A Odd Opportunities

ACM discovered there is a secret worldwide organization (Mafia) that operates in complete secrecy. Each member of Mafia has contacts to some other members, but not necessarily to all of them.

The new (recently elected) head of the Mafia came with a brilliant plan to make their operation even more secret: Some members will have to eliminate some of their contacts and to stop communicating with them. It has been announced that it is very important whether the number of contacts is odd or even. Each member has been instructed whether the number of their contacts must remain an odd or an even number.

A little bit of chaos developed after this announcement. Nobody understands why this is so important, but nobody dares to protest. Your task is to write a program that will suggest a suitable solution and select contacts to be dropped.

Input Specification

The input will consist of several test scenarios. Each scenario starts by a line with two positive integers V and E. V is the number of Mafia members ($1 \leq V \leq 1000$) and E is the total number of contacts ($0 \leq E \leq V.(V-1)/2$).

Then there are E lines describing individual contacts. Each line contains two integers v_1 and v_2, which means that there is a contact between members v_1 and v_2 ($1 \leq v_1, v_2 \leq N$, $v_1 \neq v_2$).

Then there is one more line in each scenario containing exactly V lowercase characters. Each character corresponds to one member and is either "`o`" (the number of contacts must remain odd) or "`e`" (the number of contacts must remain even). Obviously, the first character corresponds to member 1, second character to member 2, etc.

The last scenario is followed by a line containing two zeros.

Output Specification

For each scenario, output a subset of original contacts such that will satisfy the even/odd conditions for all members. On the first line, output one integer number R — the number of contacts that remain ($0 \leq R \leq E$). Then, output R lines, each of them specifying two members that remain in contact. Make sure no pair appears more than once.

If it is not possible to find a suitable subset of contacts, output the word "`impossible`". If there are more correct solutions, you may choose any of them.

Sample Input

```
5 6
1 2
2 3
3 4
4 5
1 3
1 4
oeooo
3 1
1 2
oeo
5 0
eeeee
5 0
eeoee
5 0
eeoeo
4 2
1 2
4 3
eeee
0 0
```

Output for Sample Input

```
3
4 5
4 3
1 4
impossible
0
impossible
impossible
0
```

B Peculiar Primes

The level of corruption in some countries is really high. It is hard to imagine that these unethical manners have already hit the academic field. Some rumors are spreading that some students tried to bribe their lecturers to get better grades. Would you believe it?

But the real situation may be even much worse. ACM has a very strong suspicion that somebody has bribed mathematicians in the Academy of Sciences in order to forge some of their results. In particular, it is suspected that a very influential person wants to prefer some prime numbers over others.

It is said that many mathematicians have already completely stopped using some primes and they create only those numbers that can be "assembled" without those primes. Your task is to verify this hypothesis.

Given a set of prime numbers, your program should output all integer numbers that can be created solely by multiplying these primes, without using any other primes.

Input Specification

The input will consist of several test scenarios. Each scenario starts by a line with a single positive integer N $(1 \leq N \leq 10)$ — the number of primes in the set.

On the second line of a scenario, there are N integer numbers $2 \leq P_1 < P_2 < P_3 < \ldots < P_N < 10000$, separated by a space. You are guaranteed that all these numbers are prime.

On the third line of each scenario, there are 2 integers X and Y $(1 \leq X \leq Y < 2^{31})$, separated by a space.

The last scenario is followed by a line containing single zero.

Output Specification

Your task is to print all positive integer numbers in the closed interval $[X, Y]$ that have no other prime factors than those given in the input (P_i). Print all such numbers in the increasing order, with no duplicates and separated by a single comma character ("`,`"). If there are no such numbers, print the word "`none`" instead.

Note that the number 1 does not need any primes to be constructed and is therefore always allowed.

