icpc international collegiate programming contest

ICPC North America Regionals 2019

ICPC Southeast USA

Regional Contest

Official Problem Set

Division 1

icpc.foundation

Division 1

Hosted by:

College of Charleston
Florida International University
Kennesaw State University
University of West Florida

Carryless Square Root

Time limit: 1 second

Carryless addition is the same as normal addition, except any carries are ignored (in base 10). Thus, $37 + 48$ is 75, not 85.

Carryless multiplication is performed using the schoolbook algorithm for multiplication, column by column, but the intermediate sums are calculated using *carryless* addition. Thus:

$$9 \cdot 1234 = 9000 + (900 + 900) + (90 + 90 + 90) + (9 + 9 + 9 + 9)$$
$$= 9000 + 800 + 70 + 6 = 9876$$

$$90 \cdot 1234 = 98760$$

$$99 \cdot 1234 = 98760 + 9876 = 97536$$

Formally, define c_k to be the k^{th}digit of the value c. If $c = a \cdot b$ then

$$c_k = \left[\sum_{i+j=k} a_i \cdot b_j \right] mod\ 10$$

Given an integer $\boldsymbol{n}$, calculate the smallest positive integer $\boldsymbol{a}$ such that $\boldsymbol{a} \cdot \boldsymbol{a} = \boldsymbol{n}$ in *carryless* multiplication.

Input

The input consists of a single line with an integer $\boldsymbol{n}$ ($1 \leq \boldsymbol{n} \leq 10^{25}$).

Output

Output the smallest positive integer that is a *carryless* square root of the input number, or -1 if no such number exists.

Sample Input	Sample Output
6	4
149	17
123476544	11112
15	-1

Computer Cache

Time limit: 5 seconds

Your computer has a cache consisting of n different addresses, indexed from 1 to n. Each address can contain a single byte. Initially all cache bytes start off with the value zero.

You have m different pieces of data you want to store. Each piece of data is a byte array. The pieces of data may have different lengths, and any particular piece of data may be stored multiple times at different locations.

You are going to do q operations on your computer. There are three types of operations:

$1\ i\ p$ Load piece of data i starting at position p in the cache. This overwrites any previously stored value in the cache. It is guaranteed that this is a valid operation (i.e., the data will not go beyond the end of the cache). It is possible for multiple versions of some data to be loaded in multiple positions of the cache at once.

$2\ p$ Print the byte that is stored in cache address p.

$3\ i\ l\ r$ Increment the l^{th} through r^{th} bytes in the i^{th} piece of data. Since these are bytes, you must increment modulo 256. This does not affect values that are already loaded in the cache. It only affects the piece of data, and future loads of the piece of data.

Input

The first line of input contains three integers n, m and q ($1 \le n, m, q \le 5 \cdot 10^5$), where n is the size of the computer's cache, m is the number of pieces of data, and q is the number of operations.

Each of the next m lines describes a piece of data, as a sequence of space separated integers. The first integer on the line, k_i ($1 \le k_i, \sum k_i \le 5 \cdot 10^5$), indicates the number of integers to follow. Each of the next k_i integers x ($0 \le x \le 255$) are the contents of the piece of data.

Each of the next q lines will have two, three, or four space-separated integers representing an operation, in order, as described above. Either:

$$1\ i\ p \quad \text{or} \quad 2\ p \quad \text{or} \quad 3\ i\ l\ r$$

With ($1 \le i \le m$), ($1 \le p \le n$), and ($1 \le l \le r \le k_i$). There is guaranteed to be at least one 2 operation.

Output

For each 2 operation, output the integer value of cache location p, one per line.

Sample Input	Sample Output
5 2 10 3 255 0 15 4 1 2 1 3 2 1 1 2 2 1 1 1 2 1 2 4 3 1 1 2 2 1 1 1 2 2 2 2 5	0 255 1 255 0 3

Elven Efficiency

Time limit: 5 seconds

Like many creatures featured in programming problems, the animals of the forest love playing games with stones. They recently came up with a game to teach the younger animals about divisibility. In this game, each animal starts with a pile of stones. At the start of the game, a series of numbers is called out. For each number that is called, every animal whose number of stones is divisible by the called number scores a point. At the end of the game, the animal with the most points wins.

Emma the forest elf has watched the forest animals play this game many times, and has grown tired of watching the winning animal gloat about how many points they scored. To prevent this from happening, she plans to meddle in the next game the animals play to ensure that no animal scores any points. She plans to wait atop a nearby tree, and keep track of how many stones each animal has. Each round, if an animal is about to score a point, she can toss a stone into that animal's pile, increasing their number of stones by one. The tossed stone stays in that pile for the rest of the game. Throughout the course of the game, she may need to toss several stones into the same pile. But stones are heavy, and she wants to carry as few as possible to the top of her hideout tree. She already knows how many stones each of the $\boldsymbol{n}$ animals will start with, as well as the number to be called out in each of the $\boldsymbol{m}$ rounds of the game, but she wants you to calculate the minimum total number of stones she will have to throw to ensure that no animal scores any points.

