Problem A
Baggage

An airline has two flights leaving at about the same time from ICPCity, one to city B and one to city A. The airline also has n counters where passengers check their baggage. At each counter there is a pair of identical baggage bins, one for city B and one for city A.

Just before the flights depart, each pair of baggage bins is moved by a motorized cart to a sorting area. The cart always moves two bins at a time, one for city B and one for city A. After all the bins have been moved, they line up in the sorting area like this:

B A B A B A ... B A

That is, there are $2n$ baggage bins in a row, starting with a bin for city B, then one for city A, and so forth. The task now is to reorder them so all the baggage bins for city A precede the baggage bins for city B. Then the bins can be loaded on the appropriate aircraft.

The reordering is done by moving pairs of adjacent baggage bins (not necessarily B then A), again via the motorized cart. For proper balance, the cart must always carry two bins, never just one. A pair of bins must always be moved to an empty space that is at least two bins wide. On the left of the first bin are some empty spaces that can be used as needed during the reordering.

When the reordering process begins, the bin locations are numbered from 1 (initially containing the leftmost B baggage bin) to $2n$ (initially containing the rightmost A baggage bin). There are $2n$ initially empty spaces to the left of the bins, numbered from 0 to $-2n+1$, as shown in Figure **??** for the case $n = 4$.

								B	A	B	A	B	A	B	A
-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8

Figure A.1: Initial configuration of bins and empty spaces for $n = 4$

Given n, find a shortest sequence of moves that will reorder the bins so that all the A bins are to the left of all the B bins. At the end of the process, it is possible that the leftmost A bin is at some location other than 1, but the bins must be adjacent in a sequence of $2n$ locations.

Input

The input consists of a single test case, which consists of the integer n ($3 \leq n \leq 100$).

Output

Display a shortest sequence of moves that will correctly reorder the bins. Each move is of the form "f `to` t", where f and t are integers representing the movement of the bins in locations f and $f+1$ to locations t and $t+1$. If multiple solutions are possible, display any one of them.

Sample Input 1

```
5
```

Sample Output 1

```
8 to -1
3 to 8
6 to 3
0 to 6
9 to 0
```

Sample Input 2

```
8
```

Sample Output 2

```
10 to -1
3 to 10
14 to 3
7 to 14
0 to 7
11 to 0
4 to 11
15 to 4
```

Problem B
Surely You Congest

You are in charge of designing an advanced centralized traffic management system for smart cars. The goal is to use global information to instruct morning commuters, who must drive downtown from the suburbs, how best to get to the city center while avoiding traffic jams.

Unfortunately, since commuters know the city and are selfish, you cannot simply tell them to travel routes that take longer than normal (otherwise they will just ignore your directions). You can only convince them to change to different routes that are equally fast.

The city's network of roads consists of intersections that are connected by bidirectional roads of various travel times. Each commuter starts at some intersection, which may vary from commuter to commuter. All commuters end their journeys at the same place, which is downtown at intersection 1. If two commuters attempt to start travelling along the same road in the same direction at the same time, there will be congestion; you must avoid this. However, it is fine if two commuters pass through the same intersection simultaneously or if they take the same road starting at different times.

Determine the maximum number of commuters who can drive downtown without congestion, subject to all commuters starting their journeys at exactly the same time and without any of them taking a suboptimal route.

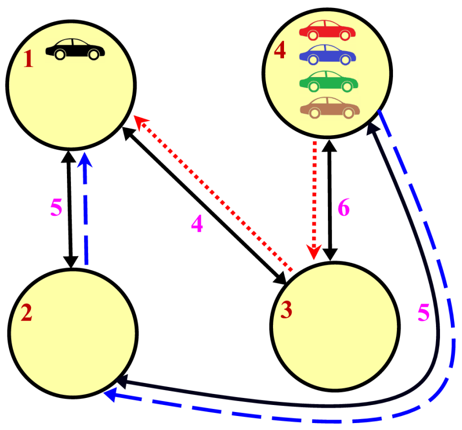

Figure B.1: Illustration of Sample Input 2.

In Figure **??**, cars are shown in their original locations. One car is already downtown. Of the cars at intersection 4, one can go along the dotted route through intersection 3, and another along the dashed route through intersection 2. But the remaining two cars cannot reach downtown while avoiding congestion. So a maximum of 3 cars can reach downtown with no congestion.

