Problem A
Cost of Living

A newspaper columnist recently wrote a column comparing Disneyland's price increases to other things in the economy; e.g., If gasoline were to have increased at the same rate as Disneyland's admission since 1990, it would cost $\$6.66$ per gallon.

Consider the prices of a number of commodities over a number of consecutive years. There is an inflation rate ($\mathrm{inflation}(x)>0$) for year x to the next year ($x+1$). Also, each commodity A has a modifier ($\mathrm{modifier}(A)>0$) that is fixed for that commodity. So, for commodity A in year $x+1$:

$$\mathrm{price}(A,x+1)=\mathrm{price}(A,x)\cdot\mathrm{inflation}(x)\cdot\mathrm{modifier}(A)$$

Unfortunately, the modifiers are unknown, and some of the prices and inflation rates are unknown.

Given some inflation rates, the prices for a number of commodities over a number of consecutive years, and some queries for prices for certain commodities in certain years, answer the queries.

Input

Each test case will begin with a line with three space-separated integers $y(1\le y\le 10)$, c ($1\le c\le 100$), and q ($1\le q\le y\cdot c$), where y is the number of consecutive years, c is the number of commodities, and q is the number of queries to answer.

Each of the next $y-1$ lines will contain a single real number r ($1.0\le r\le 1.5$, or $r=-1.0$), which are the inflation rates. A value of -1.0 indicates that the inflation rate is unknown. The first inflation rate is the change from year 1 to year 2, the second from year 2 to year 3, and so on. Known inflation rates will conform to the limits specified; unknown but uniquely determinable inflation rates may not, and are only guaranteed to be strictly greater than zero.

Each of the next $y$lines will describe one year's prices. They will contain c space-separated real numbers p ($1.0<p<1000000.0$, or $p=-1.0$), which indicate the price of that commodity in that year. A value of -1.0 indicates that the price is unknown.

Each of the next q lines will contain two space-separated integers a ($1 \leq a \leq c$) and b ($1 \leq b \leq y$), which represent a query for the price of commodity a in year b. All queries will be distinct.

All prices that can be uniquely determined will be strictly greater than zero and strictly less than 1000000.0. Values for prices and inflation rates in the input may not be exact, but will be accurate to 10 decimal places. All real values contain no more than 10 digits after the decimal point.

Output

Produce q lines of output. Each line should contain a single real number, which is the price of the given commodity in the given year, or -1.0 if it cannot be determined. Answer the queries in the order that they appear in the input. Your answers will be accepted if they are within an absolute or relative error of 10^{-4}.

Sample Input 1

```
4 2 2
1.3333333333
1.2500000000
-1
3.00 -1
4.00 8.00
5.00 10.00
-1 11.00
2 1
1 4
```

Sample Output 1

```
6.0000000000
5.5000000000
```

Round Trips

Micah lives a peaceful life with his family in one of Canada's most beautiful provinces. The vast wealth he has accumulated allows him to indulge in an extravagant hobby: collecting vintage automobiles. On Sunday afternoons, Micah and his family enjoy taking long, leisurely drives in these classic cars. Because of the unusual road system in the province, they have made something of a game out of planning their weekly outings.

Micah's province contains n towns that are connected by a network of roads. Every road joins some town x to a different town y, and all roads are *one-way (!)* There is never more than one road from any town x to any other town y (although there may be a road going in the reverse direction), and other than the fact that roads may meet at their endpoints (towns), no two roads intersect (this is facilitated by an elaborate system of overpasses and underpasses).

Each Sunday after lunch, Micah and his family plan and then embark on what they call a *round trip*. This involves first going to one of the n towns (via helicopter, of course; driving there would detract from the graph theoretic purity of the entire excursion), getting into one of Micah's fine cars (also transported there by helicopter), and then driving from town to town along the various one-way roads in such a way that they always end up back at the town where they started (whereupon helicopters transport everyone/everything back home). There is one family cardinal rule: during one of these round trips, they can never drive along the same road twice. Overall, a round trip can be represented as a sequence of towns

$$x_0 \quad x_1 \quad x_2 \quad \dots \quad x_{T-1} \quad x_T$$

where (a) $T \geq 2$, (b) $x_0 = x_T$ is the starting town, (c) there is a (one-way) road from x_i to x_{i+1} for $0 \leq i < T$, and (d) no road is repeated. Note that $x_0, x_1, \dots, x_{T-1}$ are not necessarily all distinct.

In their endless quest for adventure, Micah and his family have decided that they never want to take the same round trip twice, so Micah has designed a simple algorithm to count exactly how many round trips are possible. It saddens them, though, when this algorithm reveals that this number is in fact *finite*, which means that one day they will have used up all the possible round trips. However, there is hope! From time to time, the province constructs a new road, which is very exciting because a new road may mean new round trips. Unfortunately, Micah's evil twin, Hacim (separated at birth), has recently been appointed Director of the Inter-County Paving Commission (ICPC), and he is determined to use his new power to minimize all this vexing family-oriented fun. Hacim cannot prevent the construction of new roads altogether, of course (that would eventually get him fired), but he *can* influence which new roads get constructed. For Hacim, "fun" means approving the construction of a new road that does not create any new round trips for Micah and his family. Hacim wonders how long he can continue doing this, i.e., how many new roads can be constructed without creating any new round trips. Since he is not as good as Micah at designing counting algorithms, he needs your help to figure this out.

Note that a new road, like an existing road, can meet other roads at its endpoints, but otherwise it cannot intersect any other road. Also, the construction of new roads is, of course, cumulative (roads never get removed).

Input

The first line of input contains two space-separated integers, n and m ($1 \le n \le 100000$, $0 \le m \le 200000$), where n is the number of towns (indexed $0, 1, \ldots, n-1$) and m is the number of one-way roads. This is followed by m lines, each containing two space-separated integers, x and y ($0 \le x, y \le (n-1)$, $x \ne y$), indicating that there is a one-way road from town x to town y. No road will be specified more than once.

Output

Output a single integer, the maximum number of one-way roads that can be constructed without creating any new round trips.

Sample Input 1

```
2 1
0 1
```

Sample Output 1

```
0
```

Sample Input 2

```
5 7
3 2
4 0
3 1
0 3
4 2
1 0
3 4
```

Sample Output 2

```
2
```

Pipes

Kažimír got an important job at the plumbing department. He needs to connect certain pairs of locations by pipes. However, as the department's budget is very low, they have only two types of pipe parts: straight and "elbow" (L) shaped ones, all of the same size. Moreover, each pipe part is embedded into a square tile and the department provides a grid board onto which one can easily attach these tiles. The tiles can be rotated. The advantage of this design is that when two adjacent tiles' pipes align properly, there is no leaking. However, not having the option to go into the third dimension somewhat limits Kažimír connection options. Help him figure out whether he can connect the pairs of locations!

