In order to automate his workload at the factory, Mirko wants to put up to use his old box-sorting robot. There are **N** boxes in the factory, and each box is labelled with an **unique** integer in range 1 to N. Mirko's task is to sort the boxes in the ascending order of their labels.

Sorting robot can only perform one specific **operation**: given the sequence of positions, robot can do a **cyclic swap** of boxes at those positions. Given sequence does not contain any position more than once.

For example, let's assume that the boxes are currently in order [4, 1, 5, 2, 3], and Mirko provides his robot with the sequence [2, 1, 3]. Robot will then rearrange the boxes so that the second box will go to position 1, first box will go to the position 3, and the third one will take the position 2. Obtained sequence of labels is [1, 5, 4, 2, 3].

Write a program that will sort the boxes using the **minimal number** of operations. Each sequence given to the robot can be arbitrary long.

INPUT

The first line of input contains integer **N** ($2 \leq \mathbf{N} \leq 1000$), the number of boxes in the factory.

The following line contains N integers in range 1 to N, labels of boxes in order. No integer appears twice.

OUTPUT

The first line should contain the integer **X**, **minimum number of operations** required.

The following **X** lines should contain the sequences given to robot, one sequence per line.

Each line should start with the length of the sequence, followed by a colon, a single white space, and then space seperated sequence of positions.

NOTE: There may be multiple solutions and you can output any of the.

SCORING

Program will receive 50% of the points assigned to that test case if the number of operations is not minimal but is not greater than 1000. Of course, operations provided should yield an sorted sequence.

EXAMPLE TEST DATA

input	**input**	**input**
3 3 2 1 **output** 1 2: 3 1	5 2 3 1 5 4 **output** 2 3: 1 2 3 2: 5 4	5 1 2 3 4 5 **output** 0

Recently there was a population census in Mirko's country. Along with numerous other data that was being collected, very important part was the data about television ratings.
Each of the **N** citizens provided two timestamps in the following format:

HH:MM:SS – HH:MM:SS

The first timestamp describes the time of the day when citizen started watching television, and the second one the time when that citizen stopped watching. Citizen also watched television during the first and the last second of the given interval. Note that it is possible to start watching before midnight, eg., at 23:45:30, and not finish until the next day, eg., at 01:15:00.

After all the data has been collected, statisticians are gathered in order to analyse it.
We define the popularity of some second as the **total number of citizens that were watching television during that second**. Furthermore, the popularity of the given time interval is defined as the sum of popularities of seconds contained within that interval, divided by the length of the interval.

Calculate the popularities of the **Q** given time intervals that are of special interest to the statisticians.

INPUT

The first line of input contains integer **N** (**N** ≤ 100 000), the number of citizens.
The following **N** lines each contain two timestamps given by that citizen, in the format described above (0 ≤ **HH** ≤ 23, 0 ≤ **MM** ≤ 59, 0 ≤ **SS** ≤ 59).
In the following line there is an integer **Q** (**Q** ≤ 100 000), describing the number of time intervals that statisticians are interested in.
The following **Q** lines contain time intervals in the same format as above.

OUTPUT

For each of the **Q** given intervals, output it's popularity in seperate line.
Solution will be accepted if apsolute or relative error is at most 10^{-6}.

SCORING

In test cases worth a total of 25% points, **N** ≤ 500 and **Q** ≤ 500 will hold.
In test cases worth a total of 25% points, **N** ≤ 500 and **Q** ≤ 100 000 will hold.
In test cases worth a total of 25% points, **N** ≤ 100 000 and **Q** ≤ 500 will hold.

EXAMPLE TEST DATA

input

```
5
00:00:00 - 00:00:01
00:00:01 - 00:00:03
00:00:00 - 00:00:02
00:00:05 - 00:00:09
00:00:06 - 00:00:06
5
00:00:00 - 00:00:03
00:00:07 - 00:00:09
00:00:06 - 00:00:06
00:00:05 - 00:00:09
00:00:00 - 00:00:09
```

output

```
2.0000000000
1.0000000000
2.0000000000
1.2000000000
1.4000000000
```

input

```
3
00:00:00 - 10:00:00
10:00:00 - 00:00:00
01:01:01 - 02:02:02
4
00:00:00 - 23:59:59
23:59:59 - 23:59:58
23:59:59 - 23:59:59
08:34:43 - 12:22:17
```

output

```
1.0424074074
1.0424074074
1.0000000000
1.0000732332
```

In the far away land there is a big river and a number of villages beside it. Villages are numbered 0 through **M**, in order along the river. Distance bewteen consecutive villages is exactly 1 mile.

Mirko lives in the village marked with 0. He is in the business of transporting people between villages with his boat. Today, Mirko will travel from his village to village **M**, and he will also transport some people along the way.

There are **N** people that wish to travel today, and for each of them we know their starting point as well as their destination. Mirko's boat can accommodate as many people as needed.

Let's say for example that person A is traveling from village 2 to village 8, and B from village 6 to village 4. Mirko will, as always, start from his village 0, pick up A at village 2, pickup B at 6, go back to 4 and leave B there, proceed to village 8, leave A there, and then continue to his final destination, village **M**. This scenario is given in the first test case below.

Write a program that will find the **minimum number of miles** that Mirko must travel in order to transport everyone to their destinations and reach village **M** at the end.