Input

The first line of input contains two space-separated integers $\boldsymbol{n}$ and $\boldsymbol{m}$ ($1 \leq \boldsymbol{n}, \boldsymbol{m} \leq 10^5$), where $\boldsymbol{n}$ is the number of animals, and $\boldsymbol{m}$ is the number of rounds of the game.

Each of the next $\boldsymbol{n}$ lines contain a single integer $\boldsymbol{a}$ ($1 \leq \boldsymbol{a} \leq 3 \cdot 10^5$), which are the numbers of stones held by each animal.

Each of the next $\boldsymbol{m}$ lines contain a single integer $\boldsymbol{k}$ ($2 \leq \boldsymbol{k} \leq 3 \cdot 10^5$), which are the numbers called out, in order.

Output

Output a single integer, which is the minimum number of stones that Emma must use to prevent any and all animals from scoring any points.

Sample Input	Sample Output
3 5 10 11 12 2 11 4 13 2	12

Swap Free

Time limit: 1 second

A set of words is called *swap* free if there is no way to turn any word in the set into any other word in the set by swapping only a single pair of (not necessarily adjacent) letters.

You are given a set of n words that are all anagrams of each other. There are no duplicate letters in any word. Find the size of the largest swap free subset of the given set. Note that it is possible for the largest swap free subset of the given set to be the set itself.

Input

The first line of input contains a single integer n ($1 \leq n \leq 500$).

Each of the next n lines contains a single word w ($1 \leq |w| \leq 26$).

Every word contains only lower-case letters and no duplicate letters. All n words are unique, and every word is an anagram of every other word.

Output

Output a single integer, which is the size of the largest *swap free* subset.

Sample Input	Sample Output
6 abc acb cab cba bac bca	3
11 alerts alters artels estral laster ratels salter slater staler stelar talers	8
6 ates east eats etas sate teas	4

Fixed Point Permutations

Time limit: 1 second

A permutation of size n is a list of integers $(p_1, p_2, \ldots, p_n)$ from 1 to n such that each number appears exactly once.

The number of *fixed points* of a permutation is the number of indices i such that $p_i = i$.

Given three numbers n, m, and k, find the k^{th} lexicographically smallest permutation of size n that has exactly m *fixed points* (or print `-1` if there are fewer than k permutations that satisfy the condition).

Input

The single line of input contains three space-separated integers

$$n\ (1 \le n \le 50) \quad m\ (0 \le m \le n) \quad k\ (1 \le k \le 10^{18})$$

where n is the size of the permutations, m is the number of desired *fixed points*, and the output should be the k^{th} lexicographically smallest permutation of the numbers 1 to n that has exactly m *fixed points*.

Output

Output the desired permutation on a single line as a sequence of n space-separated integers, or output `-1` if no such permutation exists.

Sample Input	Sample Output
3 1 1	1 3 2
3 2 1	-1
5 3 7	2 1 3 4 5

Interstellar Travel

Time limit: 5 seconds

You are planning to travel in interstellar space in hope of finding habitable planets. You have already identified $\boldsymbol{n}$ stars that can recharge your spaceship via its solar panels. The only work left is to decide the orientation of the spaceship that maximizes the distance it can travel.

Space is modeled as a 2D plane, with the Earth at the origin. The spaceship can be launched from the Earth in a straight line, in any direction. A star can provide enough energy to travel a certain distance if the spaceship is launched at a certain angle with the X-axis. If the angle is not perfectly aligned, then the spaceship gets less energy.

Each star has a distance $\boldsymbol{t}$ that the spaceship can travel if launched at an angle $\boldsymbol{a}$ with the X-axis, and a multiplier $\boldsymbol{s}$ which indicates the decay of distance as angles diverge from $\boldsymbol{a}$. if the launch direction makes an angle of $\boldsymbol{b}$ with the X-axis, then the spaceship gets enough energy to travel a distance of:

$$\mathbf{max}\,(0, \boldsymbol{t} - \boldsymbol{s} \cdot dist[\boldsymbol{a}, \boldsymbol{b}])$$

where $dist[\boldsymbol{a}, \boldsymbol{b}]$ is the minimum non-negative radians needed to go between angles $\boldsymbol{a}$ and $\boldsymbol{b}$. The total distance that the spaceship can travel is simply the sum of the distances that each star contributes.

Find the maximum distance that the starship can travel.

