

Problem A: Art of War

Introduction

The *Warring States Period* (473–221 BC) refers to the centuries of turmoil following the Spring and Autumn Period. China was divided into many little kingdoms that were constantly fighting with each other. Unlike in previous ages, when chivalry played an important role in battles and the states fought mostly for balance of power or to resolve disputes, in this period the aim of battle was to conquer and completely annihilate the other states. Eventually seven states, known as the “Seven Great Powers” rose to prominence: Qi, Chu, Yan, Han, Zhao, Wei, and Qin. After numerous alliances and counter-alliances, Qin defeated all the other states one by one, putting an end to the Warring States Period.

You are given a map that shows the position of the capital for each state, and the borders between the states as a series of line segments. Your job is to determine which states were fighting with each other. This is pretty easy to determine—if two states had a common border, then they were fighting.

Input

The input contains several blocks of test cases. Each case begins with a line containing two integers: the number $1 \leq n \leq 600$ of states, and the number $1 \leq m \leq 4000$ of border segments. The next n lines describe the coordinates of capitals, there are two integers in each line. The next m lines after that describe the m border segments. Each line contains four integers x_1, y_1, x_2 and y_2 , meaning that there is a border segment from (x_1, y_1) to (x_2, y_2) . (It is not given in the input what the two states on the two sides of the border are, but it can be deduced from the way the borders go.)

Each state is enclosed by a continuous borderline. The states are surrounded by an infinite wasteland, thus a border segment either separates two states, or a state from the wasteland. It is not possible that the same state is on both sides of a border segment, or the wasteland is on both sides of a border segment. There is exactly one capital in each state, and there is no capital in the wasteland. The border segments do not cross each other, they can meet only at the end points.

The input is terminated by a block with $n = m = 0$.

Output

For each test case, you have to output n lines that describe the enemies of the n states (recall that if two states share a border, then they are enemies). Each line begins with an integer, the number x of enemies the given state has. This number is followed by x numbers identifying the enemies of the state. These numbers are between 1 and n , and number 1 refers to the first capital appearing in the input, number n refers to the last.

Sample Input

```
4 12
3 2
11 8
12 17
1 19
0 0 10 0
10 0 20 0
20 0 20 10
20 10 20 20
20 20 10 20
10 20 0 20
0 20 0 10
0 10 0 0
10 0 10 10
0 10 10 10
20 10 10 10
10 20 10 10
4 16
170 13
24 88
152 49
110 130
60 60 140 60
140 60 140 140
140 140 60 140
60 140 60 60
0 0 200 0
200 0 200 200
200 200 0 200
0 200 0 0
40 40 160 40
160 40 160 160
160 160 40 160
40 160 40 40
20 20 180 20
180 20 180 180
180 180 20 180
20 180 20 20
0 0
```

Sample Output

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

Problem B: Beijing Guards

Introduction

Beijing was once surrounded by four rings of city walls: the Forbidden City Wall, the Imperial City Wall, the Inner City Wall, and finally the Outer City Wall. Most of these walls were demolished in the 50s and 60s to make way for roads. The walls were protected by guard towers, and there was a guard living in each tower. The wall can be considered to be a large ring, where every guard tower has exactly two neighbors.

The guard had to keep an eye on his section of the wall all day, so he had to stay in the tower. This is a very boring job, thus it is important to keep the guards motivated. The best way to motivate a guard is to give him lots of awards. There are several different types of awards that can be given: the Distinguished Service Award, the Nicest Uniform Award, the Master Guard Award, the Superior Eyesight Award, etc. The Central Department of City Guards determined how many awards have to be given to each of the guards. An award can be given to more than one guard. However, you have to pay attention to one thing: you should not give the same award to two neighbors, since a guard cannot be proud of his award if his neighbor already has this award. The task is to write a program that determines how many different types of awards are required to keep all the guards motivated.

