Problem A. Average Convex Hull

Input file:	`average.in`
Output file:	`average.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

A convex hull of a set S on a plane is a minimal convex polygon that contains all points of S.

You are given n points on a plane. One of the points is selected uniformly at random and removed from the set.

Find the average number of vertices of the convex hull of the resulting set. For the purpose of this problem if the convex hull is a segment, it has 2 vertices. If the convex hull is a non-degenerate polygon, angles at all of its vertices are strictly less than π.

Input

The first line of the input file contains n — the number of points ($3 \leqslant n \leqslant 200\,000$). The following n lines contains two integers each — coordinates of the given points. Coordinates do not exceed 10^9 by absolute values. No two points coincide.

Output

Output the average number of vertices of the convex hull of the resulting set of points as an irreducible fraction p/q.

Examples

average.in	average.out
5 0 0 0 4 4 0 3 3 4 4	17/5

Problem B. Binary Suffix Array

Input file:	`binary.in`
Output file:	`binary.out`
Time limit:	1 second
Memory limit:	256 megabytes

A binary word is a word that consists of 0-s and 1-s.

Consider a binary word w of length n. Suffix array of w is an array $a[1..n]$ such that $w[a[i]..n]$ is the i-th suffix in lexicographical order among all suffices of w. For example, if w="`001011`", the lexicographical order of its suffices if "`001011`", "`01011`", "`011`", "`1`", "`1011`", "`11`", so its suffix array if $(1, 2, 4, 6, 3, 5)$.

You are given a suffix array a of some binary word w. Restore w.

Input

The first line of the input file contains n — the length of w ($1 \leqslant n \leqslant 300\,000$). The second line contains n distinct integers ranging from 1 to n — the suffix array of w.

Output

Output such word w that the array in the input file is its suffix array. If there are several possible words w, output any one. If there is no such word, output "`Error`" instead.

Examples

binary.in	binary.out
6 1 2 4 6 3 5	001011

Problem C. Collision Detection

Input file:	`collision.in`
Output file:	`collision.out`
Time limit:	1 second
Memory limit:	256 megabytes

Giggle company is developing the car that will ride automatically without a driver at general purpose roads. Currently they have built a prototype that can ride along the straight line. Now they want to test their collision detection mechanism, so they are planning to run the following test.

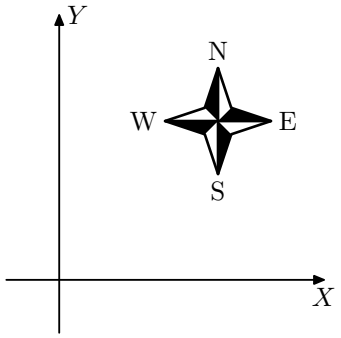
Consider a car as a point on a plane. The test will involve n cars. The cars will be put to predefined positions and oriented in one of the four directions: facing north, south, west or east. After that the cars will start moving in the directions they face with a speed of 1 meter per second. The collision detection system must notice dangerous approach of cars to each other and prevent collisions.

In order to get expected behavior of the cars Giggle engineers want to know the pair of the cars that have the highest danger of collision. As a measure of collision danger between a pair of cars they use the smallest distance between the cars after they start moving, the less is that distance the higher is the danger of collision. The distance between the cars is measured as the length of the segment connecting them.

Input

Input file contains several test cases.

The first line of each test case contains n — the number of cars in the experiment ($2 \leqslant n \leqslant 100\,000$). The following n lines contain two integers x_i and y_i followed by one character c_i and describe initial positions of the cars and their direction. Coordinates are given in meters and do not exceed 10^8 by their absolute values. Direction is given as one of the following characters: "`N`" — north, "`S`" — south, "`E`" — east, "`W`" — west. No two cars occupy the same position.



The last test case is followed by $n = 0$, it must not be processed. The sum of values of n for all test cases doesn't exceed 100 000.

Output

For each test case output three lines. The first line must contain d — the smallest distance in meters between a pair of cars after they start moving. The second line must contain two integers a and b — numbers of the cars that will get at distance d at some moment after they start moving. Cars are numbered starting from 1 in order they are given in the input. The third line must contain t — time in seconds from the beginning of the experiment until the cars a and b will be at distance d. If there are several possible answers, output any one.

