Problem A

Gnome Tetravex

Input file: tetravex.in

Hart is engaged in playing an interesting game, Gnome Tetravex, these days. In the game, at the beginning, the player is given n*n squares. Each square is divided into four triangles marked four numbers (range from 0 to 9). In a square, the triangles are the left triangle, the top triangle, the right triangle and the bottom triangle. For example, Fig. 1 shows the initial state of 2*2 squares.

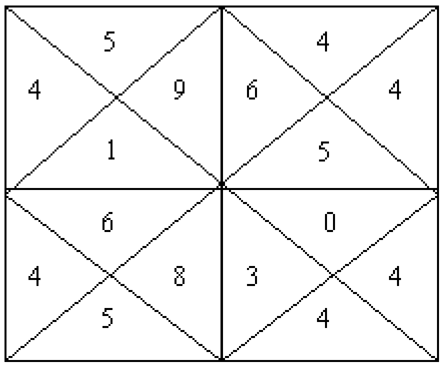


Fig. 1 The initial state with 2*2 squares

The player is required to move the squares to the termination state. In the termination state, any two adjoining squares should make the adjacent triangle marked with the same number. Fig. 2 shows one of the termination states of the above example.

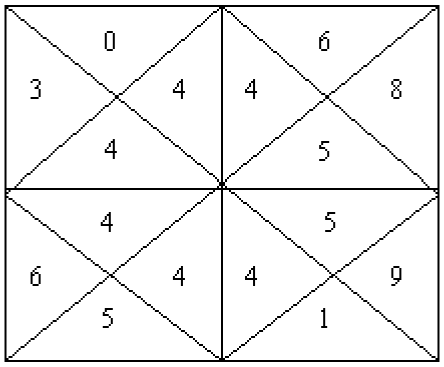


Fig. 2 One termination state of the above example

It seems the game is not so hard. But indeed, Hart is not accomplished in the game. He can finish the easiest game successfully. When facing with a more complex game, he can find no way out.

One day, when Hart was playing a very complex game, he cried out, “The computer is making a goose of me. It's impossible to solve it." To such a poor player, the best way to help him is to tell him whether the game could be solved. If he is told the game is unsolvable, he needn't waste so much time on it.

Input

The input file consists of several game cases. The first line of each game case contains one integer n, $0 \leqslant n \leqslant 5$, indicating the size of the game.

The following $n*n$ lines describe the marking number of these triangles. Each line consists of four integers, which in order represent the top triangle, the right triangle, the bottom triangle and the left triangle of one square.

After the last game case, the integer 0 indicates the termination of the input data set.

Output

You should make the decision whether the game case could be solved. For each game case, print the game number, a colon, and a white space, then display your judgment. If the game is solvable, print the string "Possible". Otherwise, please print "Impossible" to indicate that there's no way to solve the problem.

Print a blank line between each game case.

Note: Any unwanted blank lines or white spaces are unacceptable.

Sample Input

```
2
5 9 1 4
4 4 5 6
6 8 5 4
0 4 4 3
2
1 1 1 1
2 2 2 2
3 3 3 3
4 4 4 4
0
```