Input

The input consists of a single test case. The first line contains three integers n, m, and c, where n ($1 \leq n \leq 25\,000$) is the number of intersections, m ($0 \leq m \leq 50\,000$) is the number of roads, and c ($0 \leq c \leq 1\,000$) is the number of commuters. Each of the next m lines contains three integers x_i, y_i, and t_i describing one road, where x_i and y_i ($1 \leq x_i, y_i \leq n$) are the distinct intersections the road connects, and t_i ($1 \leq t_i \leq 10\,000$) is the time it takes to travel along that road in either direction. You may assume that downtown is reachable from every intersection. The last line contains c integers listing the starting intersections of the commuters.

Output

Display the maximum number of commuters who can reach downtown without congestion.

Sample Input 1	Sample Output 1

```
3 3 2
1 2 42
2 3 1
2 3 1
2 3
```
	`2`

Sample Input 2	Sample Output 2

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

Problem C
Factors

The fundamental theorem of arithmetic states that every integer greater than 1 can be uniquely represented as a product of one or more primes. While unique, several arrangements of the prime factors may be possible. For example:

$$\begin{aligned} 10 &= 2 \cdot 5 \\ &= 5 \cdot 2 \end{aligned} \qquad\qquad \begin{aligned} 20 &= 2 \cdot 2 \cdot 5 \\ &= 2 \cdot 5 \cdot 2 \\ &= 5 \cdot 2 \cdot 2 \end{aligned}$$

Let $f(k)$ be the number of different arrangements of the prime factors of k. So $f(10) = 2$ and $f(20) = 3$.

Given a positive number n, there always exists at least one number k such that $f(k) = n$. We want to know the smallest such k.

Input

The input consists of at most 1 000 test cases, each on a separate line. Each test case is a positive integer $n < 2^{63}$.

Output

For each test case, display its number n and the smallest number $k > 1$ such that $f(k) = n$. The numbers in the input are chosen such that $k < 2^{63}$.

Sample Input 1	Sample Output 1

1
2
3
105
	1 2
	2 6
	3 12
	105 720

Problem D
Maze Reduction

Jay runs a small carnival that has various rides and attractions. Unfortunately, times are tough. A recent roller coaster accident, flooding in the restrooms, and an unfortunate clown incident have given Jay's carnival a bad reputation with the public. With fewer paying customers and reduced revenue, he will need to cut some costs to stay in business.

One of the biggest carnival attractions is a large, confusing maze. It consists of a variety of circular rooms connected by narrow, twisting corridors. Visitors love getting lost in it and trying to map it out. It has come to Jay's attention that some of the rooms might be effectively identical to each other. If that's the case, he will be able to reduce its size without anyone noticing.

Two rooms A and B are *effectively identical* if, when you are dropped into either room A or B (and you know the map of the maze), you cannot tell whether you began in A or B just by exploring the maze. The corridor exits are evenly spaced around each room, and you cannot mark or leave anything in a room (in particular, you cannot tell whether you have previously visited it). The only identifying feature that rooms have is their number of exits. Corridors are also twisty enough to be indistinguishable from each other, but when you enter a room you know which corridor you came from, so you can navigate a little by using the order they appear around the room.

Jay has appealed to the Association for Carnival Mazery for help. That's you! Write a program to determine all the sets of effectively identical rooms in the maze.

Input

The input consists of a single test case. The first line contains an integer n, the number of rooms in the maze ($1 \leq n \leq 100$). Rooms are numbered from 1 to n. Following this are n lines, describing each room in order. Each line consists of an integer k, indicating that this room has k corridors ($0 \leq k < 100$), and then k distinct integers listing the rooms each corridor connects to (in clockwise order, from an arbitrary starting point). Rooms do not connect to themselves.

Output

Display one line for each maximal set of effectively identical rooms (ignoring sets of size 1) containing the room numbers in the set in increasing order. Order the sets by their smallest room numbers. If there are no such sets, display `none` instead.