Sample Input

```
1
3
1 12
2
2 3
10 20
3
2 3 5
20 30
1
17
20 30
0
```

Output for Sample Input

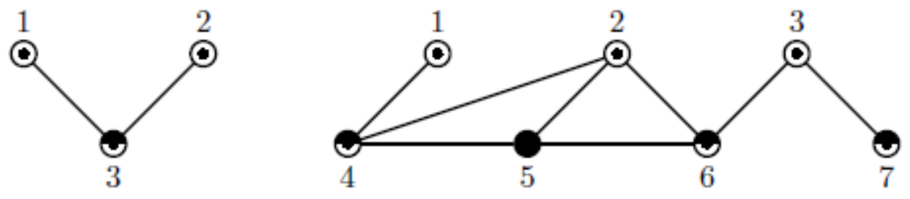
```
1,3,9
12,16,18
20,24,25,27,30
none
```

Problem C K-Graph Oddity

Input file:	`kgraph.in`
Output file:	`kgraph.out`

You are given a connected undirected graph with an odd number of vertices. The degree of the vertex, by definition, is the number of edges incident to it. In the given graph the degree of each vertex does not exceed an odd number k. Your task is to color the vertices of this graph into at most k distinct colors, so that the colors of any two adjacent vertices are distinct.

The pictures below show two graphs. The first one has 3 vertices and the second one has 7 vertices. In both graphs degrees of the vertices do not exceed 3 and the vertices are colored into at most 3 different colors marked as '◔', '◑' and '◕'.



Input

The first line of the input file contains two integer numbers n and m, where n is the number of vertices in the graph ($3 \le n \le 9999$, n is odd), m is the number of edges in the graph ($2 \le m \le 100\,000$). The following m lines describe edges of the graph, each edge is described by two integers a_i, b_i ($1 \le a_i, b_i \le n$, $a_i \ne b_i$) — the vertex numbers connected by this edge. Each edge is listed at most once. The graph in the input file is connected, so there is a path between any pair of vertices.

Output

On the first line of the output file write a single integer number k — the minimal odd integer number, such that the degree of any vertex does not exceed k. Then write n lines with one integer number c_i ($1 \le c_i \le k$) on a line that denotes the color of i-th vertex.

The colors of any two adjacent vertices must be different. If the graph has multiple different colorings, print any of them. At least one such coloring always exists.

Sample input and output

`kgraph.in`	`kgraph.out`

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

`7 8`
`1 4`
`4 2`
`2 6`
`6 3`
`3 7`
`4 5`
`5 6`
`5 2`
	`3`
	`1`
	`1`
	`1`
	`2`
	`3`
	`2`
	`2`

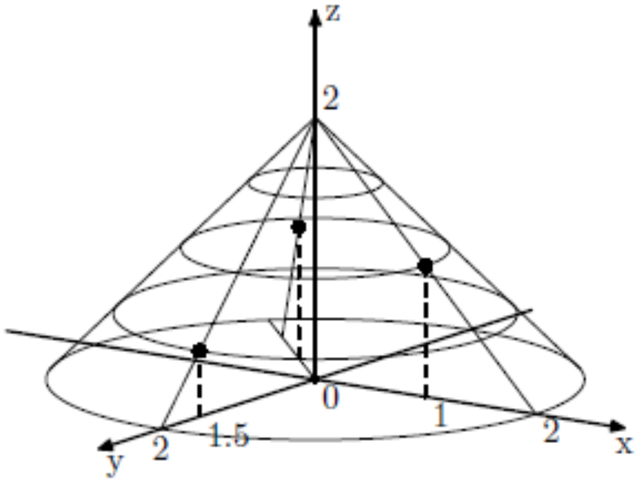
Problem D. Dome of Circus

Input file: `dome.in`
Output file: `dome.out`

A travelling circus faces a tough challenge in designing the dome for its performances. The circus has a number of shows that happen above the stage in the air under the dome. Various rigs, supports, and anchors must be installed over the stage, but under the dome. The dome itself must rise above the center of the stage and has a conical shape. The space under the dome must be air-conditioned, so the goal is to design the dome that contains minimal volume.

You are given a set of n points in the space; (x_i, y_i, z_i) for $1 \le i \le n$ are the coordinates of the points in the air above the stage that must be covered by the dome. The ground is denoted by the plane $z = 0$, with positive z coordinates going up. The center of the stage is on the ground at the point $(0, 0, 0)$.

The tip of the dome must be located at some point with coordinates $(0, 0, h)$ with $h > 0$. The dome must have a conical shape that touches the ground at the circle with the center in the point $(0, 0, 0)$ and with the radius of r. The dome must contain or touch all the n given points. The dome must have the minimal volume, given the above constraints.