The grid board is of dimension $m \times n$. There are ℓ pairs of locations that need to be connected. Every location is along a side of the board, in the middle of a side of a grid square. Once Kažimír places a pipe tile on a grid square, he is not allowed to place another tile on the same square.

Input

The first line contains k, the number of input instances.

Each input instance is described on several lines. The first line contains three positive integers m, n, and ℓ. The second line contains 2ℓ numbers $s_1, t_1, s_2, t_2, \dots, s_\ell, t_\ell$ that describe the pairs of locations: Kažimír needs to connect location s_i to t_i for every $i \in \{1, \dots, \ell\}$. The locations are described by numbers between 1 and $2(m+n)$, with 1 being the left side of the square in the top-left corner, then 2 is the left side of the square immediately below, 3 below that and so on, going in counterclockwise order around the boundary of the board; $2(m+n)$ is the top side of the top-left square. You may assume that $1 \le m \cdot n \le 10000$ and that there are no duplicate locations – all the s_i and $t_j$numbers are distinct.

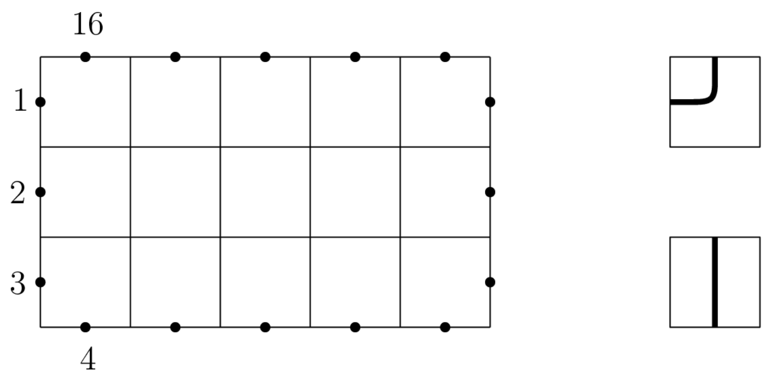


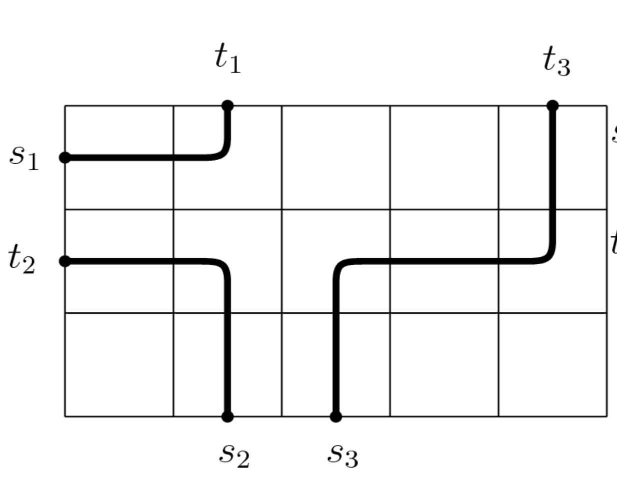
Figure 1: Numbering of locations for an $m \times n = 3 \times 5$ board. The types of tiles are shown on the right.

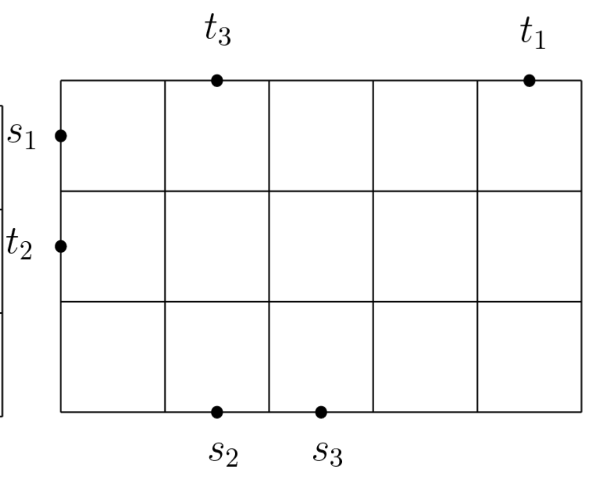
Output

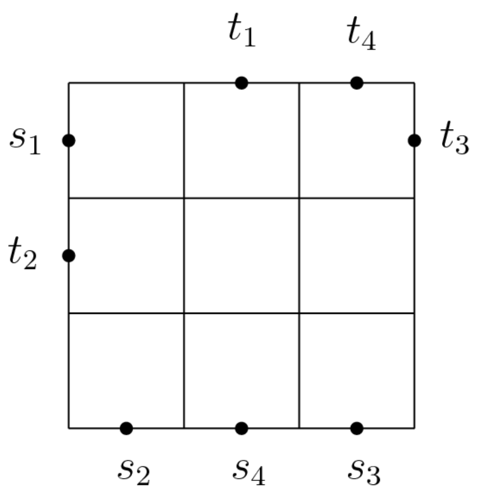
The output contains k lines. The i-th line corresponds to the i-th input. It contains `YES` if it is possible to connect all the pairs of locations and `NO` otherwise.

Note

The three sample inputs are depicted below—sample input 1 on the left, with possible pipe connections between the locations; sample inputs 2 and 3 are in the middle and on the right, respectively.







Sample Input 1

```
3
3 5 3
1 15 5 2 6 12
3 5 3
1 12 5 2 6 15
3 3 4
1 11 4 2 6 9 5 10
```

Sample Output 1

```
YES
NO
NO
```

Distance

The City of Manhattan is organized as a grid of streets and avenues, with streets running in the North-South direction and avenues running in the East-West direction. Streets are numbered from East to West starting from 1, and avenues are numbered from North to South starting from 1. Each intersection is labelled by the street and avenue number (s,a). The distance between two intersections $(s1,a1)$ and $(s2,a2)$ is $|s1-s2|+|a1-a2|$.

Your company operates several food trucks at different intersections in Manhattan and you want to have them spread out so they do not compete with each other. To estimate how spread out they are, you have decided to compute the total distance between every distinct pair of your food trucks. A small total distance would mean that on average, a pair of food truck is too close together.

What is the total distance between every distinct pair of food trucks?

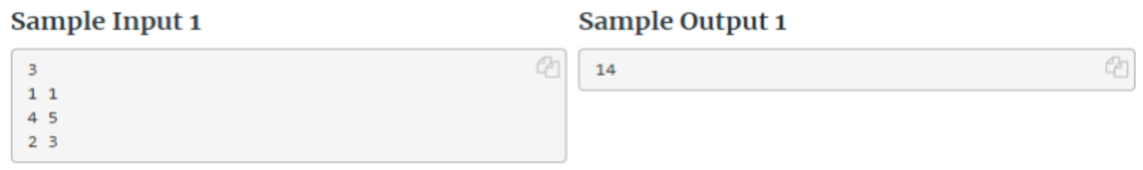
Input

The first line of input contains an integer $N(2 \le N \le 200000)$, which is the number of food trucks.