Sample Input 1

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

Sample Output 1

```
2 4
5 6
7 8 9 10 11 12 13
```

Sample Input 2

```
6
3 3 4 5
0
1 1
1 1
2 1 6
1 5
```

Sample Output 2

```
none
```

Problem E
Surveillance

The International Corporation for Protection and Control (ICPC) develops efficient technology for, well, protection and control. Naturally, they are keen to have their own headquarters protected and controlled. Viewed from above, the headquarters building has the shape of a convex polygon. There are several suitable places around it where cameras can be installed to monitor the building. Each camera covers a certain range of the polygon sides (building walls), depending on its position. ICPC wants to minimize the number of cameras needed to cover the whole building.

Input

The input consists of a single test case. Its first line contains two integers n and k ($3 \le n \le 10^6$ and $1 \le k \le 10^6$), where n is the number of walls and k is the number of possible places for installing cameras. Each of the remaining k lines contains two integers a_i and b_i ($1 \le a_i, b_i \le n$). These integers specify which walls a camera at the i^{th} place would cover. If $a_i \le b_i$ then the camera covers each wall j such that $a_i \le j \le b_i$. If $a_i > b_i$ then the camera covers each wall j such that $a_i \le j \le n$ or $1 \le j \le b_i$.

Output

Display the minimal number of cameras that suffice to cover each wall of the building. The ranges covered by two cameras may overlap. If the building cannot be covered, display `impossible` instead.

Sample Input 1

```
100 7
1 50
50 70
70 90
90 40
20 60
60 80
80 20
```

Sample Output 1

```
3
```

Sample Input 2

```
8 2
8 3
5 7
```

Sample Output 2

```
impossible
```

Sample Input 3

```
8 2
8 4
5 7
```

Sample Output 3

```
2
```

Problem F
Support Vector Machine

In machine learning, the Support Vector Machine (SVM) is a popular classifier. Given an n-dimensional vector $x \in \mathbb{R}^n$, an SVM produces a label $h(x) \in \{-1, +1\}$ depending on where x lies relative to an n-dimensional hyperplane.

An example application is email spam classification. We could convert the text of an email into a vector x of counts of word occurrences in the email. Each vector element would represent the count for one dictionary word. Then a trained SVM could classify the email as 'spam' ($h(x) = +1$) or 'non-spam' ($h(x) = -1$).

Recall that for vectors $x, y \in \mathbb{R}^n$,

- x_i represents the ith element of x,
- $x^T y = \sum_{i=1}^n x_i y_i$ is the vector inner product of x and y, and
- $||x|| = \sqrt{x^T x}$ is the length of vector x.

The parameters that define the SVM are $w \in \mathbb{R}^n$ and $b \in \mathbb{R}$. The distance from a point x to the hyperplane defined by (w, b) is

$$d(x) = (w^T x + b)/||w||$$

so that the hyperplane is the set $H = \{\, x \mid d(x) = 0 \,\}$ and is $b/||w||$ units from the origin at its closest. Note also that w is orthogonal to the hyperplane. The SVM classifier is formally defined as

$$h(x) = \begin{cases} +1, & \text{if } d(x) \geq 0 \\ -1, & \text{otherwise} \end{cases}$$

For this problem you are given SVM parameters and a list of query points. For each query x, identify $d(x)$.

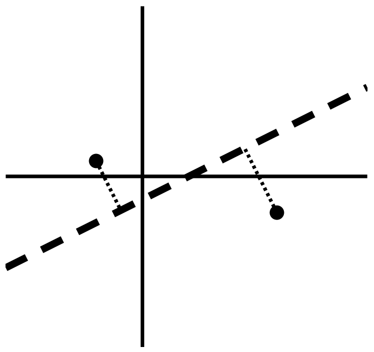

Figure F.1: Rendering of the hyperplane (thick dashed line) and query points (circles) for the sample input. Other dashed lines connected each query point to its closest point on the hyperplane.

Input

The input contains one test case. The first line contains an integer $1 \le n \le 1000$. The second line contains $n+1$ floating-point values which are the values of $w_1, w_2, \ldots, w_n$ and b. Each remaining line (up to the end of file) is a query vector x, given as n floating point values (in the same order as the elements of w). There are at most 1000 queries. All floating point values in the input are in the range $[-1000, 1000]$ with at most 5 digits past the decimal point. At least one element of w will be non-zero.

Output

For each query x, print the value of $d(x)$. The answer should be accurate to within 10^{-4} relative or absolute error.

Sample Input 1

```
2
-1 2 3
-3 1
8.71 -2.35
```

Sample Output 1

```
3.57771
-4.65549
```