The 2006 ACM Asia Programming Contest - Shanghai

sponsored by IBM
hosted by Shanghai University
Shanghai, China

November 22, 2006

This problem set should contain ten (10) problems on nineteen (19) pages. Please inform a runner immediately if something is missing from your problem set.

ACM TEAM SELECTION

Input File: acmteam.in

The ACM-ICPC brings together the top student programmers from all over the world, and provides them with opportunities to develop critical skills which will give them a competitive edge when they launch careers in information technology areas. More than 5,600 teams from 84 countries had competed in regional contests last year. An ever larger number of teams - more than 7,000 teams from different countries worldwide - have registered in this year's regional contests. However, due to the limited capacity of each site, only a small amount of the registered teams can be allowed to participate in the on-site contest. It is really hard for the contest organizers to determine which teams should be allowed to participate. One of the possible solutions is to hold a preliminary internet contest before the on-site competition. The following part describes a simplified version of rules for team selection:

Up to three teams from each school can participate in the on-site contest, depending on how many following conditions the school in question meets:

a) A team from this school has solved at least M problems in the preliminary contest;
b) Some of the teams from this school ranked top 20 in previous World Finals;
c) This school has hosted a provincial contest this year.

Your task is to write a program to help the contest holders to calculate how many teams are allowed to participate in the on-site final contest.

Input

There are multiple test cases in the input file. Each test case starts with three integers S, T and M $(1 \leqslant S \leqslant 100, 1 \leqslant T \leqslant 2000, 0 \leqslant M \leqslant 10)$, representing the number of schools, the number of teams participating in the preliminary contest, and the minimum number of problems which is required to be solved in order to enter the on-site competition, respectively.

Each of the following S lines consists of three integers Id, P and Q, $(1 \leqslant Id \leqslant S, 0 \leqslant P, Q \leqslant 1)$, representing the Id of the school, whether this school satisfies condition b), and whether this school satisfies condition c).

The last part of each test case consists of T lines. There are two integers on each of the T lines, Sid and Tot $(1 \leqslant Sid \leqslant S, 0 \leqslant Tot \leqslant 10)$, meaning that a team from school Sid had solved Tot problems in the preliminary contest.

Two consecutive test cases are separated by a blank line. $S = 0, T = 0, M = 0$ indicates the end of input and should not be processed by your program.

Output

For each test case, print the total number of teams which are allowed to participate in the on-site competition on a separate line in the format as indicated in the sample output.

Sample Input	Output for the Sample Input

```
5 8 6
5 0 1
4 0 0
1 0 0
3 1 1
2 1 1
2 6
3 3
2 9
5 7
4 8
3 6
2 8
1 6

5 8 6
3 0 1
5 1 1
2 0 1
1 1 1
4 1 0
5 7
2 5
4 5
5 5
3 3
5 6
2 0
4 7

0 0 0
```
	```
	Case 1:  10
	Case 2:  9
	```

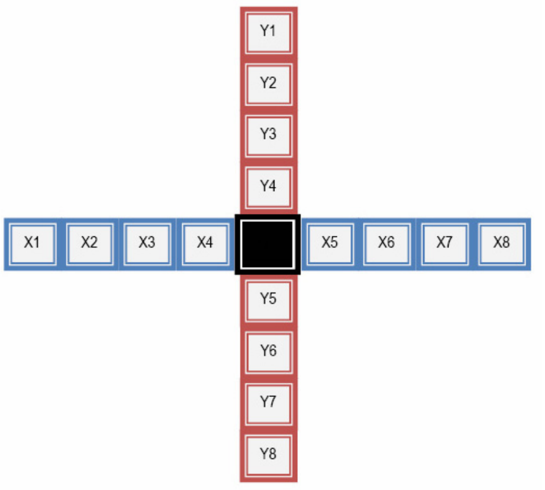
BALANCING THE SCALE

Input File: balance.in

You are given a strange scale (see the figure below), and you are wondering how to balance this scale. After several attempts, you have discovered the way to balance it — you need to put different numbers on different squares while satisfying the following two equations:

$$\begin{array}{l}x_1 * 4 + x_2 * 3 + x_3 * 2 + x_4 = x_5 + x_6 * 2 + x_7 * 3 + x_8 * 4 \\ y_1 * 4 + y_2 * 3 + y_3 * 2 + y_4 = y_5 + y_6 * 2 + y_7 * 3 + y_8 * 4\end{array}$$

How many ways can you balance this strange scale with the given numbers?