Output for the Sample Input

```
Game 1: Possible

Game 2: Impossible
```

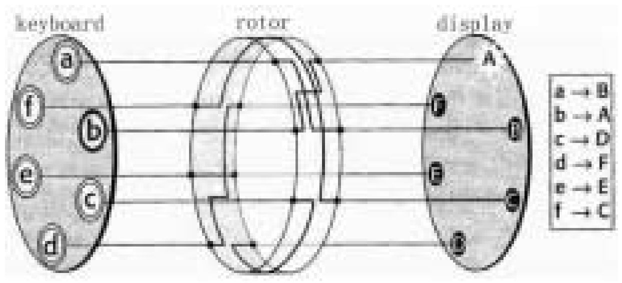
Problem B

Enigma

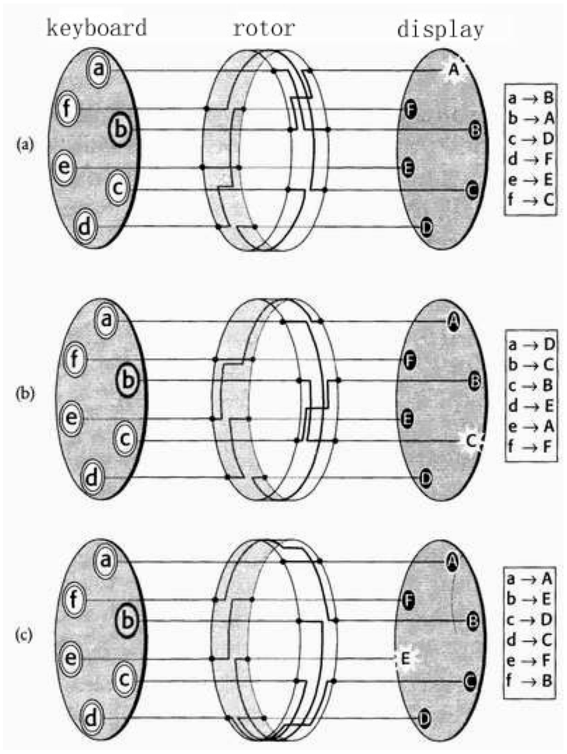
Input file: enigma.in

In World War II, Germany once used an electronic encryption machine called Enigma, which played a decisive role in the initial victories of Nazi Germany. It was proved to be one of the most reliable encryption systems in history. However, it was the blind trust on the reliability of the machine that brought about the doom of its user.

The structure of a one-rotor Enigma is shown as follows (the Enigma has only six keys):

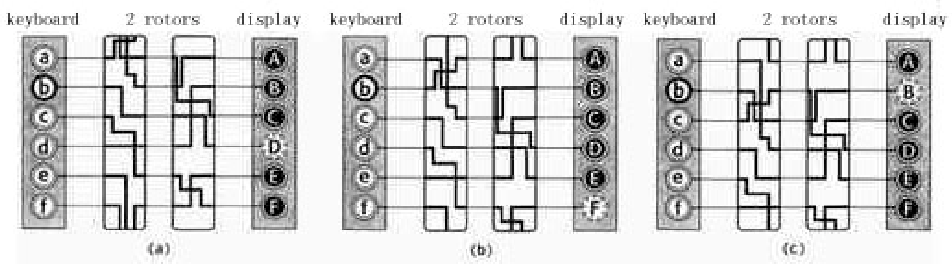


The key element of the Enigma is the rotor, as shown in the second figure, which uses electronic circuits to transform plaintext (input from keyboard) into cryptograph (output on screen). When one key on the keyboard is pressed, the corresponding cryptograph is shown on screen. Then the rotor will automatically revolve a one-letter-step to a different position. The following figures illustrate how the rotor works when letter “b” is pressed three successively times:



When letter “b” is pressed for the first time, the signal goes through the circuit and “A” is shown on screen. When the key is released, the rotor revolves one-letter-step to a different position that changes all the corresponding circuits so that each letter now has a different cryptograph. When letter “b” is pressed for the second time, the corresponding cryptograph is “C”. So when letter “b” is pressed for the third time, the cryptograph is “E” according to the principle specified above.

Now the following figure shows the structure of a two-rotor Enigma.



The difference is that when a key is released, the second rotor won’t revolve a step until the first one has finished one circle and returns to the original position. This is also the same in the case of three-rotor Enigma. That is: Only after the first rotor has finished one circle and return to the initial status, the second rotor will revolve a step. And only after the second rotor has finish one circle, the third rotor will revolve a step.

However, how did the Allied Forces obtain the information encrypted by Enigma? A person named Hans-Thilo Schimdt was very essential. He acted as a spy and provided the initial status of the three rotors in each Enigma to the Allied Forces once a month. The Allied Forces thus got everything they wanted by deciphering the intercepted cryptograph using the information offered by the spy.

Now, please design a program to obtain the plaintexts using the information offered by the Allied Forces.

Input

The input file contains several test cases representing several three-rotor Enigmas. The last test case in the input file is followed by a line containing a number 0.