The next $n$lines describe the food trucks' locations. Each of these lines contains two integers s $(1 \le s \le 1000000)$, which is the street number of this food truck, and a $(1 \le a \le 1000000)$, which is the avenue number of this food truck.

Output

Display the total distance between every distinct pair of food trucks.



Sample Input 1

```
3
1 1
4 5
2 3
```

Sample Output 1

```
14
```

N-sum

Now one’s goose is cooked and the raw carrots are smoked!

Per-Magnus’ boss stormed into his office and complained about the addition program you had written for him previously. It can only add *two* numbers, which is of course completely unusable! How could you even come up with such a silly idea?

Fix your program posthaste, so that it instead sums up N integers for a given N.

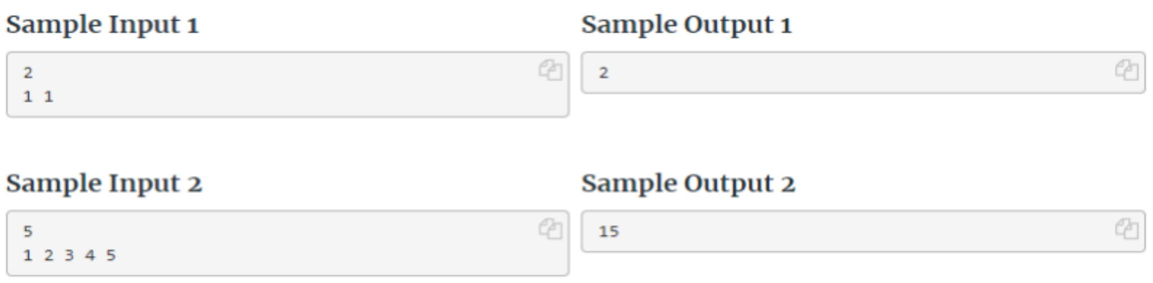
Input

The first line of the input contains an integer $N (2 \leq N \leq 10)$, the number of integers your program should add.

The next line contains the Nintegers to add, each between 0 and 1000.

Output

Output a single integer – the sum of the N integers from the input.



Sample Input 1

```
2
1 1
```

Sample Output 1

```
2
```

Sample Input 2

```
5
1 2 3 4 5
```

Sample Output 2

```
15
```

Julmust

An array of julmust bottles with a glass of julmust in front.
Maj loves the traditional Swedish Christmas soft drink called *julmust*. She likes it so much that she has convinced her local store to start selling it already on the 1st of October!

Maj is a creature of habit. She consumes exactly the same number of bottles of julmust every day from the 1st of October until Christmas Eve (the 24th of December). The number of bottles per day is a positive integer r, where $0<r\leq R$.

You need to determine the total number of julmust bottles she has consumed so far. You need to be done at the end of Christmas Eve, 85 days after she started drinking.

At the end of every day, you ask Maj how much julmust she had so far. You can only ask one question per day.

Interaction

The interaction starts by giving your program a single line of input with an integer R, the maximum number of bottles Maj could possibly consume per day. Interaction proceeds in round, with one round per day. In each round, your program asks if Maj has consumed X bottles of julmust since the 1st of October up to and including the current day. The format of a question is "X" where X is a non-negative integer. Thus, your first question "$X1$" means "Have you consumed $X1$ bottles on the 1st of October," your second question "$X2$" means "Have you consumed $X2$ bottles in total on the 1st and 2nd of October," and so on.

Maj's response to each question can be read from standard input. Her response is a single line with one of the following three words:

- "less" if Maj has consumed strictly fewer than X
- bottles so far.

- “exact” if Maj has consumed exactly X
- bottles so far.
- “more” if Maj has consumed strictly more than X
    - bottles so far.

When your program receives “exact” it has successfully completed the task and should exit, not asking any more questions.

Your program is judged *wrong* if you have not received an “exact” response after 85 rounds.

Flushing

In interactive problems your program will be judged by a judge program. If your program does not flush the output stream when printing output the judge program might not receive your output in time and your program might be judged as Time Limit Exceeded. To avoid this, make sure to always flush your output after each time you print.