Evaluation program will make all comparisons of floating point numbers with 10^{-6} absolute or relative tolerance.

Output empty line between test cases.

Examples

collision.in	collision.out
4 0 0 E 3 3 N 0 4 S 4 0 E 4 0 0 S 3 3 N 0 4 S 4 0 E 2 0 0 S 0 1 N 0	2.8284271247461903 1 3 2.0 3.0 2 3 0.5 1.0 2 1 0.0

Problem D. Dual Cure

Input file:	dual.in
Output file:	dual.out
Time limit:	2 seconds
Memory limit:	256 megabytes

Epidemic of rhino flu is the new problem in North-Eastern Antarctica. Rhino flu virus is very polymorphic, so curing it is very difficult. Fortunately the doctors have developed an approach that allows curing rhino flu if the source of the infection is known.

There are n sick with rhino flu persons in NEAH (North Eastern Antarctica Hospital). For each person in NEAH it is known whom she was infected by. Some persons got virus from rhinos, some persons were infected by other persons.

There are 2 doctors in NEAH that are going to cure the patients. Curing one patient including taking all necessary tests takes 1 hour. In order to cure some person X infected by a person Y, the doctor needs to know the results of Y's medical tests. Therefore Y must be cured before X. Moreover, if Y was cured by another doctor, the documents that contain Y's test results must be scanned by doctors' assistants and brought from one doctor to another, it takes another hour, so X can be cured only one hour after Y is finished being cured.

Now the doctors want to cure all n patients as soon as possible, so they minimize the time the last person is cured. Help them to arrange curing in such way that their goal is achieved.

Input

The first line of the input file contains n — the number of patients ($1 \leqslant n \leqslant 100\,000$). The following line contains n integer numbers $a_1, a_2, \ldots, a_n$. For each i from 1 to n the number a_i is a person that person i was infected by, or 0 if she was infected by a rhino ($a_i < i$). Persons are numbered from 1 to n.

Output

The first line of the output file must contain t — the minimal number of hours needed to cure all patients by the two doctors. The following t lines must contain two numbers each, the i-th of these lines must contain u_i and v_i — the numbers of persons cured by the first and the second doctor on the i-th hour, respectively. If the doctor doesn't cure anybody during some hour, the corresponding number must be equal to 0.

Examples

dual.in	dual.out
7 0 1 2 2 2 0 2	5 1 6 2 0 3 0 4 5 7 0

Problem E. Elections

Input file:	elections.in
Output file:	elections.out
Time limit:	3 seconds
Memory limit:	256 megabytes

Parliament elections in Flatland are based on electoral lists. Each party creates a list of its members that it would like to delegate to parliament. People of Flatland vote and the seats in parliament are distributed proportional to the number of votes given to each party. Top members of the list occupy the seats.

However, recent parliament elections in Flatland caused a lot of discussion. Famous politicians participated in electoral lists of various parties thus increasing their rankings, but after the elections they abandoned their seats passing them to people closer to the bottom of the list. It turned out that people voted for some famous persons that didn't go to the parliament. People argue that it would be more fair to give abandoned seats to other parties rather than people at the bottom of the list of the abandoning party. However things get more complicated when people from several parties abandon their seats.

To deal with the situation the new scheme of forming the parliament was developed. Let n parties take part in elections, let the parliament contain s seats. Before the elections each party forms a list of its delegates and the people vote for the parties. Let V people vote and v_i votes be given for the i-th party.

Each party creates a list of its delegates that abandon their seats. After that each party is assigned *preliminary seats*, the i-th party is assigned a_i preliminary seats. Denote sum of all a_i as A. Let b_i be number of delegates among the first a_i persons in the list of the i-th party that do not abandon their seats. The sum $\sum_{i=1}^{n} b_i$ must be equal to s, and corresponding b_i persons of the i-th party get seats in the parliament. To make distribution of seats in the parliament as fair as possible, numbers a_i must be chosen in such way that $\sum_{i=1}^{n} \left| \frac{a_i}{A} - \frac{v_i}{V} \right|$ is minimized.

