Problem A
Prime Digital Roots
Input: `roots.txt`

The digital root of a number is found by adding together the digits that make up the number. If the resulting number has more than one digit, the process is repeated until a single digit remains.

Your task in this problem is to calculate a variation on the digital root — a *prime digital root*. The addition process described above stops when there is only one digit left, but will also stop if the original number, or any of the intermediate numbers (formed by addition) are prime numbers. If the process continues and results in a single digit that is not a prime number, then the original number has no prime digital root.

> An integer *greater than* one is called a prime number if its only positive divisors (factors) are one and itself.
>
> - For example, the first six primes are 2, 3, 5, 7, 11, and 13.
> - Number 6 has four positive divisors: 6, 3, 2, and 1. Thus number 6 is *not* a prime.
> - Caveat: number 1 is *not* a prime.

Here are examples of prime digital roots:

- 1 is not a prime number, so 1 has no prime digital root.
- 3 is a prime number, so the prime digital root of 3 is 3.
- 4 is not a prime number, so 4 has no prime digital root.
- 11 is a prime number, so the prime digital root of 11 is 11.
- 642 is not a prime number, so adding its digits gives $6 + 4 + 2 = 12$. This is not a prime number, so adding again gives $1 + 2 = 3$. This is a prime number, so the prime digital root of 642 is 3.
- 128 is not a prime number, so adding its digits gives $1 + 2 + 8 = 11$. This is a prime number, so the prime digital root of 128 is 11.
- 886 is not a prime number, so adding its digits gives $8 + 8 + 6 = 22$. This is not a prime number, so adding again gives $2+2 = 4$. This is not a prime number, so 886 has no prime digital root.

Input

The input will contain a single integer on each line in the range 0 to 999999 inclusive. The end of the input will be indicated by the value 0.

Output

If the input number has a prime digital root, then the input number must be output right aligned with a field width of 7. It must be followed by a single space, and then by the calculated prime digital root also right aligned with a field width of 7.

If the input number has no prime digital root, then the input number should be output as defined above followed by 4 spaces followed by the word none (in lowercase). The terminating zero should not be output.

Sample Input

```
1
3
4
11
642
128
886
0
```

Output for the Sample Input

```
      1    none
      3       3
      4    none
     11      11
    642       3
    128      11
    886    none
```

Problem B
Ski Slopes
Input: `ski.txt`

A skier wants to ski down from the top of a mountain to its base. There are several possible routes, using different slopes enroute, and passing through some flat areas. The effort expended in skiing down a slope depends upon the length of the slope and the speed of skiing. For each slope, there is a maximum advisable speed. The skier wants to use a route that minimizes the average effort spent per unit distance traveled (i.e., the total effort expended divided by the total distance traveled).

You are given the map of the mountain slopes. That is, the flat areas and the slopes connecting these areas are given. Note that on a slope, one can only ski downwards. For each slope, you are also given the length of the slope and the maximum advisable speed for it. The effort expanded in skiing down a particular slope is given by the following formula.

$$e = d \times (70 - s) \text{ if } s \leq 60, \text{ and } e = d \times (s - 50) \text{ if } s > 60$$

where e is the effort required, d is the distance traveled, and s is the speed of travel.

You have to determine the minimum average effort per unit distance that the skier has to expend in order to reach the mountain base, while staying within the maximum advisable speed at every slope.

Input

The input may have multiple test cases. The first line of input gives the number of test cases T. For each test case, the first line of input gives the number of flats, N ($N \leq 100$), and the number of slopes, R ($R \leq 10000$), connecting them respectively. The flats are assumed to be numbered from 1 to N. The flats at the top and the base of the mountain are assumed to be numbered 1 and N respectively. Each of the next R lines describes a slope by giving: the numbers of the flats at the top and the bottom of the slope, the maximum advisable speed for the slope, and the length of the slope respectively.

Output

For each test case, output a single number (to an accuracy of 2 decimal places) that gives the minimum average effort per unit distance that needs to be expended to ski down from the mountain top to the base. The output for each test case should be on a separate line.