INPUT

The first line of input contains integers **N** and **M** ($\mathbf{N} \leq 300\,000$, $3 \leq \mathbf{M} \leq 10^9$).
The following **N** lines contain two integers, starting point and destination for each villagers that want's to travel today. These number will be in range 0 to **M**.

OUTPUT

Output the minimum number of miles that Mirko must travel.

SCORING

In test cases worth a total of 40% points, $\mathbf{N} \leq 5000$ will hold.
In test cases worth a total of 50% points, $\mathbf{M} \leq 2000000$ will hold.

EXAMPLE TEST DATA

input	**input**
2 10 2 8 6 4 **output**	8 15 1 12 3 1 3 9 4 2 7 13

14	12 11 14 11 14 13 **output** 27

Mirko is a truck driver. His job is to travel between cities by roads, loading and unloading the cargo. His truck is so big that it can load unlimited number of packages, but the automated loading system enables only the **last loaded package** to be unloaded. There exists **26 different types of packages**, each of which is denoted by a letter of english alphabet.

The cities are connected by **one-way roads** of length **1 kilometer**. More precisely, there exists 3 types of roads, conveniently denoted by 1, 2 and 3:

1. **each time** Mirko drives down the road of this type, he **must load** exactly one package of the appropriate type for that road
2. **each time** Mirko drives down the road of this type, he **must unload** exactly one package of the appropriate type for that road
3. Mirko can drive down the road of this type **without** loading or unloading packages (no loading/unloading)

Mirko is required not to load/unload any cargo except when driving down the roads of type 2 or 3, as stated above.

Mirko can travel along **E** roads connecting **N** cities. His starts in the city denoted by number 1 and his goal is to reach city denoted by number **N**. When arriving in the city numbered **N**, Mirko's truck is not required to be empty.

Write a program which computes the number of **different ways** that Mirko can travel so that he traverses at most **K** kilometers.

INPUT

The first line of input contains positive integers **N**, **E** and **K** (**$2 \leq N \leq 50$, $1 \leq E \leq 2450$, $1 \leq K \leq 50$**), which denote the number of cities, the number of roads and the maximum number of kilometers that Mirko may traverse before reaching his destination.

The following E lines contain the description of the roads along which Mirko can travel. Each type of road have its own format:

1. **"x y C"**, where x and y are positive integers which describe the direction of the road and C is an **uppercase letter** of english alphabet which denotes the type of package that Mirko must load onto the truck.
2. **"x y c"**, where x and y are positive integers which describe the direction of the road and C is an **lowercase letter** of english alphabet which denotes the type of package that Mirko must unload from the truck.
3. **"x y"**, where x and y are positive integers which describe the direction of the road

In the above formats, the road are traversible when traveling from the city denoted by x to the city denoted by y. Also, it will always be true that **$1 \leq x, y \leq N$**, **$x \neq y$**, and no two roads will connect two cities in the same direction.

OUTPUT

In a single line of output, print the number of different ways Mirko can arrive in the city numbered N, starting from the city numbered 1, while respecting the aforementioned requirements. Since this number can be quite big, print **the remainder of that number when divided by 10007**.

EXAMPLE TEST DATA

input	**input**
2 1 10 1 2 a **output** 0	7 9 5 1 2 A 2 3 B 2 5 5 3 C 3 4 b 3 6 c 3 7 4 7 a 6 7 a **output** 4

Mirko drew a **2*N** by **2*N** chessboard and decided to play the following game:

On each square, he inscribed an integer denoting the **value** of that square. In the middle of the first row (columns N and N+1), he placed two bishops. Mirko now calculates the **lines of sight** of the two bishops: these are squares on the **same diagonal** as one of the bishops, but not on the square covered by any of the bishops.

For instance, if N is 3, the squares within the line of sight of the two bishops (denoted by 'X') at their starting positions (denoted by 'L') are as follows:

```
00LL00
0XXXX0
XX00XX
X0000X
000000
000000
```

In a limited number of moves, Mirko tries to gain the maximum score by the following strategy:

1. Before making any moves, Mirko calculates the sum of the values on squares within the lines of sight

of the two bishops. That is Mirko's initial score.

2. On each turn, Mirko chooses one of the bishops and moves it to a square within its line of sight
3. On its new position, the bishop can now see new squares. The sum of those squares, **previously unseen by any of the bishops from the start of the game**, is now added to Mirko's score.

Write a program which calculates the maximum score that Mirko can achieve in **K** turns.

INPUT

The first line of input contains two integer **N** and **K** ($1 \leq \mathbf{N} \leq 10$, $0 \leq \mathbf{K} \leq 100$), half of the dimension of the chessboard and the number of turns.

The following 2*N lines contain the values of the squares in the corresponding row, 2*N values per row. Those values are integers from range -1 000 000 to 1 000 000, inclusive.

OUTPUT

In a single line of output, print the maximum score that Mirko can achieve.

SCORING

In test cases worth a total of 40% points, **K** will be less or equal to 5.

EXAMPLE TEST DATA

input	**input**
2 0 0 -9 -9 0 0 1 1 0 1 0 0 1 0 0 6 0	2 1 0 -9 -9 0 0 1 1 0 1 0 0 1 0 0 6 0
output	**output**
4	1