Input

The input contains several blocks of test cases. Each case begins with a line containing a single integer $1 \leq n \leq 10000$, the number of guard towers. The next n lines correspond to the n guards: each line contains an integer, the number of awards the guard requires. Each guard requires at least 1, and at most 100000 awards. Guard i and $i+1$ are neighbors, they cannot receive the same award. The first guard and the last guard are also neighbors.

The input is terminated by a block with $n = 0$.

Output

For each test case, you have to output a line containing a single integer, the minimum number x of award types that allows us to motivate the guards. That is, if we have x types of awards, then we can give as many awards to each guard as he requires, and we can do it in such a way that the same type of award is not given to neighboring guards. A guard can receive only one award from each type.

Sample Input

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

Sample Output

```
8
5
3
```

Problem C: Crime

Introduction

Shanghai is one of the safest cities in China. However, the mayor of Shanghai decided to eliminate crime completely. In order to do this, the Perfect Police Patrol System (PPPS) was proposed. The city consists of intersections and streets, and every street connects two intersections. The idea is to put a PPPS department at some intersections. The officers in a department can patrol all the streets entering this intersection (and only these streets). To achieve maximum security, the departments have to be placed such that every street is patrolled, that is, there is a department at one of its two ends. Installing these departments are very expensive, hence they want to build as few of them as possible. Moreover, in the past there were unfortunate incidents where two PPPS patrols fought with each other in a dark street, because they could not recognize that they are both from the police. Therefore, we do not want to place departments at both ends of a street, that would be too dangerous. Your task is to write a program that will determine whether it is possible to design such a perfect patrol system, and if so, then what is the minimum number of departments that has to be established.

Input

The input contains several blocks of test cases. Each case begins with a line containing two integers: the number $1 \leq n \leq 1000$ of intersections, and the number $1 \leq m \leq 20000$ of streets. The next m lines describe the m streets. Each of these lines contain two positive integers that identify the two intersections at the two ends of the street (the intersections are numbered $1, 2, \dots, n$). Because of the various tunnels, bridges, temporarily closed streets, etc., you cannot assume anything about the topology of the streets.

The input is terminated by a block with $n = m = 0$.

Output

For each test case, you have to output a line containing a single integer, the minimum number of departments that have to be installed. If it is not possible to satisfy the requirements, then write 'Impossible' (without quotes).

Sample Input

```
7 5
1 2
1 3
5 6
6 7
1 2
5 6
1 2
1 3
2 4
3 4
3 5
4 5
0 0
```

Sample Output

```
2
Impossible
```

Problem D: Desert

Introduction

The Gobi Desert is the second largest desert in the world. It lies north of Huang He (Yellow River) on the border of China and Mongolia, extending some 500 km into both countries. Unlike the Sahara, only 5% of the Gobi Desert is covered by sand dunes, thus it is capable of supporting a wide variety of wild life, such as wild sheep, wild ass, and the Gobi bear. The desert is also home to nomadic tribes.

The oases in the Gobi Desert are often rich and fertile. However, life is a bit boring there, and you may have to walk hundreds of kilometers to find a movie or theater. To bring some entertainment to the people, we will install three television stations in the desert. According to the standards of the Gobi Desert Broadcasting Company, we have the following requirements:

- A station can be installed only in an oasis.
- Two stations cannot be in the same oasis.
- A station broadcasts in a given direction, and an oasis receives the program only if its direction differs by at most 45 degrees from the direction of the station. Example: a station at coordinates (10, 10) can broadcast to the oases at (10, 11) and (11, 10) at the same time. It can also broadcast to the oases at (10, 11) and (9, 11) at the same time. However, it cannot broadcast to all three oases at (10, 11), (11, 10), and (9, 11) at the same time.
- We have to install three stations such that all three of them can be received in *every* oasis.
- We assume that if a station is installed in an oasis, then this oasis can receive the program regardless of the direction of the station.

Input

The input contains several blocks of test cases. Each case begins with a line containing an integer $1 \leq n \leq 20000$, the number of oases. The next n lines contain three integers each, the coordinates of the oases and the cost of installing a station there. The coordinates are between -25000 and 25000 , and the cost is between 1 and 10000000.