Each case begins with a line containing an integer m ($1 \leqslant m \leqslant 26$) which indicates the number of sequential letters each rotor has. The first letter will always be A. (for example, $m = 6$ tells each rotor has 6 keys from A to F). The following three lines describe the initial status of the three rotors respectively. Each of them contains a string consisting of m capital character. For instance, a rotor with the initial status “BADFEC” indicates that the initial encrypt mechanism is to convert “abcdef” to “BADFEC”, that is, original letter “a” corresponding to cryptograph letter “B”, “b” to “A”, “c” to “D”, “d” to “F”, “e” to “E” and “f” to “C”. The forth line of each case contains an integer n which tells the number of cryptographs generated by the above Enigma. Then the following n lines are the n cryptographs respectively, which consist of m capital characters each.

Output

For each test case, the output should consist of two parts. The first line is the number of

Enigma and a colon. The following lines are the plaintexts deciphered from the corresponding cryptographs. Each plaintext should be printed in one line. Note: The characters in the plaintext should be converted to the corresponding lowercases before they are printed.

Insert a blank line between test cases.

Sample Input

```
6
BADFEC
ABCDEF
ABCDEF
1
ACE
0
```

Output for the Sample Input

```
Enigma 1:
bbb
```

Problem C

Area

Input file: area.in

Jerry, a middle school student, addicts himself to mathematical research. Maybe the problems he has thought are really too easy to an expert. But as an amateur, especially as a 15-year-old boy, he had done very well. He is so rolling in thinking the mathematical problem that he is easily to try to solve every problem he met in a mathematical way. One day, he found a piece of paper on the desk. His younger sister, Mary, a four-year-old girl, had drawn some lines. But those lines formed a special kind of concave polygon by accident as Fig. 1 shows.

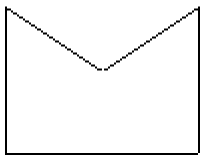


Fig. 1 The lines his sister had drawn

"Great!" he thought, "The polygon seems so regular. I had just learned how to calculate the area of triangle, rectangle and circle. I'm sure I can find out how to calculate the area of this figure." And so he did. First of all, he marked the vertexes in the polygon with their coordinates as Fig. 2 shows. And then he found the result--0.75 effortless.

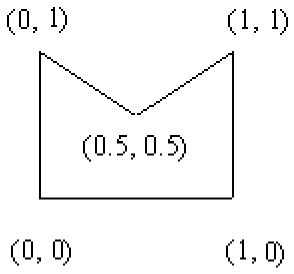


Fig.2 The polygon with the coordinates of vertexes

Of course, he was not satisfied with the solution of such an easy problem. "Mmm, if there's a random polygon on the paper, then how can I calculate the area?" he asked himself. Till then, he hadn't found out the general rules on calculating the area of a random polygon. He clearly knew that the answer to this question is out of his competence. So he asked you, an erudite expert, to offer him help. The kind behavior would be highly appreciated by him.

Input

The input data consists of several figures. The first line of the input for each figure contains a single integer *n*, the number of vertexes in the figure. ($0 \leqslant n \leqslant 1000$).

In the following *n* lines, each contain a pair of real numbers, which describes the coordinates of the vertexes, (x_i, y_i). The figure in each test case starts from the first vertex to the second one, then from the second to the third, …… and so on. At last, it closes from the *n*th vertex to the first one.

The input ends with an empty figure (n = 0). And this figure not be processed.

Output

As shown below, the output of each figure should contain the figure number and a colon followed by the area of the figure or the string “Impossible”.

If the figure is a polygon, compute its area (accurate to two fractional digits). According to the input vertexes, if they cannot form a polygon (that is, one line intersects with another which shouldn’t be adjoined with it, for example, in a figure with four lines, the first line intersects with the third one), just display “Impossible”, indicating the figure can’t be a polygon. If the amount of the vertexes is not enough to form a closed polygon, the output message should be “Impossible” either.

Print a blank line between each test cases.

Sample Input

```
5
0 0
0 1
0.5 0.5
1 1
1 0
4
0 0
0 1
1 0
1 1
0
```

Output for the Sample Input

```
Figure 1: 0.75

Figure 2: Impossible
```

Problem D

NTA

Input file: nta.in

The NTA (Non-deterministic Tree Automata) is a kind of tree structure device. The device is built in a set of operating rules. With these rules the device can produce several signals, which will form a signal system. In such a system, one signal is the starting signal, several signals are the acceptable signals, and the others are the auxiliary ones. A pair of signals is said to be an acceptable pair if both two signals of the pair are acceptable.