Input

The first line of the input file contains a single integer number n ($1 \le n \le 10\,000$) — the number of points under the dome. The following n lines describe points with three floating point numbers x_i, y_i, and z_i per line — the coordinates of i-th point. All coordinates do not exceed 1000 by their absolute value and have at most 2 digits after decimal point. All z_i are positive. There is at least one point with non-zero x_i or y_i.

Output

Write to the output file a single line with two floating point numbers h and r — the height and the base radius of the dome. The numbers must be precise up to 3 digits after decimal point.

Sample input and output

`dome.in`	`dome.out`

`1`
`1.00 0.00 1.00`
	`3.000 1.500`

`2`
`1.00 0.00 1.00`
`0.00 1.50 0.50`
	`2.000 2.000`

`3`
`1.00 0.00 1.00`
`0.00 1.50 0.50`
`-0.50 -0.50 1.00`
	`2.000 2.000`

Problem E. Evacuation Plan

Input file:	`evacuation.in`
Output file:	`evacuation.out`

Flatland government is building a new highway that will be used to transport weapons from its main weapon plant to the frontline in order to support the undergoing military operation against its neighbor country Edgeland. Highway is a straight line and there are n construction teams working at some points on it.

During last days the threat of a nuclear attack from Edgeland has significantly increased. Therefore the construction office has decided to develop an evacuation plan for the construction teams in case of a nuclear attack. There are m shelters located near the constructed highway. This evacuation plan must assign each team to a shelter that it should use in case of an attack.

Each shelter entrance must be securely locked from the inside to prevent any damage to the shelter itself. So, for each shelter there must be some team that goes to this shelter in case of an attack. The office must also supply fuel to each team, so that it can drive to its assigned shelter in case of an attack. The amount of fuel that is needed is proportional to the distance from the team's location to the assigned shelter. To minimize evacuation costs, the office would like to create a plan that minimizes the total fuel needed.

Your task is to help them develop such a plan.

Input

The first line of the input file contains n — the number of construction teams ($1 \le n \le 4000$). The second line contains n integer numbers — the locations of the teams. Each team's location is a positive integer not exceeding 10^9, all team locations are different.

The third line of the input file contains m — the number of shelters ($1 \le m \le n$). The fourth line contains m integer numbers — the locations of the shelters. Each shelter's location is a positive integer not exceeding 10^9, all shelter locations are different.

The amount of fuel that needs to be supplied to a team at location x that goes to a shelter at location y is equal to $|x - y|$.

Output

The first line of the output file must contain z — the total amount of fuel needed. The second line must contain n integer numbers: for each team output the number of the shelter that it should be assigned to. Shelters are numbered from 1 to m in the order they are listed in the input file.

Sample input and output

`evacuation.in`	`evacuation.out`

`3`
`1 2 3`
`2`
`2 10`
	`8`
	`1 1 2`

Problem G Bureaucracy

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

Long ago, in a kingdom far, far away the king decided to keep a record of all laws of his kingdom. From that moment whenever a new law was passed, a corresponding record was added to the law archive.

Many centuries later lawyers discovered that there were only two types of laws in the kingdom:

- *direct law*, that states a new norm;
- *canceling law*, that cancels one of the previous laws.

The law is considered *active* if and only if there is no active law that cancels it.

You are to write program that finds out which laws are still active.

Input

The first line of the input file contains an integer number n ($1 \le n \le 100\,000$) — the number of passed laws.

The following n lines describe one law each. Each description has one of the following formats:

- "`declare`", meaning that a direct law was passed.
- "`cancel` i", where i is the number of law being cancelled by this one.

The laws are numbered from one.

Output

The first line of the output file must contain the number of active laws. Following lines must contain numbers of these laws listed in increasing order.

Example

bureau.in	bureau.out
5 declare cancel 1 declare cancel 2 cancel 3	3 1 4 5

Problem H. Homo or Hetero?

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

Consider a list of numbers with two operations:

- `insert number` — adds the specified number to the end of the list.
- `delete number` — removes the first occurrence of the specified number from the list. If the list does not contain the number specified, no changes are performed.

For example: the result of the insertion of a number 4 to the list $[1, 2, 1]$ is the list $[1, 2, 1, 4]$. If we delete the number 1 from this list, we get the list $[2, 1, 4]$, but if we delete the number 3 from the list $[1, 2, 1, 4]$, the list stays unchanged.