The input is terminated by a block with $n = 0$.

Output

For each test case, you have to output a separate line containing three integers: the oases where the stations are installed. The oases are numbered from 1 to n . The three numbers should be printed in increasing order. If there are multiple solutions, then select the solution where the cost of installing the three stations is minimal (there will be at most one minimal solution). If it is not possible to place the stations such that all the requirements are satisfied, then write 'Impossible' (without quotes).

Sample Input

```
8
0 5 1
5 0 2
10 5 3
5 10 4
0 0 5
10 10 6
0 10 7
10 0 8
4
0 0 1
1 1 10
2 0 2
1 5 20
0
```

Sample Output

```
5 6 7
1 3 4
```

Problem E: Explore Tibet

Introduction

After having to reinstall your computer 42 times on the same day, you decided to take a short break—you will spend the next 5 years (or more) in Tibet. The population of China is very unevenly distributed: in the eastern coastal areas the population density can be above 400 people per square kilometer, while in the western plateaus there are less than 10 people per square kilometer. But no part of China is more sparsely populated than Tibet, where 2.3 million people share 1.2 million square kilometers. You hope that in the small villages and monasteries of Tibet, no one will ask you to fix their computer.

But which village should you choose? Looking at the map of Tibet, you see a large number of interesting places. You want to go to a place where you can visit many monasteries. Each village has a number of monasteries. Your plan is that you go from village to village to visit as many monasteries as possible. However, you can only travel 30 kilometers a day, and it is not safe to spend the night in the wilderness. Thus, depending on your initial position, you can visit only some of the villages. Therefore, you have to choose your initial position (the village where you start your holiday) carefully, if you want to maximize the number of monasteries that can be visited.

A final note: Tibet enjoys an average of 3,000 hours of sunshine a year, so don't forget your sun glasses and suntan cream!

Input

The input contains several blocks of test cases. Each case begins with a line containing an integer $1 \leq n \leq 1000$, the number of villages. The next n line contains three integers each: the two coordinates of the village (in kilometers), and the number of monasteries in the village. The coordinates are between 0 and 20000, the number of monasteries in a village is at most 1000. The input is terminated by a block with 0 villages.

Output

For each test case you have to output two integers on a line (separated by spaces). The first integer identifies the village where you want to go (this number is between 1 and n), and the second integer is the number of monasteries that can be visited starting from this location. If there is more than one village that maximizes the number of reachable monasteries, then choose the one that has the smallest index.

Sample Input

```
6
100 100 8
0 0 10
0 10 3
10 30 4
1000 1000 4
100 128 3
0
```

Sample Output

```
2 17
```

Problem F: Fixing the Great Wall

Introduction

The Great Wall of China is truly one of the greatest wonders of the world. In 3rd century BC, Emperor Qin Shi Huang connected the defensive structures built earlier by the states of Qin, Yan, and Zhao kingdoms. The purpose of the wall was to defend against raids by the barbarians from Mongolia and Manchuria. The wall was extended and renovated in later centuries, creating an impressive 6,700 km long fortification.

The centuries have left their mark on the wall, there are several sections that need immediate renovation. These sections have to be repaired as soon as possible since they deteriorate every day: if we do not fix them now, it will be more expensive to repair them later. Thus the Ministry of Monuments have designed and built the world's first Great Wall Automatic Repair Robot (GWARR), to repair the damaged sections (we are in the 21st century, aren't we?) Your task is to write the software that will guide the robot and decide the order in which the sections are to be repaired.

For the purpose of this problem, we assume that the Great Wall is a long straight line, and every location on the wall is identified by a single number (say, the distance from one end). The GWARR is placed at some location on the wall and it can move with constant speed in both directions. For each damaged section you are given its location, how much it would cost to repair now, and how the cost would increase if repaired later. The GWARR works so efficiently that once it is at the exact location of the damaged section it can repair the wall immediately.