The trees discussed here are all binary trees. Every non-leaf node has two successors. In any finite tree, each node has a signal-transmitting element. When a signal arrives at one node, the signal meets the signal transmitting substance, and triggers off signal reactions, which will produce several pairs of signals. Then the device selects a pair of signals non-deterministically and sends them to its successors. The first signal in the signal pair is sent to the left successive node and the second one is sent to the right successive node.

The whole operation for an NTA is as follows:

The device first sends the starting signal to the root node. According to the signal transmitting substance at the root node, the device selects a pair of signals non-deterministically and sends the first to the left son and the second to the right son. Each of the two signals then meets the signal transmitting substance at the corresponding node and produces another two signals. The course proceeds down until the signals arrive at the leaves.

If a signal reaches one leaf and the leaf can produce a pair of acceptable signals, we say the leaf is “shakable”. A transmission of signals from the root to leaves is said to be valid if all leaves are “shakable”. A tree structure with signal transmitting substance is valid if there exists such a valid transmission. A tree is invalid if all the transmissions are invalid.

For simplicity, we denote the signal transmitting elements by consecutive lowercase letters “a”, “b”, “c”, etc.. The signals of an NTA are consecutive numbers 0,1,2, ..., and so on. The first signal 0 is always a starting signal. Thus the signals for a 4-signal NTA are “0” “1” “2” and “3”. Accepting signals are arranged at the end of the number sequence so that if a 4-signal NTA has two accepting signals, the accepting signals are “2” and “3”. The transition rules of signals are based on a transition table. For example, the following table describes a transition table with four signals “0”, “1”, “2”, “3” and with three signal transmitting elements “a”, “b” and “c”.

T	a	b	c
0	(1,2)	(2,1)	(1,0)
1	(2,2)	(0,2)、(1,0)	(3,2)
2	(2,2)	(2,3)	(1,2)
3	(1,2)	(2,1)	(3,2)

In this transition table some reactions of signals on certain signal transmitting elements are deterministic, and others are non-deterministic. In the example above, if signal “1” reaches the node with the transmitting element “b”, the reaction is non-deterministic.

Now your task is to write a program to judge if a tree structure with certain signal transmitting substance is valid.

Input

The input file contains several cases. Each case describes a sequence of NTA descriptions and some initial tree configurations. The first line for each case consists of three integers *n*, *m* and *k*. The integer *n* is the number of signals, *m* indicates the number of accepting signals, and *k* is number of signal transmitting elements. The following $n \times k$ lines describe the transition table in row-major order. Each transition of a signal on signal transmitting element is given on a separate line. On such line every two numbers represent a possible transition.

This is followed by the description of tree structures. For every tree structure a number *L* is given on a separate line to indicate the level of the tree. The following *L*+1 lines containing a sequence of letters describe the tree structure. Each level is described in one line. There exist one space between two successive letters. The 0-th level begins firstly. In the tree structure, the empty nodes are marked by “*”. The tree structure with *L*=-1 terminates the configurations of tree structures for that NTA, and this structure should not be judged.

The input is terminated by a description starting with *n*=0, *m*=0 and *k*=0. This description should not be processed.

Note: In each case, NTA will have at most l5 signals and 10 characters. The level of each tree will be no more than 10.

Output

For each NTA description, print the number of the NTA (NTAl, NTA2, etc.) followed by a colon. Then for each initial tree configuration of the NTA print the word “Valid” or “Invalid”.

Print a blank line between NTA cases.

Sample Input

```
4 2 3
1 2
2 1
1 0
2 2
0 2 1 0
3 2
2 2
2 3
1 2
1 2
2 1
3 2
3
a
b c
a b c b
b a b a c a * *
2
b
a b
```

```
b c * *
-1
0 0 0
```

Output for the Sample Input

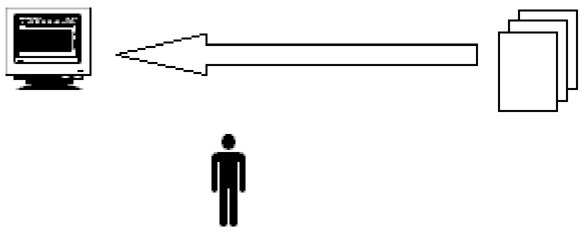
```
NTA1:
Valid
Invalid
```

Problem E

Mainframe

Input file: frame.in

Mr. Ronald is responsible for the administration of the mainframe in ACM (Agent on Computing of Mathematics). The agent undertakes the mathematical computing jobs from some companies, and gain the rewards after has fulfilled the jobs on the mainframe. So the mainframe is very valuable to ACM. Mr. Ronald is required to arrange the order of those jobs running on mainframe. Once a job is to run, he examines the free resources for the job. If the resources meet the job's requirement, he assigns those resources to the job. Otherwise, the job will be suspended until there are enough resources.