Sample Input

```
2
4 5
1 4
1 4 30 60
1 2 50 40
1 3 60 20
2 4 60 50
3 4 50 50
3 3
1 3
1 2 50 40
1 3 40 20
2 3 20 30
```

Output for the Sample Input

```
14.44
30.00
```

Problem C
Space Ant
Input: `ant.txt`

The most exciting space discovery occurred at the end of the 20th century. In 1999, scientists traced down an ant-like creature in the planet Y1999 and called it M11. It has only one eye on the left side of its head and just three feet all on the right side of its body and suffers from three walking limitations:

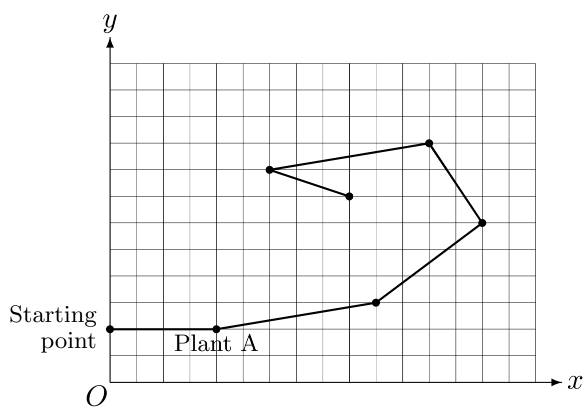
1. It can not turn right due to its special body structure.
2. It leaves a red path while walking.
3. It hates to pass over a previously red colored path, and never does that.

The pictures transmitted by the Discovery space ship depicts that plants in the Y1999 grow in special points on the planet. Analysis of several thousands of the pictures have resulted in discovering a magic coordinate system governing the grow points of the plants. In this coordinate system with x and y axes, *no two plants share the same* x *or* y.

An M11 needs to eat exactly one plant in each day to stay alive. When it eats one plant, it remains there for the rest of the day with no move. Next day, it looks for another plant to go there and eat it. If it can not reach any other plant it dies by the end of the day. Notice that it can reach a plant in any distance.

The problem is to find a path for an M11 to let it live longest.

Input is a set of (x, y) coordinates of plants. Suppose A with the coordinates (x_A, y_A) is the plant with the least y-coordinate. M11 starts from point $(0, y_A)$ heading towards plant A. Notice that the solution path should not cross itself and all of the turns should be counter-clockwise. Also note that the solution may visit more than two plants located on a same straight line.



Input

The first line of the file is M, the number of test cases to be solved ($1 \leq M \leq 10$). For each test case, the first line is N, the number of plants in that test case ($1 \leq N \leq 50$), followed by N lines for each plant data. Each plant data consists of three integers: the first number is the unique plant index $(1, \ldots, N)$, followed by two positive integers x and y representing the coordinates of the plant. Plants are sorted by the increasing order on their indices in the input file. Suppose that the values of coordinates are at most 100.

Output

Output file should have one separate line for the solution of each test case. A solution is the number of plants on the solution path, followed by the indices of visiting plants in the path in the order of their visits.

Sample Input

```
2
10
1 4 5
2 9 8
3 5 9
4 1 7
5 3 2
6 6 3
7 10 10
8 8 1
9 2 4
10 7 6
14
1 6 11
2 11 9
3 8 7
4 12 8
5 9 20
6 3 2
7 1 6
8 2 13
9 15 1
10 14 17
11 13 19
12 5 18
13 7 3
14 10 16
```

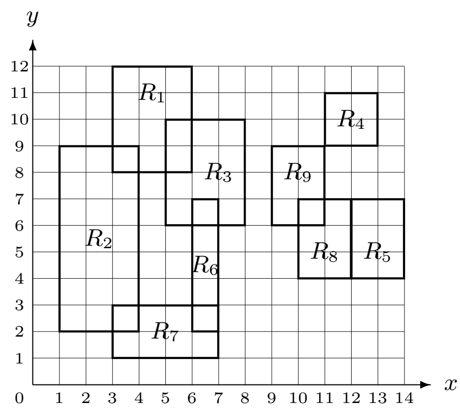
Output for the Sample Input

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

Problem D
Rectangle Coloring
Input: `rect.txt`

You are given n axis-parallel on a plane. Here, an axis-parallel rectangle is a rectangle whose edges are parallel to either x-axis or y-axis. You are to find the number of colors to paint the given n rectangles according to the following rules:

1. Each rectangle has to be painted with one color.
2. A pair of intersecting rectangles must have the same color. Two rectangles are intersecting if their intersection is not empty when we regard rectangle as a set of points including the boundary.
3. A rectangle R_a must have the same color as R_b if there is a sequence of rectangles $R_a = R_{i_1}, R_{i_2}, \ldots, R_{i_k} = R_b$ such that R_{i_j} and $R_{i_{j+1}}$ are intersecting for all $1 \le j < k$; otherwise, they must have different colors. For instance, rectangle R_9 in the following figure must have the same color as R_4, R_5, R_8, and have a different color from R_1, R_2, R_3, R_6, R_7.



Input

The input consists of T test cases. The number of test cases (T) is given in the first line of the input file. Each test case begins with a line containing an integer N, $1 \le N \le 200$, that represents the number of rectangles in the test case. Each of the following N lines contains four positive integers x_1, y_1, x_2, and y_2, $1 \le x_1, y_1, x_2, y_2 \le 10000$, representing a rectangle. (x_1, y_1) and (x_2, y_2) are the (x, y)-coordinates of the lower-left and upper-right corners of the rectangle, respectively. The four integers are delimited by one or more spaces. From the $(N + 3)$-th line, the remaining test cases are listed in the same manner as above.

Output

The ouput should contain the number of colors, one per line.

Sample Input

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

Output for the Sample Input

```
2
1
```

Problem E
Easier Done than Said?
Input: `say.txt`

Password security is a tricky thing. Users prefer simple passwords that are easy to remember (like *buddy*), but such passwords are often insecure. Some sites use random computer-generated passwords (like *xvtpzyo*), but users have a hard time remembering them and sometimes leave them written on notes stuck to their computer. One potential solution is to generate "pronounceable" passwords that are relatively secure but still easy to remember.

FnordCom is developing such a password generator. You work in the quality control department, and it's your job to test the generator and make sure that the passwords are acceptable. To be acceptable, a password must satisfy these three rules:

1. It must contain at least one vowel.
2. It cannot contain three consecutive vowels or three consecutive consonants.
3. It cannot contain two consecutive occurrences of the same letter, except for '`ee`' or '`oo`'.

(For the purposes of this problem, the vowels are '`a`', '`e`', '`i`', '`o`', and '`u`'; all other letters are consonants.) Note that these rules are not perfect; there are many common/pronounceable words that are not acceptable.

The input consists of one or more potential passwords, one per line, followed by a line containing only the word '`end`' that signals the end of the file. Each password is at least one and at most twenty letters long and consists only of lowercase letters. For each password, output whether or not it is acceptable, using the precise format shown in the sample output.

Sample Input

```
a
tv
ptoui
bontres
zoggax
wiinq
eep
houctuh
end
```

Output for the Sample Input

```
<a> is acceptable.
<tv> is not acceptable.
<ptoui> is not acceptable.
<bontres> is not acceptable.
<zoggax> is not acceptable.
<wiinq> is not acceptable.
<eep> is acceptable.
<houctuh> is acceptable.
```

Problem F
Match Maker
Input: `match.txt`

You are a manager of a match maker company ACM (Amazing Coupling Marriage) whose main role is to make happy matches between men and women.

N men and N women who are registered to the company want to marry as soon as possible. Each man and each woman have a list of preferences for all the people of the opposite sex. The most preferable person will come at the first position in the list, the second preferable person will come at the next, and so on. The table below shows a set of preference lists that might exist among 4 men and 4 women.