Input

The input contains several blocks of test cases. Each case begins with a line containing three integers: an integer $1 \leq n \leq 1000$, the number of damaged sections, an integer $1 \leq v \leq 100$, the speed of the GWARR in distance units/time units, and an integer $1 \leq x \leq 500000$, the initial position of the GWARR. The next n lines describe the n damaged sections that have to be repaired. Each line contains three integers: the location $1 \leq x \leq 500000$ of the section, the cost $0 \leq c \leq 50000$ of repairing it immediately, and $1 \leq \Delta \leq 50000$, the increase in cost per time unit. Therefore, if the section is repaired after t time units have passed, then we have to pay $c + t\Delta$ units of money. It can be assumed that the locations of the sections are all different, and the initial location of the robot is not on the list of damaged sections.

The input is terminated by a test case with $n = v = x = 0$.

Output

For each test case, you have to output a line containing a single number, the minimum cost of repairing the wall. This number should be an integer, round *down* the result, if necessary. It can be assumed that the minimum cost is not more than 1000000000.

In the optimum solution for the first test case below, we first fix location 998 at the cost of 600, then the location 1010 at the cost of 1400, and finally we fix the location 996 at the cost of 84, giving the total cost 2084.

Sample Input

```
3 1 1000
1010 0 100
998 0 300
996 0 3
3 1 1000
1010 0 100
998 0 3
996 0 3
0 0 0
```

Sample Output

```
2084
1138
```

Problem G: Gambling

Introduction

Gambling has always been very popular in China. Although it was prohibited for most of the time, people nevertheless played Mah Jong, Pai Gow, Fan-Tan, Sic Bo, and other games in secret. In the 1930s, Shanghai was home to many illegal gambling dens, controlled by powerful gangs. Most of them were closed in 1949 by the Communists, making Shanghai a safer place.

In this problem we consider a lesser-known game called Ah Ce Emm. In this game you receive a random amount of pebbles. Your goal is to lose all these pebbles, if you can, then you get a prize. You have several options for modifying the number of pebbles, but you have to pay a certain amount of money for each move:

- **The fire:** if you have at least 11 pebbles, then you can throw away exactly 11 pebbles by paying x_1 .
- **The dragon:** if the number of your pebbles is divisible by 3, then you can throw away exactly one third of your pebbles by paying 1 for each pebble thrown away. Thus if you have 12 pebbles, then with this move you can reduce the number of pebbles to 8 by paying 4.
- **The eagle:** you can ask for exactly 7 new pebbles by paying x_2 .
- **The courage:** you can double the number of your pebbles and get one additional pebble by paying 1 for each new pebble you get. Thus if you have 3 pebbles, then this move increases the number of pebbles to 7, at the cost of 4.

The amounts x_1 and x_2 vary from game to game. You are not allowed to have more pebbles than initially: if a move would increase the number of pebbles above the original number, then you cannot choose this move. Your task is to write a program that, given the number of pebbles and the values x_1 , x_2 , determines the minimum cost of losing all the pebbles.

Input

Each line of the input contains 3 integers, and corresponds to a different test case. The first number $1 \leq n \leq 200000$ is the initial number of pebbles. The second and third numbers $1 \leq x_1, x_2 \leq 500000$ are the cost of the fire and the eagle.

The input is terminated by a line containing three zeros.

Output

For each test case you have to output an integer on a separate line, the minimum cost of losing all the pebbles. If there is no way of reducing the number of pebbles to zero, then write 'Impossible' (without quotes).