Because of unfamiliar with the task at first, he turned everything upside down. As time went by, he became competent on it. Moreover, he had concluded a set of byelaw as following:

1. The mainframe has M CPUs and N memories can be assigned.

2. There exists a queue for the jobs waiting to be executed. You may assume the queue is large enough to hold all the waiting jobs.

3. A job J_i which need A_i CPUs and B_i memories, reaches the queue on time T_i. The job is required to be accomplished before time U_i. After successfully completed, ACM may get V_i($) as the reward. If it finishes before the timeline, the extra bonus is W_i($) per hour. If the job is late, the punishment is X_i($) per hour. For example, we may assume that a job's value is 10$, its timeline is 8, and the punishment is 2$ per hour. If the job is completed at time 10, ACM will get 10-(10-8)*2=6$.

4. When the job start executing, the required CPUs and memories are seized by this job, and couldn't be assigned again for the other job to be executed simultaneously. After completing the job, those resources will be released. If the resources are enough, more jobs could be executed simultaneously.

5. For the sake of the share in the mainframe's computing capability, each job will be finished just in an hour from the start of executing. You may assume each job costs exactly one hour.

6. When there are no jobs to be executed, the mainframe will be idle until a job arrives at the job queue.

7. If there are more than one jobs arrive at the queue, the more valuable job will be executed first. You may assume the values of the jobs are always unequal ($V_i \neq V_j$).

8. If the free CPUs or memories couldn't satisfy the requirement of the job, the job will be suspended for an hour without occupying any resources. An hour later, the resources will be examined again for this job, regardless the other jobs in the queue. If the requirement unsatisfied again, it remains suspended for the next hour, and other jobs in the queue will try to be assigned the resources. Otherwise the job will seize the required CPUs and memories and start executing.

9. When more than one jobs are suspended, the earlier arrived will try to be assigned first.

Using the byelaw, Mr. Ronald may deal with the routines very well. But now, besides the routines, ACM ask him to compute the income according to the job list. Given the timeline F, he has to calculate the jobs that had been executed or should be executed. Of course, according to job J_i, if $U_i>F$ and the job hadn't been executed, it shouldn't been taken into account; but those which had been executed or $U_i<=F$ should been counted. If the job hadn't been executed, it will not bring ACM any value, which means only punishment to the timeline should be calculated

Indeed, his programming ability is not good enough, and he does not like to compute manually. So he is uneasy about it. Could you help him to solve this problem?

Input

The input contains several test cases, each of which describes the mainframe's resources and the job list. Each test case begins with a line containing a single integer F, $0\leqslant F\leqslant 10000$, the time line. The following line consists of three integers M, N and L (M, N, $L\geqslant 0$). M is the number of CPU in the mainframe, and N is the memory size. L represents the number of jobs in the job list. There will be 10000 jobs at most.

The subsequent L lines in the test case describe the information of the jobs. The data which describing job J_i consist of 7 integers A_i, B_i, T_i, U_i, V_i, W_i, X_i. A_i and B_i indicate the requirements on CPU and memory (A_i, $B_i\geqslant 0$). T_i and U_i indicate the job's arriving time and the timeline ($0\leqslant T_i<U_i$). V_i, W_i, X_i are the reward, bonus and punishment of the job (V_i, W_i, $X_i\geqslant 0$).

The input file ends with an empty test case (F=0). And this case should not be processed.

Output

Your program must compute the total income of the mainframe according to the job list. For each test case, print the case number, a colon, and a white space, then the income.

Print a blank line after each test case.

Note: Don't count the jobs which hadn't been executed, and their timelines are later than F.