Now members of the electoral committee need help of skilled programmers to find optimal values for a_i.

Input

The first line of the input file contains two integer numbers n and s ($2 \leqslant n \leqslant 10$, $1 \leqslant s \leqslant 50$).

Description of n parties follows. Description of the i-th party consists of two lines. The first line contains v_i — the number of votes given for the i-th party ($0 \leqslant v_i \leqslant 10000$, at least one v_i is positive). The second line contains m_i — the number of delegates from the i-th party that abandon their seats, followed by m_i integers — their positions in the electoral list of the i-th party (numbers ranging from 1 to 10000 listed in increasing order). The sum of m_i for all parties doesn't exceed 50.

Output

Output n numbers — the optimal values for a_i.

Examples

elections.in	elections.out
3 8 10 3 1 2 3 5 0 5 0	5 3 3

Problem F. Free of Squares

Input file:	`free.in`
Output file:	`free.out`
Time limit:	1 second
Memory limit:	256 megabytes

Consider words over binary alphabet $B = \{0, 1\}$. Let us fix a word w of length l. Let us enumerate positions in w from 1 to l, so, for example, if $w =$ "0110", $w[1] = 0$, $w[2] = 1$, $w[3] = 1$ and $w[4] = 0$.

Consider i from 1 to l, i is said to be *square free position* if for all k such that $i + 2k - 1 \leqslant l$ the words $w[i \ldots i+k-1]$ and $w[i+k \ldots i+2k-1]$ are different. For example, in a word "0110" position 1 is square free because "0"≠"1" and "01"≠"10", but position 2 is not because "1"="1".

Let us denote as $\sigma(k)$ the number of square free positions in w between 1 and k, inclusive. For example, in a word "0110" we have $\sigma(1) = 1$, $\sigma(2) = 1$, $\sigma(3) = 2$, $\sigma(4) = 3$.

A word w of length l is said to be *rather free of squares* if for all k from 1 to l the following inequality is satisfied: $2\sigma(k) \geqslant k$. So, a word "0110" is rather free of squares, but a word "0011" is not.

Given l, find all rather free of squares words of length l.

Input

The input file contains one integer l ($1 \leqslant l \leqslant 40$).

Output

The first line of the output file must contain k — the number of rather free of squares binary words of length l. The following k lines must contain these words. Words must be listed in lexicographical order.

Examples

free.in	free.out
4	4 0100 0110 1001 1011

Problem G. Gas Transportation

Input file:	`gas.in`
Output file:	`gas.out`
Time limit:	1 second
Memory limit:	256 megabytes

Flatland is planning to export gas to Edgeland. A system of gas pipes already exists containing n gas nodes numbered from 1 to n and m pipes, each pipe connects two nodes and can transport 1 million m^3 of gas per month in either direction. It is possible to transport gas from any node to any other node using the network. GasFlat, the main gas company of Flatland, has already connected its gas source node to the network as node 1 and Edgeland main gas storage is also connected to the network as node n.

The problem is that each of m pipes has its owner who asks for payment for transporting gas along the pipe. To reduce risks and expenses, GasFlat has organized Gas Transportation Association (GTA) and invites owners of the pipes to join it. If the owner of the i-th pipe joins association, he would get monthly payment equal to w_i flatland dollars (FD).

However, some owners think that the payment suggested to them is not fair. They know market price for gas transportation c, they decided that if some subset A of pipe owners would form its own coalition such that it is possible to transform $f(A)$ million m^3 of gas using only their pipes, they could charge GasFlat a total $f(A) \cdot c$ FD per month.

Let us call GasFlat's suggestion *fair* if no set A of pipe owners can form a coalition in such way that value $w(A) = \sum_{i \in A} w_i$ is strictly less than $f(A) \cdot c$.

Given a pipe network and GasFlat's suggestion $\{w_i\}$ check whether it is fair, and if it is not, find a set A of pipe owners that can form a better coalition.

Input

The first line of the input file contains three integers: n, m and c ($2 \leqslant n \leqslant 500$, $1 \leqslant m \leqslant 5000$, $1 \leqslant c \leqslant 10^9$). The following m lines describe pipes, each pipe is described by three integers: a_i and b_i — nodes it connects, and w_i — GasFlat's suggestion to the owner of this pipe ($0 \leqslant w_i \leqslant 10^9$).