Input

The first line of input contains a single integer $\boldsymbol{n}$ ($1 \leq \boldsymbol{n} \leq 10^5$), which is the number of stars.

Each of the next $\boldsymbol{n}$ lines contains three space-separated real numbers

$$\boldsymbol{t}\ (0.0 < \boldsymbol{t} \leq 1000.0) \quad \boldsymbol{s}\ (0.0 \leq \boldsymbol{s} \leq 100.0) \quad \boldsymbol{a}\ (0.0 \leq \boldsymbol{a} < 2\pi)$$

where $\boldsymbol{t}$ is the greatest distance that star's energy can support, $\boldsymbol{s}$ is that star's decay multiplier, and $\boldsymbol{a}$ is the best angle for achieving the greatest distance from energy from that star.

Output

Output a single real number, which is the maximum distance that the spacecraft can travel. Your answer must be accurate to within 10^{-6} absolute or relative error.

Sample Input	Sample Output
2 100 1 1 100 1 1.5	199.500000
4 100 1 0.5 200 1 1 100 0.5 1.5 10 2 3	405.500000

Jumping Path

Time limit: 10 seconds

You are given a rooted tree where each vertex is labeled with a non-negative integer.

Define a *Jumping Path* of vertices to be a sequence of vertices $v_1, v_2, \ldots, v_k$ where v_i is an ancestor of v_j for all $i < j$. Note that v_i is an ancestor of v_{i+1}, but not necessarily the parent of v_{i+1} (hence the *jumping* part of a *jumping path*).

Compute two quantities:

- The length (number of vertices) of the longest *jumping path* where the labels of the vertices are nondecreasing.
- The number of *jumping paths* of that length where the labels of the vertices are nondecreasing.

Input

The first line of input contains an integer $\boldsymbol{n}$ ($1 \leq \boldsymbol{n} \leq 10^6$), which is the number of vertices in the tree. Vertices are numbered from 1 to $\boldsymbol{n}$, with vertex 1 being the tree root.

Each of the next $\boldsymbol{n}$ lines contains an integer $\boldsymbol{x}$ ($0 \leq \boldsymbol{x} \leq 10^6$), which are the labels of the vertices, in order.

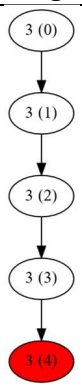
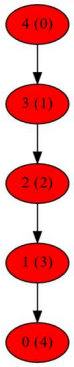
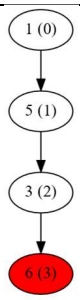
Each of the next $\boldsymbol{n} - 1$ lines contains an integer $\boldsymbol{p}$ ($1 \leq \boldsymbol{p} \leq \boldsymbol{n}$), which are the parents of nodes 2 through $\boldsymbol{n}$, in order.

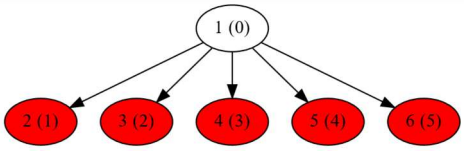
It is guaranteed that the vertices form a single tree, i.e., they are connected and acyclic.

Output

Output a single line with two integers separated by a space.

The first integer is length of the longest *jumping path* where the labels of the vertices are nondecreasing. The second integer is the number of *jumping paths* of that length where the labels of the vertices are nondecreasing. As the second integer may be large, give its value modulo 11092019.

Sample Input	Sample Output	Diagram
5 3 3 3 3 3 1 2 3 4	5 1	
5 4 3 2 1 0 1 2 3 4	1 5	
4 1 5 3 6 1 2 3	3 2	

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

Levenshtein Distance

Time limit: 1 second

The *Levenshtein Distance* between two strings is the smallest number of simple one-letter operations needed to change one string to the other. The operations are:

- Adding a letter anywhere in the string.
- Removing a letter from anywhere in the string.
- Changing any letter in the string to any other letter.

Given a specific alphabet and a particular query string, find all other unique strings from that alphabet that are at a *Levenshtein Distance* of 1 from the given string, and list them in alphabetical order, with no duplicates.

Note that the query string must not be in the list. Its *Levenshtein Distance* from itself is 0, not 1.

Input

Input consists of exactly two lines. The first line of input contains a sequence of unique lower-case letters, in alphabetical order, with no spaces between them. This is the alphabet to use.

The second line contains a string s ($2 \leq |s| \leq 100$), which consists only of lower-case letters from the given alphabet. This is the query string.

Output

Output a list, in alphabetical order, of all strings which are a *Levenshtein Distance* of 1 from the query string s. Output one word per line, with no duplicates.