Sample Input

```
10
4 256 3
1 16 2 3 10 5 6
2 128 2 4 30 10 5
2 128 2 4 20 10 5
0
```

Output for the Sample Input

```
Case 1: 74
```

Problem F

Great Equipment

Input file: equip.in

Once upon a time, there lived Catherine Ironfist, the Queen of Enroth. One day, she received the news of her father's death. So she sailed for Erathia to attend her father's funeral. Fearing the worst, she assembled a military fleet as her escort. On reaching the coast of Erathia, Catherine found an allied wizard's tower, devastated from battle and abandoned. There she learned that a black-hearted knight poisoned her father using a goblet of wine, and Erathia was falling to the enemies. And then, she mustered local armies, and marched to Erathia's castle, restoring lost land along the way.

During the battles, she found that the equipments for the soldiers were in urgent need. And she knew clearly that the best equipments were made by the battle dwarf's workshop in the world. The workshop's equipments were well known for the firmness and top-quality. "Cloak of the Undead King", "Armor of the Damned", "Angelic Helm" are the nonesuch ones. But unfortunately, the workshop was seated at the Erathia's castle, the territory under the enemy's control. So she sent a brave volunteer to come into the castle and asked for the workshop's help.

"It's our pleasure to help the righteous heroine." Rion, the leader of the workshop sent the message to Catherine, " We haven't enough resources to build the nonesuch equipments. So we'll try to offer the ordinary equipments as more as possible. Still, those ones are much better the equipments made by other workshops. But we have faced a difficult problem. The castle is in a state of siege. The guards prohibited the equipments to be carried away from the castle. We have to ask for the trade caravans' help. As you know, each trade caravan's capability of carrying equipments is limited. If they had carried a little more, they would be detected by the guards, which would lead them into the prison. So you have to arrange how to carry these equipments."

The workshop would offer helms, armors and boots. These three ones had different defend capabilities. Also, their weight and size were different from each other. What's more, Rion had told Catherine that if armed with those three ones together, they would form a set of equipments, which might provide much more defend capability. As far as the trade caravan was concerned, each one had its special weight limitation and size limitation. Catherine had to make the decision on how to arrange the transportation plan to provide her soldiers as more defend capabilities as possible. Could you help her to finish the plan?

Input

The input describes several test cases. The first line of input for each test case contains a single integer n, the number of trade caravans ($0 \leqslant n \leqslant 100$).

The following four lines describe the information of those equipments. The first line contains three integers w_1, s_1 and d_1, indicating the weight, size and defend capabilities of the helm. The integers w_2, s_2 and d_2 in the second line represent the weight, size and defend capabilities of the armor. Also, in the third line, w_3, s_3 and d_3 are the weight, size and defend capabilities of the boot.

The fourth line contains four integers c_1, c_2, c_3 and d_4. Among those integers, c_1, c_2, c_3 are the number of helms, armors and boots in a set of equipments, d_4 is the capability of this set.

In the test case, following those data are n lines, describing the carrying capabilities of the trade caravans. Each line contains two integers, x_i and y_i, indicating the weight limit and size limit of a trade caravan.

The input is terminated by a description starting with n = 0. This description should not be processed.

Note: Because of the trade caravans' carrying capabilities, you may assume the quantities of the helms, armors and boots will not exceed 500 respectively.

Output

Your program must compute the defend capability of the best carrying plan. That is, after having performed the carrying plan, the defend capability of the equipments which have been carried away from the castle should be the largest. For each test case in the input file, print the case number and a colon, and then the defend capability of those equipments.

Print a blank line between test cases.

Sample Input

```
3
1 1 3
5 6 10
2 1 2
1 1 1 50
1 1
5 6
2 1
0
```

Output for the Sample Input

```
Case 1: 50
```

Problem G

Operand

Input file: operand.in

Professor Maple teaches mathematics in a university. He have invented a function for the purpose of obtaining the operands from an expression. The function named op(i,e) can be described as follows: The expression e may be divided into sub-expression(s) by the operator, which has the lowest priority in the expression. For example, the expression “a*b+b*c+c*d” should be divided into three sub-expressions “a*b”, “b*c” and “c*d”, because the operator “+” has the lowest priority. The purpose of this function is to extract the i^{th} sub-expression as the result. So, in the example above, op(2,e)=b*c.