In Python3:

```
print(x, flush=True)
```

In Java:

```
System.out.println(x);
System.out.flush();
```

In C++:

```
std::cout << x << "\n" << std::flush;
```

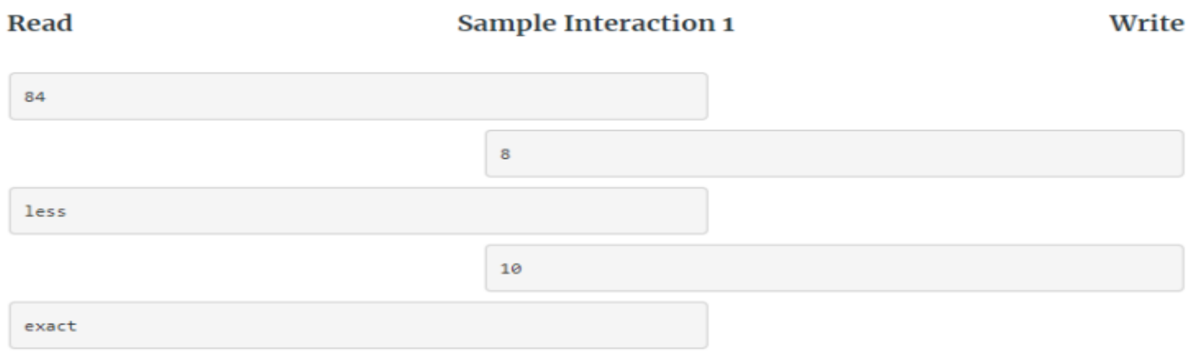
Scoring

Group	Points	Limits
1	25	$R=84$
2	20	$R=1500$
3	55	$R=1000000$

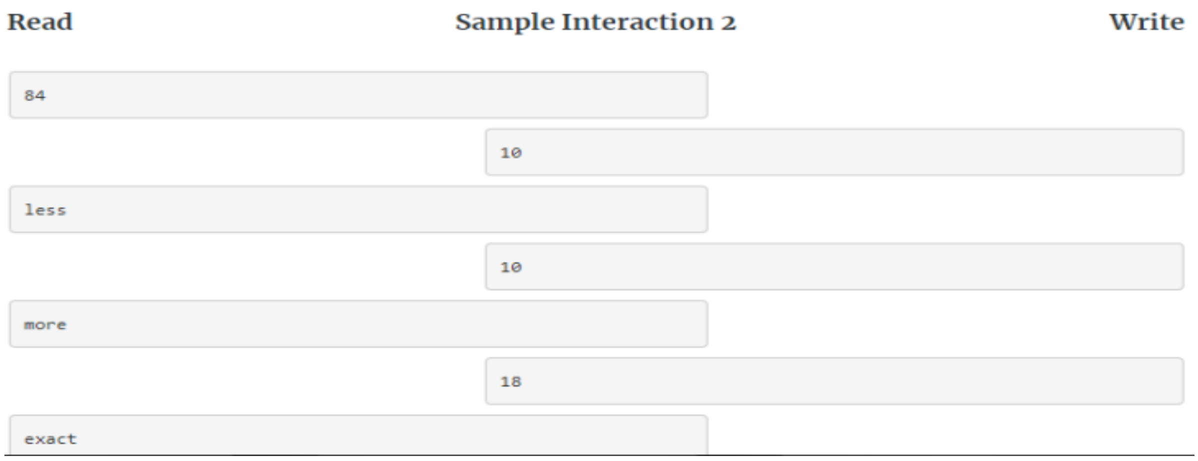
Sample Description

In Sample Interaction 1, Maj consumes 5 bottles of julmust each day. On the first day you ask if she has consumed 8 bottles, but she has only consumed 5 so far. On the second day you ask if she has consumed 10 bottles, which is true and the interaction ends.

In Sample Interaction 2, Maj consumes 6 bottles of julmust each day. On the first day you ask if she has consumed 10 bottles, but she has only consumed 6 so far. On the second day you ask if she has consumed 10 bottles, but now she has consumed 12 bottles. On the third day you ask if she has consumed 18 bottles, which is true and the interaction ends.



Read	Sample Interaction 1	Write
84		
		8
less		
		10
exact		



Read	Sample Interaction 2	Write
84		
		10
less		
		10
more		
		18
exact		

Bluetooth

Harald Bluetooth depicted in the 17th century.

King Harold of Denmark (c. 998–1098) wants to eat an apple to celebrate the founding of the town Lund. Harold’s teeth are in terrible condition; some are missing, some are rotten and discoloured, earning him the nickname “Bluetooth.”

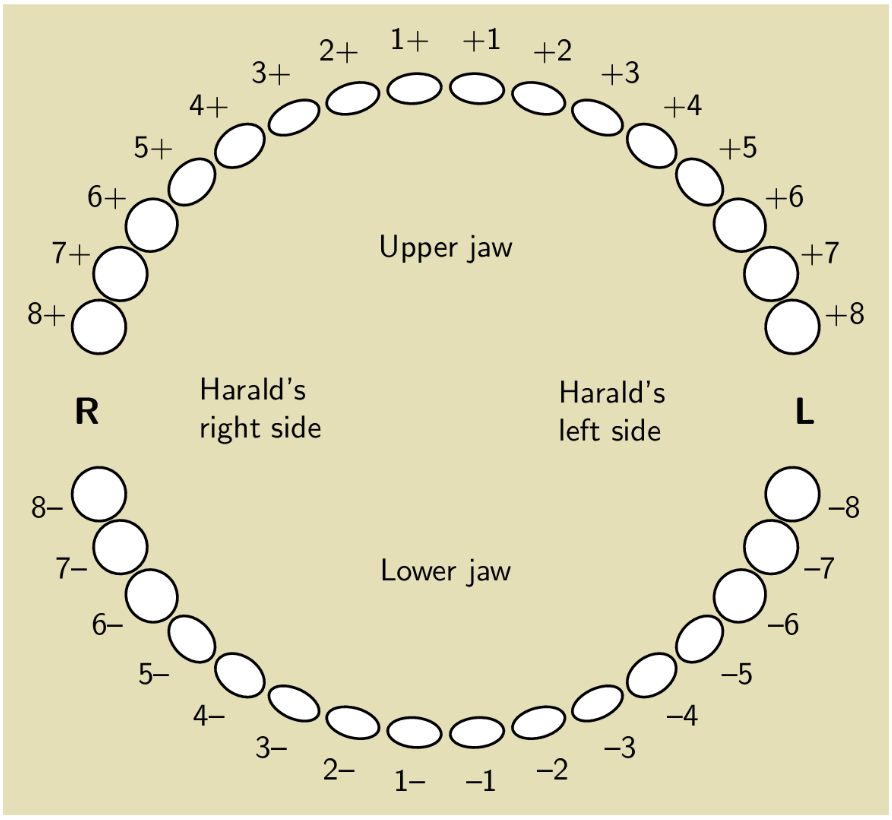
Harold can chew the apple on the left or the right side of his mouth. Chewing with rotten teeth causes terrible pain, so he cannot chew on a side containing a blue tooth. He needs at least one tooth in both the upper and the lower jaw on the same side to be able to chew on that side.

As Harold's chief advisor and reluctant dentist, you have a complete overview of his dental problems. Can you help him decide on which side to chew, or if apples are too hard to handle?

Input

The first line of input is an integer n, the number of Harold's dental problems, where $1 \leq n \leq 32$. Each of the following n

lines describes a dental problem. A dental problem consists of a space separated tooth and a condition, »m« (for »missing«) or »b« (for »blue«). No tooth will appear more than once in the problem description. Teeth are described using the nomenclature created by your dentist colleague Victor Haderup:



Harald has at least one blue tooth. A tooth cannot be both missing and blue.

Output

The integer 0

if Harold can chew the apple with the left side. The integer 1 if Harold can chew the apple with the right side. The integer 2 if you recommend soup.

Scoring

Group	Points	Limits
1	19	$n = 1$
2	81	$1 \leq n \leq 32$

Sample Input 1

```
1
-5 b
```

Sample Output 1

```
1
```

Sample Input 2

```
9
8- m
7- m
6- m
5- m
4- m
3- m
2- m
1- b
+3 m
```

Sample Output 2

```
0
```

Sample Input 3

```
9
8- m
7- m
6- m
5- m
4- m
3- m
2- m
1- m
+3 b
```

Sample Output 3

```
2
```

Sample Input 4

```
15
6+ b
+2 m
+3 m
+4 m
+5 m
+6 m
+7 m
+8 m
-1 m
-2 m
-3 m
-4 m
-5 m
-6 m
-7 m
```

Sample Output 4

```
0
```

Lamps

The use of incandescent light bulbs is soon being banned within the European Union, but it can already be profitable to use low energy lamps. Your task is to write a program that, given for how long a light is on each day and the price of electricity, computes after how many days the total cost of ownership (the purchasing price of the lamp plus the cost of electricity) is lower for the low energy lamp compared to the incandescent light bulb for the first time.

We assume the following data:

	Incandescent bulb	Low energy lamp
Power (watt)	60	11
Life (hours)	1000	8000
Price (Swedish Kronor)	5	60

For simplicity, we assume that each lamp has the exact given life. This means that if you only need to have a light on for 1000

hours you only need a single incandescent bulb, while you need two if you use it for 1001

hours, since the first breaks after 1000

hours.

The cost of electricity K

for a lamp that is on for H

hours can be computed with the formula

$$K=E\cdot H\cdot P100000$$

where E

is the power of the lamp in Watt and P

is the price of electricity (in hundredths of a Krona per kilo-Watt hour).