M_1	W_2	W_4	W_1	W_3	W_1	M_4	M_1	M_2	M_3
M_2	W_1	W_2	W_3	W_4	W_2	M_4	M_3	M_2	M_1
M_3	W_2	W_3	W_4	W_1	W_3	M_1	M_4	M_2	M_3
M_4	W_1	W_3	W_2	W_4	W_4	M_3	M_2	M_1	M_4

Your task is to make matches of all the men to all the women in such a way as to respect all their preferences as much as possible. However, you must assume that anyone assigned to someone other than their first choice will be disappointed and will always prefer anyone higher up on the list. If the N matches are chosen such that there exist a man an d a woman who are not married on each other, but who would both prefer each other to their actual marriage partners, then the matches are said to be *unstable*. If no such pair exists, it is called *stable*. For example, a match "M_1W_3 M_2W_1 M_3W_4 M_4W_2" if unstable because M_1 prefers W_1 to W_3, and W_1 prefers to M_1 to M_2. The unstable couples might be separated easily after marriage; this is a definitely bad situation that you want to avoid.

In general, there are many different stable matches for a given set of preference lists. Your task is to print just one stable match among them.

Input

The input consists of T test cases. The number of test cases (T) is given in the first line of the input file. Each test case begins with a line containing an integer N less than 100, indicating that N men and N women are given. The following N lines represent the men's preferences for the women, where the i-th line contains the preference list of a man with id i in order of preferences of the N women; he prefers a woman X to another woman Y if X precedes Y in the list. The following N lines represent the women's preferences for the N men. Assume that all men and all women have consecutive id-numbers from 1 to N.

Output

Print exactly one line for each test case. The line should contain a stable match for the test case. Each match should be represented as a sequence of the women's id, according to the increasing order of men's id. The woman with the first id in the match is a partner of the man with id '1', the woman with the second id in the match is a partner of the man with id '2', and the woman with the i-th id in the match is a partner of the man with id 'i'. The consecutive women's id in the match should be separated by a single space.

Sample Input

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

Output for the Sample Input

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

Problem G
Magazine Delivery
Input: `magazine.txt`

The TTT Taxi Service in Tehran is required to deliver some magazines to N locations in Tehran. The locations are labeled L_1 to L_N. TTT assigns 3 cars for this service. At time 0, all the 3 cars and magazines are located at L_1. There are plenty of magazines available in L_1 and the cars can take as many as they want. Copies of the magazine should be delivered to all locations, observing the following rules:

1. For all $i = 2, \ldots, N$, magazines should be delivered at L_i only after magazines are delivered at L_{i-1}.
2. At any time, only one of the three cars is driving, and the other two are resting in other locations.

The time to go from L_i to L_j (or reverse) by any car is a positive integer denoted by D[i, j]. The goal is to organize the delivery schedule for the cars such that the time by which magazines are delivered to all N locations is minimum. Write a program to compute the minimum delivery time.

Input

The input file contains M instances of this problem ($1 \leq M \leq 10$). The first line of the input file is M. The descriptions of the input data follows one after the other. Each instance starts with N in a single line ($N \leq 30$). Each line i of the following $N - 1$ lines contains D[i, j], for all $i = 1, \ldots, N - 1$, and $j = i + 1, \ldots, N$.

Output

The output file contains M lines, each corresponding the solution to one of the input data. In each line, the minimum time it takes to deliver the magazines to all N locations is written.

Sample Input

```
1
5
10 20 3 4
5 10 20
```

```
8 18
19
```

Output for the Sample Input

```
22
```

Problem H
Area Covered by Circles
Input: `circle2.txt`

You are given one or more circles with radius of one unit length. Your task is to write a program that computes the area covered by the circles. Note that the circles might contact and/or intersect each other.

Use 3.1415926 for the value of π.

Input

The input consists of several data sets.

The first line of each set contains one integer n representing the number of circles. Each of the following n lines contains exactly two real numbers x and y with four decimal digits representing a coordinate of the center of a circle. You may assume $n \leq 10$ and $1 \leq x, y \leq 9$.

The end of input is indicated by zero at the first line of a data set.

Output

For each data set, output the area with the precision of two decimal places in one line. The error must be up to 0.01.

No extra white spaces are allowed to be appeared.

Sample Input

```
2
1.5000 2.0000
1.5000 3.0000
4
1.0000 1.0000
2.0000 1.0000
2.0000 2.0000
1.0000 2.0000
0
```

Output for the Sample Input

```
5.06
7.97
```