1 TRUE 2 TRUE 2 FALSE 2 TRUE 2 TRUE 2

Problem K. Adventurous

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

I used to be an adventurer like you. But then I took an arrow in the knee.
Skyrim

В поле бес нас водит, видно,
Да кружит по сторонам.
Пушкин А.С.

We came out of a forest and stopped at the point A on the road #1. Adventurers! We were eager to step on each and every unknown road. From the point A, we moved straight-perpendicular to the road #2. Adventurousness was forcing us to move faster. As soon as we reached road #2, we moved straight to the road #3, and then on and on... We walked through a hundred or two of these straight infinite roads... Finally, we reached the road #n, and moved to return to the road #1, where suddenly we found ourselves at the point A. The best of us managed to calculate the coordinates of point A on the road #1.

Can you do the same?

Input

The first line holds a single integer n — the number of roads we faced during the journey ($2 \leq n \leq 10\,000$). Each i-th of the next n lines holds coordinates X_{i1}, Y_{i1} and X_{i2}, Y_{i2} of two distinct points on the i-th road. All coordinates are integers not exceeding 100 by absolute value. Consecutive numbers on a line are separated by spaces. It is guaranteed that at least two roads are not parallel and do not coincide.

Output

Output coordinates of point A on the road #1, separated by a space. The answer is correct if relative or absolute error does not exceed 10^{-5}. You can output any solution in case of multiple solutions. Output -1 if the solution doesn't exist.

Examples

standard input	standard output

3
-1 0 1 0
1 0 0 1
0 1 -1 0
	0.00000 0.00000

2
0 0 2 0
1 1 0 2
	2.00000 0.00000

Problem L. ChessMad

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

You are given a chessboard of size $N \times N$ and some chess pieces on the board. You play a single player game. In this game, there are only four types of pieces: a pawn, a knight, a rook and a king. Each square of the chessboard is either empty or occupied with a chess piece.

The initial setup and rules are sunk into oblivion, the legend holds only the last paragraph of the rules which describes the scoring. It says:

"To calculate the score of the game, calculate the cumulative strength of all pieces on the board. Pawn adds a single point to the total strength. Knight adds two points, rook — three, king — four points. If the total number of pieces on te board is greater of equal to the number of empty squares on the board, the player wins and earns the number of points equal to the total strength of all pieces on the board. Otherwise, the player loses the game."

Based on the given information about the board, calculate the result of the game if it ends immediately.

Input

The first line of input holds a single positive integer $N < 14$. The next N lines hold N space-separated numbers each — the description of the board setup. In these lines, `0` (zero) encodes an empty square, `1` represents a pawn, `2` — a knight, `3` — a rook and `4` means that a king is on the square.

Output

Output a single integer — the number of points earned by the player. Output `-1` in case of a loss.

Examples

standard input	standard output

7
0 0 0 1 0 0 0
0 4 0 1 0 0 0
0 0 0 1 0 0 0
1 1 1 1 1 1 1
0 0 0 1 0 0 0
0 0 0 1 0 3 0
0 0 0 1 0 0 0
	-1

5
0 2 4 2 3
1 0 1 0 1
0 0 0 0 0
1 1 0 1 1
3 2 4 0 3
	30