In every test case, it is guaranteed that low energy lamp becomes cheaper within 8000

hours, i.e. you will never have to buy an additional low energy lamp.

Input

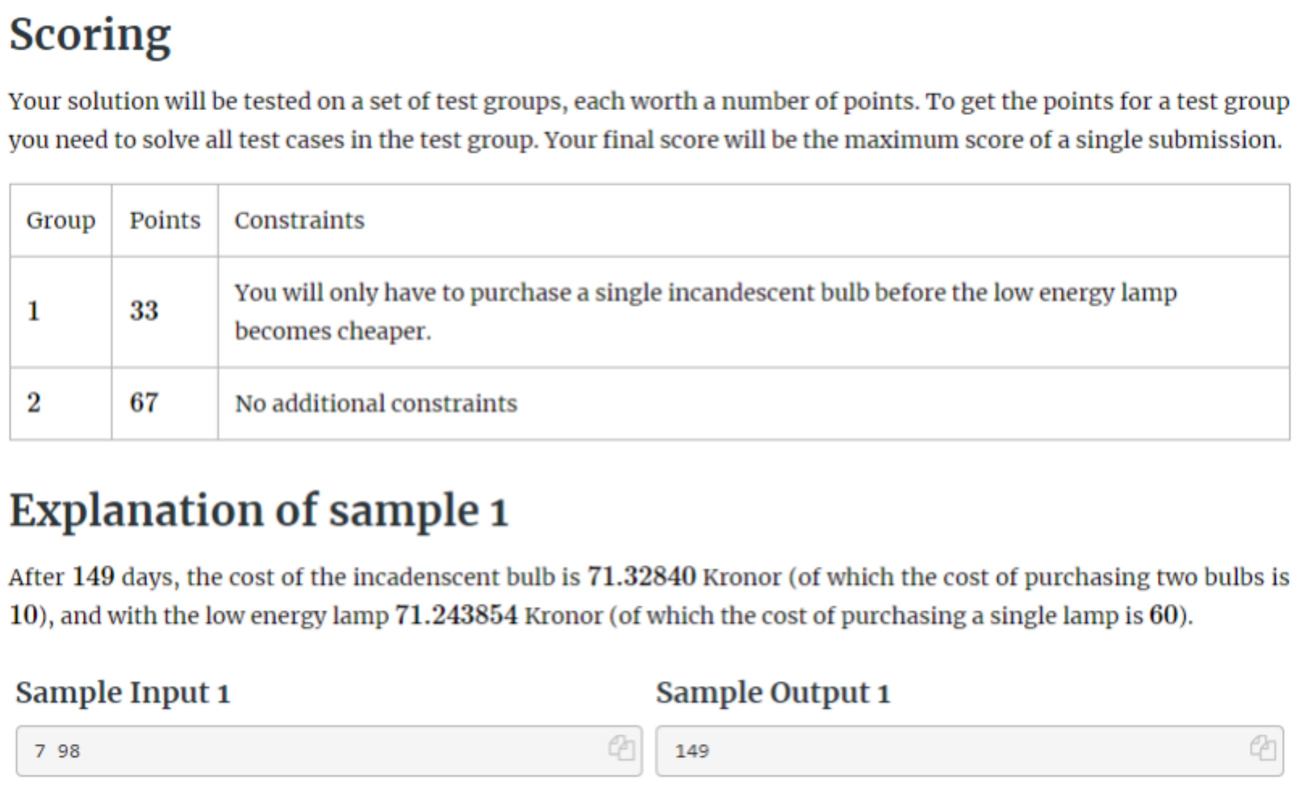
The first line and only line of input contains two integers h

$(1 \leq h \leq 24)$ and $P(1 \leq P \leq 200$

) – the number of hours per day the lamp is on, and the price of eletricity.

Ouput

Output a single integer – the number of days after which the low energy lamp is cheaper than the incandescent bulb.



Scoring

Your solution will be tested on a set of test groups, each worth a number of points. To get the points for a test group you need to solve all test cases in the test group. Your final score will be the maximum score of a single submission.

Group	Points	Constraints
1	33	You will only have to purchase a single incandescent bulb before the low energy lamp becomes cheaper.
2	67	No additional constraints

Explanation of sample 1

After 149 days, the cost of the incadenscent bulb is 71.32840 Kronor (of which the cost of purchasing two bulbs is 10), and with the low energy lamp 71.243854 Kronor (of which the cost of purchasing a single lamp is 60).

Sample Input 1

```
7 98
```

Sample Output 1

```
149
```

Töflur

You really like playing games with your friend. You are currently playing a game where each player is given n tiles with numbers on them. Each player then places the tiles down such that they form a sequence. Let a_j denote the number on the j-th tile in the sequence, for $j=1,2,\ldots,n$. The *score* of the sequence is then computed by adding up the squares of the differences of adjecent tiles, that is

$$\sum_{j=1}^{n-1}(a_j-a_{j+1})^2.$$

The player with the lowest score wins.

Input

The first line of the input is an integer $1 \le n \le 10^6$, the number of tiles you have. The next line consists of n integers each being at least one, but not larger than 10^6, the numbers on the tiles.

Output

The only line of the output should contain one integer, the lowest score you can achieve by arranging your tiles optimally.

Sample Input 1

```
9
1 2 3 1 1 2 2 3 3
```

Sample Output 1

```
2
```

Sample Input 2

```
7
4 8 7 25 95 97 6
```

Sample Output 2

```
5199
```

Stigavörður

You really like playing games with your friend. This time, however, you are stuck being the score keeper for two of your friends. They are playing a game where n numbered tiles lay on the board in a line. The players can do one of two moves each turn. They can change the number on a single tile. They can also choose two numbers j and k, such that $1 \le j \le k \le n$, and they score

$$a_j \oplus a_{j+1} \oplus \cdots \oplus a_{k-1} \oplus a_k$$

points, where a_j is the j-th tile and $a \oplus b = \gcd(a,b)$. You aren't quite sure what the goal of the game is, but that doesn't matter. All you need to do is keep track of the scores.

Input

The first line of the input contains two integers, $1 \leq n \leq 10^5$ and $1 \leq q \leq 10^5$. The next line contains n integers all larger than zero, but none larger than 10^9. The next q lines all contain three integers, x, y, and z. Each line describes a single turn in the game. The integer x is either 1 or 2. If $x=1$, then $1 \leq y \leq n$ and $1 \leq z \leq 10^9$ and this means a player changed the number on the y-th tile to z. If $x=2$, then $1 \leq y \leq z \leq n$ and this means a player scored, as described above, with $j=y$ and $k=z$.

Output

For each turn in the game, in order, where a player scores the output should conatain a line with the score achieved that turn.