Output

If GasFlat's suggestion is fair, output "`Fair`". In the other case output "`Unfair`" followed by the description of A. The description must start with k — number of elements in A followed by its elements. Pipe owners are numbered from 1 to m in order their pipes are described in the input file.

Examples

gas.in	gas.out
4 5 10 1 2 5 1 3 5 2 3 0 3 4 5 2 4 5	Fair
4 5 10 1 2 4 1 3 6 2 3 0 3 4 4 2 4 6	Unfair 3 1 3 4

Problem H. Handsome Division

Input file:	handsome.in
Output file:	handsome.out
Time limit:	1 second
Memory limit:	256 megabytes

Recall that a prime number is an integer that is divisible by exactly two integers: 1 and itself. Prime numbers series starts with $2, 3, 5, \ldots$.

Consider first n primes. Let us divide them to two parts A and B so that each prime belongs to exactly one of these two parts. Denote the product of all primes in A as a, and the product of all primes in B as b. Such division is called *handsome* if $a < b$ and $b - a$ is minimal possible.

Given n find the handsome division of the set of first n primes and print the corresponding value of a.

Input

Input file contains several test cases. Each test case consists of a number n on a line by itself ($1 \leqslant n \leqslant 30$). The last test case is followed by a line with $n = 0$, it should not be processed.

Output

For each test case print the value of a in handsome division of the set of first n primes.

Examples

handsome.in	handsome.out
1 2 3 4 5 0	Case #1: 1 Case #2: 2 Case #3: 5 Case #4: 14 Case #5: 42

Problem I. In Touch

Input file:	`intouch.in`
Output file:	`intouch.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Developer of a social network "In Touch" Paul Umnoff has decided to change the behavior of network friends feed.

Users of the network can leave messages on their wall, thus creating their microblog. Some users are friends, each user has *friends feed*, containing all messages from her friends.

Usually user can see all messages of her friends, but the newly introduced features changes this. Now user can only see messages that another user has posted while they were friends. When two users become friends, new messages one posts would be shown on other user's friends feed. If they unfriend each other, the past messages messages they posted while being friends stay on other's friends feed, but the new messages are not shown there.

To gather some statistics, Paul wants to find out how many messages can each of the users currently see in her friends feed.

Let users be identified by integers from 1 to n. You are given a list of events in order of their occurrence. Each event is either "user x posted a message in his microblog", "users x and y are now friends", or "users x and y are no longer friends". Initially no users are friends.

For each user you must print the number of messages in his friends feed after all events.

Input

The first line of the input file contains two integers: n — the number of users and m — the number of events ($1 \leqslant n \leqslant 200\,000$, $0 \leqslant m \leqslant 500\,000$). The following m lines describe events, each line contains one of the following:

- "`! x`" — user x posted a message in his microblog;
- "`+ x y`" — users x and y are now friends;
- "`- x y`" — users x and y are no longer friends.

Output

Output n numbers: for each user output the number of messages in his friends feed.

Examples

intouch.in	intouch.out
4 10 ! 1 + 1 2 + 1 3 ! 1 ! 2 - 1 2 ! 1 ! 2 ! 3 ! 1	2 1 3 0

Problem J. Jumbo World

Input file:	`jumbo.in`
Output file:	`jumbo.out`
Time limit:	2 seconds
Memory limit:	256 megabytes

Inspired by a game "Small World" by "Days of Wonder" Peter decided to create his own alternative. He called it "Jumbo World".

The game proceeds on a board consisting of several areas. Some areas are adjacent to each other. Each area has at most 5 adjacent areas. Some areas are adjacent to the edge of the board, these areas are called *border areas*. Each area has some terrain type that can give various bonuses to players.

Several players take part in the game, each controlling one *race*. Different players control different races. Each race has several *units* that can occupy various areas, each area can be occupied by several units of the same race, but units of two different races cannot occupy the same area.