Sample Input	Sample Output
eg egg	eeg eegg eg ege egeg egge eggg gegg gg ggg

Maze Connect

Time limit: 5 seconds

Given an orthogonal maze rotated 45 degrees and drawn with ASCII forward and backward slash characters (see below), determine the minimum number of walls that need to be removed to ensure it is possible to escape out of the (possibly disconnected) maze from every cell without going through a wall.

```
/\
\/
```

The above maze has only a single cell fully enclosed. Removing any wall will provide an escape route to the outside.

```
/\..
\.\.
.\/\
..\/
```

The above maze has two enclosed areas. Two walls need to be removed to connect all cells to the outside.

```
/\/\/\/\/\/\/\/\/\
\../\.\/././/\/\/\
/./\.././\/\.\/\/\/\
\/\/\.\/\/./\/..\../
/\/./\/\/./..\/\/..\
\.\.././\.\/\/./\.\/
/.../\../..\/./...//\
\/\/\/\/\/\/\/\/\/\/
```

To make every cell in the above maze accessible from the outside, 26 walls need to be removed.

Input

The first line of input contains two numbers, r and c ($1 \leq r, c \leq 1000$), which are the number of rows (r) and columns (c) in the maze.

Each of the next r lines contains a string with exactly c characters, consisting of '.', '/', or '\' only.

Define an odd (even) square in the grid of characters as one where the sum of the x and y coordinates is odd (even). Either all forward *slashes will* be in the odd squares and *all backwards* slashes in the even squares, or vice versa.

Output

Output a single integer, which is the least number of walls that need to be removed so that escape is possible from every cell in the maze.

Sample Input	Sample Output
2 2 /\ \/	1
4 4 /\.. \.\. .\/\ ..\/	2
2 2 \/ /\	0
8 20 /\/\/\/\/\/\/\/\/\/\ \../\.\/././/\/\/\/ /./\.././\/\.\/\/\/\ \/\/\.\/\/./\/..\../ /\/./\/\/./..\/\/..\ \.\.././\.\/\/./\.\/ /.../\../..\/./.../\ \/\/\/\/\/\/\/\/\/\/	26

One of Each

Time limit: 2 seconds

You are given a sequence of n integers $X = [x_1, x_2, \ldots, x_n]$ and an integer k. It is guaranteed that $1 \leq x_i \leq k$, and every integer from 1 to k appears in the list X at least once.

Find the lexicographically smallest subsequence of X that contains each integer from 1 to k exactly once.

Input

The first line of input contains two integers n and k ($1 \leq k \leq n \leq 2 \cdot 10^5$), where n is the size of the sequence, and the sequence consists only of integers from 1 to k.

Each of the next n lines contains a single integer x_i ($1 \leq x_i \leq k$). These are the values of the sequence X in order. It is guaranteed that every value from 1 to k will appear at least once in the sequence X.

Output

Output a sequence of integers on a single line, separated by spaces. This is the lexicographically smallest subsequence of X that contains every value from 1 to k.

Sample Input	Sample Output
6 3 3 2 1 3 1 3	2 1 3
10 5 5 4 3 2 1 4 1 1 5 5	3 2 1 4 5

Windmill Pivot

Time limit: 10 seconds

Consider a set of points P in the plane such that no 3 points are collinear. Construct a *windmill* as follows:

- Choose a point $p \in P$ and a starting direction such that the line through p in that direction does not intersect any other points in P. Draw that line (Note: *line*, NOT *ray*).
- Rotate the line clockwise like a windmill about the point p as its pivot until the line intersects another point $q \in P$. Designate that point q to be the new pivot, and then continue the rotation. This is called *promoting* point q.
- Continue this process until the line has rotated a full $360°$, returning to its original direction (it can be shown that the line will also return to its original position after a $360°$ rotation).

During this process, a given point in P can be a pivot multiple times. Considering all possible starting pivots and orientations, find the maximum number of times that a single point can be *promoted* during a single $360°$ rotation of a windmill. Note that the first point is a pivot, but not *promoted* to be a pivot at the start.

Input

The first line of input contains a single integer $\boldsymbol{n}$ ($2 \leq \boldsymbol{n} \leq 2000$), which is the number of points $p \in P$.

Each of the next $\boldsymbol{n}$ lines contains two space-separated integers $\boldsymbol{x}$ and $\boldsymbol{y}$ ($-10^5 \leq \boldsymbol{x}, \boldsymbol{y} \leq 10^5$). These are the points. Each point will be unique, and no three points will be collinear.

Output

Output a single integer, which is the maximum number of times any point $p \in P$ can be *promoted*, considering a full $360°$ rotation and any arbitrary starting point.

Sample Input	Sample Output
3 -1 0 1 0 0 2	2
6 0 0 5 0 0 5 5 5 1 2 4 2	3