Sample Input 1

```
4 5
1000000000 2 4 100
2 1 1
2 2 2
2 3 3
2 4 4
2 1 4
```

Sample Output 1

```
1000000000
2
4
100
2
```

Stikl

You really like playing games with your friend. A current favorite of yours is the game *Stikl*. During the setup of the game n

numbered tiles are shuffled and placed in a line. The rules of the game are simple. Both players start on a random tile. They then take turns rolling dice and make the corresponding number of moves. One move consists of moving from your inital tile to the first tile to your right that does not have a lower number than the initial tile. If no such tile exists the player wins. The only difficult part of this game is figuring out where your piece ends after a certain number of moves.

Input

The first line of the input consist of two integers $1 \leq n \leq 10^5$ and $1 \leq q \leq 10^5$. The second line of the input consists of n integers, $1 \leq a_i \leq 10^9$. Theses integers correspond to the numbers on the tiles. Then follow q queries, one on each line, containing two integers, $1 \leq s_j \leq n$ and $1 \leq d_j \leq 10^5$.

Output

For each query print a single line containing the index of the tile you end on if you start at the s_j-th tile and make d_j moves. If performing the prescribed number of moves would cause the player to win, then print „`leik lokid`”.

Sample Input 1

```
16 11
1 9 5 13 3 11 7 15 2 10 6 14 4 12 8 16
1 1
1 2
1 3
1 4
1 5
3 1
3 4
5 1
9 3
11 3
16 1
```

Sample Output 1

```
2
4
8
16
leik lokid
4
leik lokid
6
16
leik lokid
leik lokid
```

Sample Input 2

```
5 5
1 1 1 1 1
1 1
1 3
1 5
2 4
2 5
```

Sample Output 2

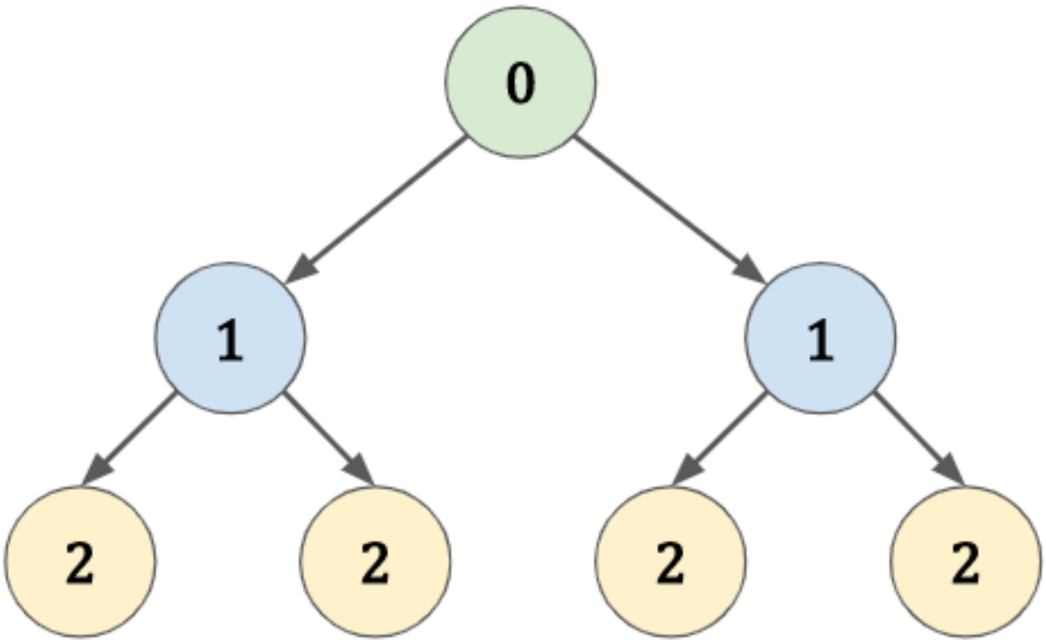
```
2
4
leik lokid
leik lokid
leik lokid
```

Widget Tree

Alice got bored when staying at home for several months due to the pandemic and decided to learn about front-end development. She learnt that project components are often modeled as a widget, and decided to write a program to calculate the cost of building and updating them.

In a particular project that she is interested in, there are N types of widgets numbered from 0 to $N-1$. Each widget may depend on at most 10 other distinct types of widget. All dependencies of a widget must be built before the widget itself could be built.

Firstly, Alice is interested in calculating the cost of building widget 0, which is equivalent to the size of its dependency tree.

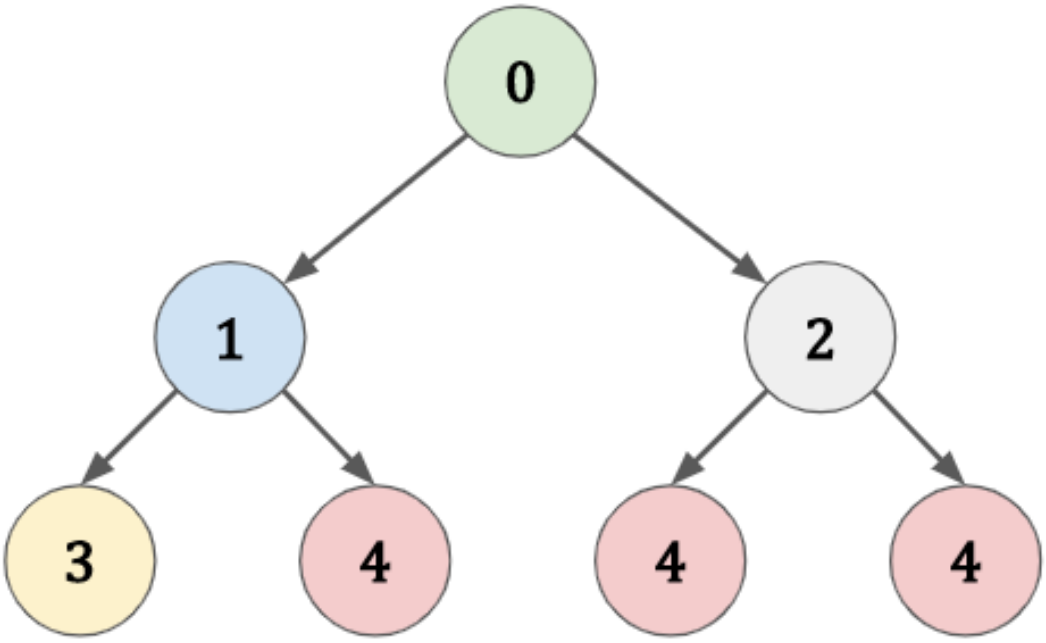


In the example above, $N=3$, widget type 0 depends on two widgets of type 1, and widget type 1 depends on two widgets of type 2. The size of the dependency tree is 7, which is also the required cost of building widget 0.