Players move in turns, a turn consists of conquering some areas. Initially a player can conquer any border area. After a player has conquered border area she can continue conquering areas adjacent to other areas she has conquered.

A player has several units of his race. Conquering an area requires $\max(1, 2 + c + b)$ units where c is the number of units of other race occupying the area and b is a bonus value depending on terrain type, conquering race and conqured race. Units conquering an area stay there occupying it until the end of the turn.

If the player doesn't have enough units to conquer any area adjacent to her areas, her turn is over and she gets points for her areas. Each area by default gives 1 point, this value can be altered by bonuses, also bonuses can alter points depending on conquests.

Races have different abilities that alter conquest bonus value b and points received in the end of the turn. The following abilities are possible.

Ability	**Description**
+k `defence`	If this race's area is being conqured, k more units are needed (b += k).
+k `defence on` T	Here T is terrain type. If this race's area with terrain type T is being conqured, k more units are needed (b += k).
+k `defence near` T	Here T is terrain type. If this race's area which has adjacent area with terrain type T is being conqured, k more units are needed (b += k).
+k `defence versus` R	Here R is a race. If this race's area is being conqured by a race R, k more units are needed (b += k).
+k `attack`	If this race conquers some area, k less units are needed (b −= k).
+k `attack on` T	Here T is terrain type. If this race conquers area with terrain type T, k less units are needed (b −= k).
+k `attack near` T	Here T is terrain type. If this race conquers area which has adjacent area with terrain type T, k less units are needed (b −= k).
+k `attack versus` R	Here R is a race. If this race conquers area which is occupied by a race R, k less units are needed (b −= k).
+k `points`	This race gets k additional points in the end of the turn for each area it occupies.
+k `points on` T	Here T is terrain type. This race gets k additional points in the end of the turn for each area of terrain type T it occupies.
+k `points for conquest`	This race gets k additional points in the end of the turn for each area conquered that contained at least one unit of other race ($c > 0$).

For all abilities $1 \leqslant k \leqslant 9$.

You are planning to make a turn in "Jumbo World" and have n units of a given race. None of your units occupies any areas, but some areas may be already occupied by other players' units. Find out what is the maximal number of points you can get in the end of this turn.

Input

The first line of the input file contains four integers: t, r, a and n — number of terrain types, races, areas and units of your race, respectively ($1 \leqslant t \leqslant 10$, $2 \leqslant r \leqslant 10$, $1 \leqslant a \leqslant 30$, $1 \leqslant n \leqslant 10$).

The following t lines contain one word composed of English lowercase letters each and specify terrain types.

Descriptions of r races follow. Each description starts with a line containing the name of the race followed by a line containing v_i — number of its abilities, followed by v_i lines describing abilities as specified above. All race names are single words composed of lowercase letters of English alphabet. Any type of ability for any terrain/race is listed at most once.

Description of areas follows. Areas are numbered from 1 to a, each area is described by four lines. The first line contains its terrain type. The second line contains a word "`border`" if it's border, or "`interior`" if it's not border. The third line contains the name of the race that initially occupies this area followed by number of units that occupy it, or a word "`empty`" if the area is empty. No race is named "`empty`". The fourth line starts with b_i — the number of areas adjacent to it, followed by their numbers. No area is adjacent to itself. If area x is adjacent to area y, then area y is adjacent to area x. It is possible to get from any area to any other area by moving between adjacent areas. There is at least one border area. No area is occupied by more than 10 units.

The following line contains the name of the race that you own.

Output

Output one number: the maximal number of points you can get.

Examples

jumbo.in	jumbo.out
3 3 6 8 grass hills mountains dwarves 1 +2 points on mountains elves 3 +1 attack on grass +1 defence on grass +1 points for conquest human 2 +1 points on grass +1 attack versus dwarves mountains border dwarves 1 4 2 3 4 5 grass border elves 2 4 1 3 5 6 hills border elves 1 4 1 2 4 6 hills interior empty 4 1 3 5 6 hills interior empty 4 1 2 4 6 grass interior empty 4 2 3 4 5 human	5

You can get 5 points by conquering area 1 with $2 + 1 - 1 = 2$ units and then conquering areas 4, 5 and 6 with 2 units each.