Sample Input

```
33 122 200
1000 100 200
2 10 10
0 0 0
```

Sample Output

```
255
1953
Impossible
```

Problem H: History

Introduction

Two world-famous professors of archaeology, Professor C. H. Eater and Professor L. Iar, decided to write a book together on the ancient history of China. They figured that by spending five years in China, they could research a sufficient amount of scientific data to write a stunning new book. They divided China into two parts: Professor C. H. Eater collected data in the eastern part of the country, while Professor L. Iar was doing research in western China. After five years, they have met with all their collected evidence. Both professors have written some chapters, these chapters have to be combined into a single book. However, they have realized with horror that some of their chapters contain contradictory data. For example, they have obtained different dates for some important events (famous battles, the deaths of some Emperors, etc.) They don't think that it will be possible to write a best-seller with contradictory data, and they really don't want to spend another five years sorting these things out. Thus they plan to get rid of some of the annoying data (while no one is looking). They ask you to determine the minimum number of chapters that will have to be thrown away to make the remaining material non-conflicting.

Both professors have written a set of *chapters*. Each chapter gives dates for certain *events*. Two chapters are *conflicting* if there is an event when the two chapters give different dates for this same event. Thus if a chapter says that event A was in 234, event B was in 291, and event C was in 262; and another chapter says that event A was in 234, event C was in 293, and event D was in 218, then these two chapters are conflicting. It is not possible to throw away only a single event from a chapter: either you throw away the whole chapter, or you keep it with all the events and dates. It can be assumed that two chapters collected by the same professor do not contradict (presumably, they have taken care of these conflicts earlier).

Input

The input contains several blocks of test cases. Each case begins with a line containing two integers n_1 , n_2 , where n_1 is the number of chapters written by Professor C. H. Eater, and n_2 is the number of chapters written by Professor L. Iar. Both numbers are at most 2000. The next n_1 lines describe the n_1 chapters of Professor C. H. Eater. Each line begins with an integer $1 \leq m \leq 1000$, the number of events whose dates are mentioned in this chapter. This number is followed by m pairs of integers. The first integer of each pair identifies the event (for simplicity, we assume that the professors assigned a unique code to each event), the second integer is the date of the event. The code is at most 1000000, the date is between -10000 and 1000 (we are talking about ancient history here!) These n_1 lines are followed by n_2 lines that describe the chapters written by Professor L. Iar, which are in the same format.

The input is terminated by a block with $n_1 = n_2 = 0$.

Output

For each test case in the input, you have to output a single integer on a separate line: the minimum number of chapters that have to be deleted to make the book non-conflicting.

Sample Input

```
3 3
3 10 999 20 888 77 100
2 30 977 88 -1
2 77 100 40 855
1 10 988
3 88 -1 20 887 77 100
2 30 955 40 -10
0 0
```

Sample Output

```
2
```

Problem I: I can't read it!

Introduction

The Chinese writing system was developed about 4000 years ago, and it has changed relatively little since then. It consists of more than 40 thousand characters, where each character means a word or an idea. Spoken Chinese differs very much from region to region, but the written language is mostly the same. Thus writing acts as a unifying common language among the different regions.

Since over one billion people speak Chinese, it is of great practical importance to have a program that is capable of translating Chinese text to English. We have developed such a program, but it is not working perfectly. The problem is that Chinese can be written in two different ways: either the characters are written from left to right, or they are written vertically, with the columns going from right to left. The translator program is confused by these two possibilities: the translated English text is reversed. More precisely, the words follow each other in a left to right order, as they should, but the letters in a word go from right to left. Thus instead of the text 'one two three', the program produces 'eno owt eerht'. Your job is to write a program that takes this reversed text, and outputs the correct translation.

Input

The input contains several blocks of test cases. Each case begins with a line containing an integer $1 \leq n \leq 100000$, the number of lines in this test case. This is followed by n lines of text. The length of each line is at most 1000 characters. The only characters appearing on these lines are the letters 'a'-'z' and the space character.

The input is terminated by a block with $n = 0$.

Output

For each test case, you have to output the n lines, with the words reversed. A word is a sequence of characters delimited by spaces, by the start of the line or by the end of the line. The spaces and new line characters have to be printed exactly as they appeared in the input.

Sample Input

```
1
eno owt eerht
2
 abcde
aabb  cac x ab
0
```

Sample Output

```
one two three
edcba
bbaa  cac x ba
```