Secondly, Alice is interested in calculating the cost of maintaining widget type 0 when updates occur. There are Q update queries in total. In each query, there are X_i $(1 \leq i \leq Q)$ types of widgets to be updated. Note that there might be multiple widgets associated with a particular type and Alice only cares for widgets that belong to the dependency tree of widget type 0.

For each query, you need to output the number of widgets that need to be rebuilt. A rebuild must be performed for a widget if it satisfies at least one of the following conditions:

1. The widget type is updated in the query.
2. At least one of its descendants or ancestors in the dependency tree has a widget type that needs to be updated in the query.



In the example above, the cost of initial build for this dependency tree is 7

. Next, suppose that there is a query where widgets with type 2 and 3 need to be updated. Based on the first condition, we know that there are two widgets that need to be rebuilt, which are those of type 2 and 3

. Afterward, based on the second condition, we discover four more widgets that need to be rebuilt:

- Two widgets of type 4, which are the descendants of widget with type 2.
- One widget of type 1, which is the ancestor of widget with type 3.

- One widget of type 0, which is the ancestor of widget with type 2 and 3.

Hence, in total, there are 6widgets that should be rebuilt. As this value may be large when we have a bigger dependency tree, you only need to output the value in modulo of M.

If it is impossible to build widget 0, you only need to output a single line of string “Invalid”.

Input

The first line of the input consists of two integers $N(1 \leq N \leq 1000)$ and M $(1 \leq M \leq 10^9)$. Each of the next N lines consists of a single integer C_i $(0 \leq C_i \leq 200)$, the number of dependencies for widget of type i. It is followed by C_i integers, where $D_{i,j}$ $(0 \leq D_{i,j} \leq N-1)$ is the j-th dependency for widget of type i. There are at most 10distinct types of widget among the dependencies of a single widget.

The next line of the input will contain two integers $Q(0 \leq Q \leq 1000)$ and T $(T=0$ or 1, explained in the output section). Each of the next Q lines consists of a single integer X_i $(0 \leq X_i \leq 200)$, the number of widget types to be updated for query i. It is followed by X_i integers, where $Y_{i,j}$ $(0 \leq Y_{i,j} \leq N-1)$ is the j-th type that needs to be updated for the i-th query.

Output

For $T=0$:

If it is not possible to build widget 0, output a single line with string “Invalid”. Otherwise:

The first line of the output must consist of a single integer, the cost of the initial build in modulo M. The next Q lines of the output must also consist of a single integer, the answer for each query in modulo M.

For $T=1$, Q is always 0:

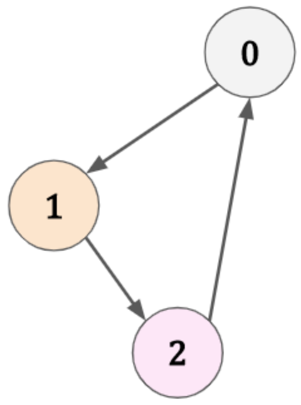
If it is not possible to build widget 0, output a single line with string “Invalid”. Otherwise, output a single line with string “Valid”.

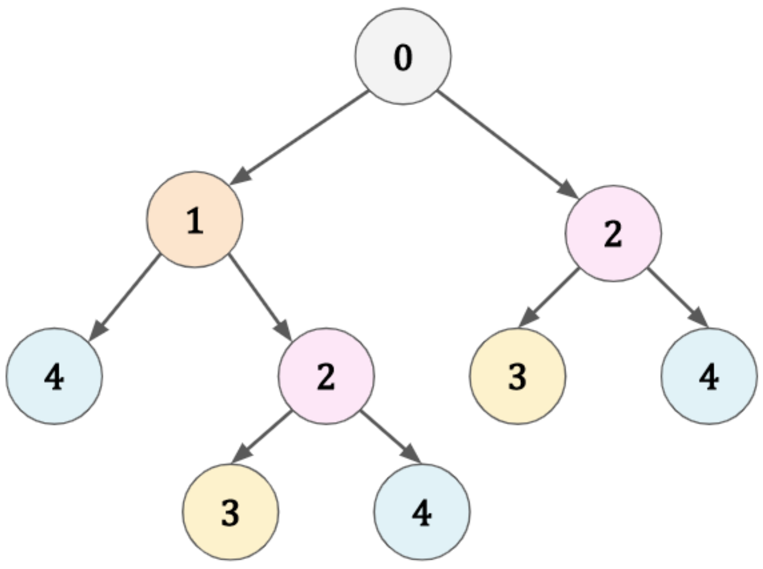
Subtasks

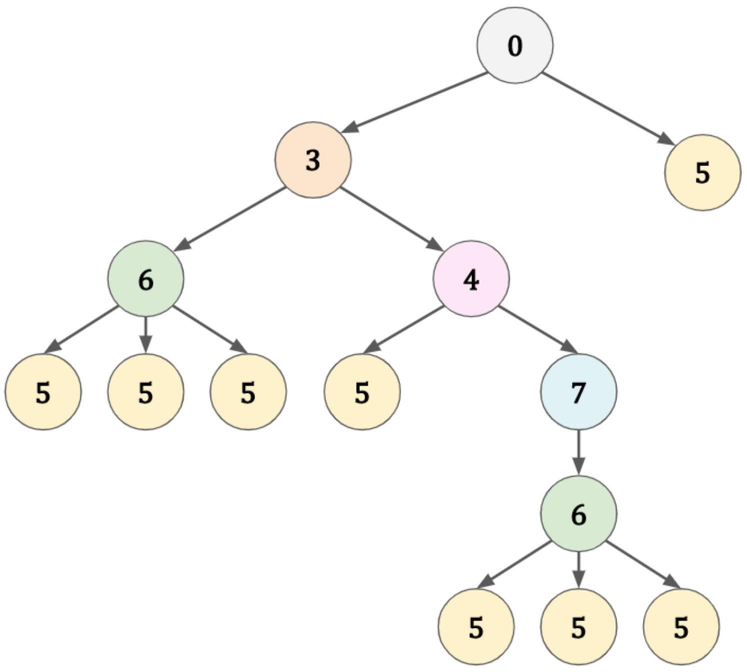
1. **(3 Points)**: Sample.
2. **(10Points)**: $T=1$ and $Q=0$.
3. **(25Points)**: $T=0$ and $Q=0$.
4. **(22Points)**: $T=0$ and $N \leq 10$ and $C_i \leq 3$ ($0 \leq i \leq N-1$).
5. **(25Points)**: $T=0$ and $N \leq 500$ and $C_i \leq 20$ ($0 \leq i \leq N-1$).
6. **(15Points)**: $T=0$.

Visualization

To help you visualize the input in the given sample test cases, you can refer to the following images which represent the dependency tree of widget 0 for sample 1, 2, and 3, respectively.







Warning

The I/O files are large. Please use fast I/O methods.