Input

There are multiple test cases in the input file. Each test case consists of 16 distinct numbers in the range [1, 1024] on one separate line. You are allowed to use each number only once.

A line with one single integer 0 indicates the end of input and should not be processed by your program.

Output

For each test case, if it is possible to balance the scale in question, output one number, the number of different ways to balance this scale, in the format as indicated in the sample output. Rotations and reversals of the same arrangement should be counted only once.

Sample Input

```
87 33 98 83 67 97 44 72 91 78 46 49 64 59 85 88
0
```

Output for the Sample Input

```
Case 1:  15227223
```

Contestants division

Input File: contest.in

In the new ACM-ICPC Regional Contest, a special monitoring and submitting system will be set up, and students will be able to compete at their own universities. However there's one problem. Due to the high cost of the new judging system, the organizing committee can only afford to set the system up such that there will be only one way to transfer information from one university to another without passing the same university twice. The contestants will be divided into two connected regions, and the difference between the total numbers of students from two regions should be minimized. Can you help the juries to find the minimum difference?

Input

There are multiple test cases in the input file. Each test case starts with two integers N and M, $(1 \leqslant N \leqslant 100000, 1 \leqslant M \leqslant 1000000)$, the number of universities and the number of direct communication line set up by the committee, respectively. Universities are numbered from 1 to N. The next line has N integers; the K^{th} integer is equal to the number of students in university numbered K. The number of students in any university does not exceed 100000000. Each of the following M lines has two integers s, t, and describes a communication line connecting university s and university t. All communication lines of this new system are bidirectional.

$N = 0, M = 0$ indicates the end of input and should not be processed by your program.

Output

For every test case, output one integer, the minimum absolute difference of students between two regions in the format as indicated in the sample output.

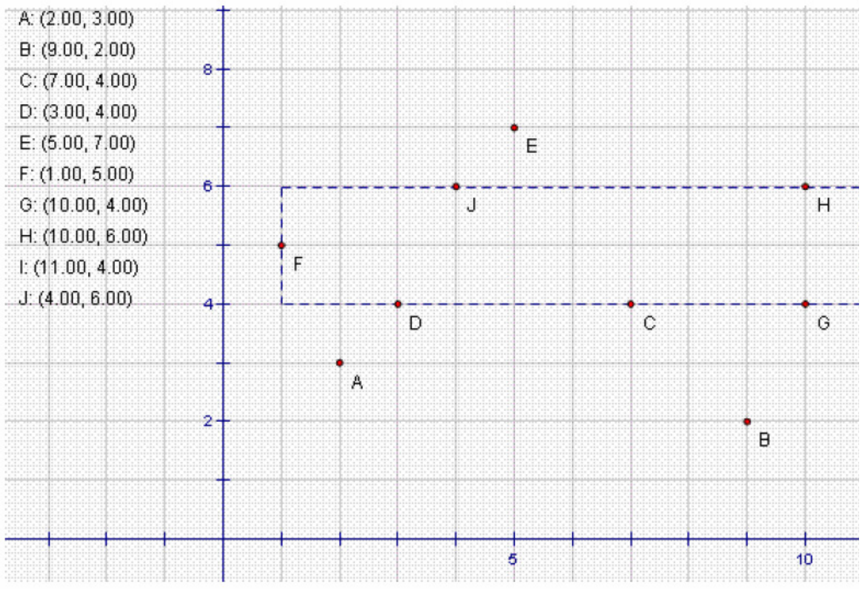
Sample Input	Output for the Sample Input

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

DISTANT GALAXY

Input File: distant.in

You are observing a distant galaxy using a telescope above the Astronomy Tower, and you think that a rectangle drawn in that galaxy whose edges are parallel to coordinate axes and contain maximum star systems on its edges has a great deal to do with the mysteries of universe. However you do not have the laptop with you, thus you have written the coordinates of all star systems down on a piece of paper and decide to work out the result later. Can you finish this task?



Input

There are multiple test cases in the input file. Each test case starts with one integer N, $(1 \leqslant N \leqslant 100)$, the number of star systems on the telescope. N lines follow, each line consists of two integers: the X and Y coordinates of the K^{th} planet system. The absolute value of any coordinate is no more than 10^9, and you can assume that the planets are arbitrarily distributed in the universe.

$N = 0$ indicates the end of input file and should not be processed by your program.

Output

For each test case, output the maximum value you have found on a single line in the format as indicated in the sample output.