The list is *homogeneous* if it contains at least two equal numbers and the list is *heterogeneous* if it contains at least two different numbers. For example: the list $[2, 2]$ is homogeneous, the list $[2, 1, 4]$ is heterogeneous, the list $[1, 2, 1, 4]$ is both, and the empty list is neither homogeneous nor heterogeneous.

Write a program that handles a number of the operations `insert` and `delete` on the empty list and determines list's homogeneity and heterogeneity after each operation.

Input

The first line of the input file contains an integer number n — the number of operations to handle ($1 \le n \le 100\,000$).

Following n lines contain one operation description each. The operation description consists of a word "`insert`" or "`delete`", followed by an integer number k — the operation argument ($-10^9 \le k \le 10^9$).

Output

For each operation output a line, containing a single word, describing the state of the list after the operation:

- "`both`" — if the list is both homogeneous and heterogeneous.
- "`homo`" — if the list is homogeneous, but not heterogeneous.
- "`hetero`" — if the list is heterogeneous, but not homogeneous.
- "`neither`" — if the list is neither homogeneous nor heterogeneous.

Example

homo.in	homo.out
11 insert 1 insert 2 insert 1 insert 4 delete 1 delete 3 delete 2 delete 1 insert 4 delete 4 delete 4	neither hetero both both hetero hetero hetero neither homo neither neither

I. Smoking gun

Andy: "Billy the Kid fired first!"

Larry: "No, I'm sure I heard the first shot coming from John!"

The arguments went back and forth during the trial after the big shoot-down, somewhere in the old wild west. Miraculously, everybody had survived (although there were serious injuries), but nobody could agree about the exact sequence of shots that had been fired. It was known that everybody had fired at most one shot, but everything had happened very fast. Determining the precise order of the shots was important for assigning guilt and penalties.

But then the sheriff, Willy the Wise, interrupted: "Look, I've got a satellite image from the time of the shooting, showing exactly where everybody was located. As it turns out, Larry was located much closer to John than to Billy the Kid, while Andy was located just slightly closer to John than to Billy the Kid. Thus, because sound travels with a finite speed of 340 meters per second, Larry may have heard John's shot first, even if Billy the Kid fired first. But, although Andy was closer to John than to Billy the Kid, he heard Billy the Kid's shot first – so we know for a fact that Billy the Kid was the one who fired first!

Your task is to write a program to deduce the exact sequence of shots fired in situations like the above.

Input

On the first line a positive integer: the number of test cases, at most 100. After that per test case:

- one line with two integers n ($2 \leq n \leq 100$) and m ($1 \leq m \leq 1\,000$): the number of people involved and the number of observations.
- n lines with a string S, consisting of up to 20 lower and upper case letters, and two integers x and y ($0 \leq x, y \leq 1\,000\,000$): the unique identifier for a person and his/her position in Cartesian coordinates, in metres from the origin.
- m lines of the form "`S1 heard S2 firing before S3`", where $S1$, $S2$ and $S3$ are identifiers among the people involved, and $S2 \neq S3$.

If a person was never mentioned as $S2$ or $S3$, then it can be assumed that this person never fired, and only acted as a witness. No two persons are located in the same position.

The test cases are constructed so that an error of less than 10^{-7} in one distance calculation will not affect the output.

Output

Per test case:

- one line with the ordering of the shooters that is compatible with all of the observations, formatted as the identifiers separated by single spaces.

If multiple distinct orderings are possible, output "`UNKNOWN`" instead. If no ordering is compatible with the observations, output "`IMPOSSIBLE`" instead.

Sample in- and output

Input

```
3
4 2
BillyTheKid 0 0
Andy 10 0
John 19 0
Larry 20 0
Andy heard BillyTheKid firing before John
Larry heard John firing before BillyTheKid
2 2
Andy 0 0
Beate 0 1
Andy heard Beate firing before Andy
Beate heard Andy firing before Beate
3 1
Andy 0 0
Beate 0 1
Charles 1 3
Beate heard Andy firing before Charles
```

Output

```
BillyTheKid John
IMPOSSIBLE
UNKNOWN
```