Sample Input 1

```
3 1000000
1 1
1 2
1 0
0 0
```

Sample Output 1

```
Invalid
```

Sample Input 2

```
7 1000000
2 1 2
2 4 2
2 3 4
0
0
3 1 4 3
3 2 3 4
3 0
1 4
2 2 3
2 3 4
```

Sample Output 2

```
9
7
8
9
```

Sample Input 3

```
8 1000000
2 5 3
1 6
2 6 1
2 4 6
2 5 7
0
3 5 5 5
1 6
10 0
2 7 3
3 4 3 1
1 2
2 1 4
1 5
2 2 5
1 6
1 5
1 5
3 1 7 5
```

Sample Output 3

```
14
13
13
0
9
14
14
12
14
14
14
```

Enjoying Elderberries

One day, a flock of hungry birds find a huge tree of elderberries. They want to eat all berries of this tree. These birds quickly alight on some leaves of the tree and eat all berries at where they stand. Now they are thinking of who will eat other berries ...

Since the flock has some *giant* birds and some *tiny* birds, and they observe that the tree has some *big* branches and some *small* branches, fairly distributing these berries among all birds turns out to be a tough problem.

Formally, the tree of elderberries can be presented as a tree with the following properties:

- The tree has n vertices numbered from 1 to n. The root of the tree is vertex 1.
- Each non-leaf vertex (vertex having at least one child) is called a *branch*. A branch is classified as either *big* or *small*. The root is always a **big** branch.
- On each leaf of the tree, there is either one *giant* bird, one *tiny* bird or one elderberry.

The process of determining which bird eats which berry is as below:

- First, every bird and every berry has a *label*. A label is a non-empty string of **at most five** lowercase English characters.
- Second, every bird has a *controlled area*. Consider the bird at vertex v:

   - If it is a **tiny** bird, its *controlled area* is the subtree rooted at vertex p_v, where p_v is the parent of v.
   - If it is a **giant** bird, its *controlled area* is the subtree rooted at vertex b_v, where b_v is the **closest** vertex to v among all **ancestors** of v which are **big** branches.

- Finally, for each berry, the bird who eats it is determined using the following rules:
   - A bird can only eat berries having the same label with it.
   - A bird can only eat berries inside its *controlled area*.
   - If there is more than one bird satisfying the two above conditions, only one bird with **smallest** *controlled area* eats it.

All birds also set some rules for labeling birds and berries:

- A bird or a berry can have same label with other birds and berries. However, no groups of **eight or more** birds have the same label.
- Each berry is inside the *controlled area* of at least one bird with same label.
- No two birds with same label have the same *controlled area*.
- From all of the above, it can be proven that the bird eating every berry can be uniquely determined.

Sadly, *tiny* birds think that these rules give advantages to *giant* birds over them. They claim that, by the rules of setting *controlled areas*, *giant* birds usually control larger areas, which means having more berries. (Although their thought is not always true.) They want their *controlled areas* to be determined using the rules of giant birds. (In other words, they want all tiny birds become giant birds).

Giant birds accept the *tiny* birds' claim. However, they try to change the labels of some birds and berries, so that if we assume **all tiny birds become giant**, the new set of labels still satisfy all the above conditions. Moreover, after changing labels, **no berries have changed owner** (i.e, every berry is still eaten by the same bird).

They want to change as few labels as possible. Please help them!

Input

The first line contains one integer $n (3 \le n \le 150000)$ — the number of vertices in the elderberry tree. The i-th of the next n lines describes the i-th vertex in either of the following formats:

- 'p_i t_i', where t_i is either 'B' or 'S': meaning that i is a **branch**. $t_i = B$ and $t_i = S$ mean that i is a *big* or *small* branch, respectively.

- ‘p_i T_i L’, where T_i is either ‘G’, ‘T’ or ‘E’ and L is a non-empty string of **at most five** English characters: meaning that i is a **leaf**. $t_i=G$,$t_i=T$ and $t_i=E$ mean that in vertex i there is either a *giant* bird, a *tiny* bird or an *elderberry*. L is the label of this bird or berry.

In both formats, p_i satisfies $1\le p_i<i$ and denotes the parent of vertex i. We assume that $p_1=0$. It is guaranteed that the input describes a valid tree, there is at least one bird and one berry, and all labels satisfy the rules presented above. Remember that **no eight birds** have the same label.

Output

- The first line contains an integer k — the minimum number of labels need changing.
- Each of the next k lines contains an integer x $(1\le x\le n)$ and a label S — meaning that the label of the bird or berry at vertex x is assigned to S. x should be a leaf of the tree. S should be a non-empty string of at most five lowercase English characters.

If there are multiple optimal solutions, you can output any one of them.

Explanation for examples

Below are figures depicting these three examples. Red vertices are big branches, green vertices are small branches, violet vertices have giant birds, yellow vertices have tiny birds and white vertices have berries.

- In the first example:
  - Initially:
    - There are 3 birds with label ‘a’ at vertices 6, 7 and 13. Their *controlled areas* are the subtrees rooted at vertices 2, 5 and 1, respectively.
    - There is 1 bird with label ‘b’ at vertex 12. Its *controlled area* is the subtree rooted at vertex 1.
    - There are 3 elderberries with label ‘a’ at vertices 3, 8 and 11. They are eaten by the birds at vertices 6, 7 and 13 , respectively.
    - There are 2 elderberries with label ‘b’ at vertices 4 and 9. They are both eaten by the bird at vertex 12.

- If all tiny birds become giant birds, both the *controlled area* of the birds at vertices 6 and 7 become the subtree rooted at vertex 2, violating the rules (No two birds with same label having same *controlled area*). Hence, at least one bird's label needs changing. We can change the label of the bird at vertex 6 to 'c'. As the bird at vertex 6 eats the berry at vertex 3, we need to change the label of the bird at vertex 3 as well. The solution in which the bird/berry at vertices 7 and 8 get changed is also accepted.

- In the second example:
  - Initially:
    - The *controlled areas* of the birds at vertices 3 and 6 are subtrees rooted at vertices 1 and 5 , respectively.
    - The bird at vertex 3 eats the berry at vertex 4.
  - If all tiny birds become giant birds, their *controlled areas* are subtrees rooted at vertices 1 and 2, respectively. As a result, the bird at vertex 6 eats the berry at vertex 4, which is invalid, (The owner of every berry must not change). Hence the label of the bird at vertex 6 must be changed.

- In the third example, no changes are necessary.

Sample Input 1

```
13
0 B
1 B
2 E a
2 E b
2 S
5 G a
5 T a
5 E a
5 E b
1 S
10 E a
10 G b
1 T a
```

Sample Output 1

```
2
3 c
6 c
```

Sample Input 2

```
6
0 B
1 B
1 T a
2 E a
2 S
5 T a
```

Sample Output 2

```
1
6 b
```

Sample Input 3

```
6
0 B
1 G y
1 E y
1 E z
1 T z
1 E z
```

Sample Output 3

```
0
```