If we regard the sub-expression as the main expression, it might be divided again and again. Obviously, the dividing process is recursive. As you see, the following example is much more complex:

```
Let p:=a^b*c+(d*c)^f*z+b
op(1,op(1,op(2,p)))=(d*c)
op(1,op(1,op(1,op(2,p))))=d*c
op(2,op(2,p))=z
op(3,p)=b
op(1,op(3,p))=b
```

Professor Maple is so lazy that he would leave the work to computer rather than do it himself, when the expression is long and complicated. Of course, without your program, the computer won’t work out the result automatically.

Input

The input file contains several test cases. The last test case in the input file is followed by a line containing a symbol “*”, indicating the end of the input data. Each test case consists of two parts. The first part describes the expression, while the second part contains several questions, which should be calculated according to the expression.

The first line of each test case contains an expression consists of the expression name, “**:=**” and the content of the expression. The expression name is a lowercase. And the content is composed by lowercases and operators “+”, “(”, “)”, “*” and “^”. For example, here is a valid expression, p:=a^b*c+(d*c)^f*z+b. Among those operators, “(” and “)” have the highest priority. The operator “^” has a lower priority, and then “*”. The priority of the operator “+” is the lowest.

The second line of each test case contains an integer *n* indicating *n* questions based on the above expression. This is followed by *n* lines. Each of them contains the description of one question, which consists of integers. For example, the question with three integers “2 1 1” describes the function op(1,op(1,op(2,e))). To compute this function, we have to keep to the following sequence: First, according to the first integer 2, divide the expression and extract the 2nd sub-expression. Then, according to the second integer 1, divide the sub-expression and extract the 1st one. Finally, according to the third integer 1, divide the outcome again, and extract the result.

Output

For each test case, display the expression name and a colon on the first line. Then display the result of each question on a line. The layout of the output is shown in the sample output.

You may assume that all expressions and functions are always valid.

Display a blank line between test cases.

Sample Input

```
p:=a^b*c+(d*c)^f*z+b
4
2 1 1
2 2
3
3 1
a:=(x+y)
3
1
1 2
1 2 1
*
```

Output for the Sample Input

```
Expression p:
op(1,op(1,op(2,p)))=(d*c)
op(2,op(2,p))=z
op(3,p)=b
op(1,op(3,p))=b

Expression a:
op(1,a)=x+y
op(2,op(1,a))=y
op(1,op(2,op(1,a)))=y
```

Problem H

Fishing Net

Input file: net.in

In a highly modernized fishing village, inhabitants there make a living on fishery. Their major tools, fishing nets, are produced and fixed by computer. After catching fishes each time, together with plenty of fishes, they will bring back the shabby fishing nets, which might be full of leaks. Then they have to inspect those nets. If there exist large leaks, they have to repair them before launching out again.

Obviously, the smaller the leaks in the fishing nets are, the more fishes they will catch. So after coming back, those fishermen will input the information of the fishing nets into the computer to check whether the nets have leaks.

The checking principle is very simple: The computer regards each fishing net as a simple graph constructed by nodes and edges. In the graph, if any circle whose length (the number of edges) is larger than 3 must has at least one chord, the computer will output "Perfect" indicating that the fishnet has no leaks. Otherwise, "Imperfect" will be displayed and the computer will try to repair the net.

Note: A circle is a closed loop, which starts from one node, passes through other distinct nodes and back to the starting node. A chord is an edge, which connects two different nodes on the circle, but it does not belong to the set of edges on the circle.

Input

The input file contains several test cases representing different fishing nets. The last test case in the input file is followed by a line containing 0 0.

The first line of each test case contains two integers, *n* and *m*, indicating the number of nodes and edges on the net respectively, $1 \leqslant n \leqslant 1000$. It is followed by *m* lines accounting for the details of the edges. Each line consists of two integers x_i and y_i, indicating there is an edge between node x_i and node y_i.

Output

For each test case, display its checking results. The word "Imperfect" suggests that the corresponding fishing net is leaking, while the word "Perfect" stands for a fishing net in good condition.

Follow the output for each net with a blank line.

Sample Input

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

```
1 2
2 3
3 1
0 0
```

Output for the Sample Input

```
Imperfect

Perfect
```