Sample Input	**Output for the Sample Input**

`10`
`2 3`
`9 2`
`7 4`
`3 4`
`5 7`
`1 5`
`10 4`
`10 6`
`11 4`
`4 6`
`0`
	`Case 1:  7`

Escape plan

Input File: escape.in

Without turning his head, Vader snarled through his mask, "The Millennium Falcon?"

Piett paused a moment before replying. He would have preferred to avoid that issue. "Our tracking scanners are on it now," he responded a bit fearfully.

Vader turned to face the admiral, his towering figure looming over the frightened officer. Piett felt a chill course through his veins, and when the Dark Lord spoke again his voice conveyed an image of the dreadful fate that would be inflicted if his commands were not executed.

"I want that ship," he hissed.

The ice planet was rapidly shrinking to a point of dim light as the Millennium Falcon sped into space. Soon that planet seemed nothing more than one of the billions of light specks scattered throughout the black void.

But the Falcon was not alone in its escape into deep space. Rather, it was followed by an Imperial fleet that included the Avenger Star Destroyer and a half-dozen TIE fighter. The fighters moved ahead of the huge, slower-moving Destroyer, and closed in on the fleeing Millennium Falcon.

— Star Wars, Episode V, the Empire Strikes Back

You are driving Millennium Falcon, the fastest starship of the entire galaxy, to escape Imperial pursuit. There are N planet systems, numbered from 0 to $N-1$, and some of them are connected by one-way hyperspace tunnels. Some hyperspace tunnels are passable at all times, while others are only available at certain times. Initially you are at the ice planet *Hoth*, the system numbered 0, at time 0 and you need to get to *Sullust*, the system numbered $N-1$. Since there are K Imperial Star Destroyers following you, and they will also make the jump into the hyperspace to continue the pursuit, you have decided to use the $K+1^{\text{th}}$ shortest path from *Hoth* to *Sullust* so as to minimize the possibility of being attacked halfway by Imperial starships; furthermore, in order to avoid detection, you do not want to risk staying at a system longer than T seconds.

Note that multiple shortest paths may require same travel time, and your travel path may not be simple (i.e. you are allowed to visit some systems and use some hyperspace tunnels more than once during your journey).

Input

There are multiple test cases in the input file. Each test case starts with four integers N, M, K and T, ($1 \leqslant N \leqslant 100$, $0 \leqslant M \leqslant 500$, $0 \leqslant K \leqslant 9$, $0 \leqslant T \leqslant 100$), the number of planet systems, the number of hyperspace tunnels between them, the pursuing Imperial Star Destroyers, and the maximal allowed time to stay at the same system, respectively. M lines follow, each line describes one of the hyperspace tunnels: four integers U, V, C, and W, ($0 \leqslant U, V \leqslant N-1$, $1 \leqslant C \leqslant 10$, $1 \leqslant W \leqslant 1000000$) meaning there's a tunnel from U to V, traveling through this path requires W seconds and it is only available every C seconds, starting from time 0 (i.e. 0, C, $2*C$, $3*C$ ... seconds).

Two successive inputs are separated by a blank line. $N = 0$, $M = 0$, $K = 0$, $T = 0$ indicates the end of input and should not be processed by your program.

Output

For every test case, you should output one integer on a separate line, the total time we need to reach *Sullust* in the format as indicated in the sample output; output `-1` if no such path can be found.

Sample Input

```
5 9 2 2
1 2 5 5
2 4 6 6
0 2 1 8
1 4 4 3
3 0 1 8
1 3 5 10
0 4 4 4
2 3 3 4
3 1 5 10

10 0 0 0

0 0 0 0
```

Output for the Sample Input

```
Case 1:  28
Case 2:  -1
```

Explanation for Sample Input / Output

The 1st shortest path in this example is $0 \to 4$, with a total travel time of 4 seconds; the 2nd shortest path is 0 (Wait 2 seconds) $\to$ 2 (Wait 2 seconds) $\to$ 4, with a total travel time of 18 seconds; the 3rd shortest path is 0 (Wait 2 seconds) $\to$ 2 (Wait 2 seconds) $\to 3 \to 0 \to 4$, with a total travel time of 28 seconds.

FALSE PERCEPTIONS

Input File: false.in

"There is no denying the clues. They are clearly there; you can't miss them. But are we interpreting the clues correctly? That is where the author tricks us. There are most definitely more than one way to look at a clue and what it signifies, but the way the author has laid them out, (like a breadcrumb trail) we are inclined to believe they lead us to the end conclusion . . ."

You have been reading Harry Potter series for some time. Since the final book will not be out in the store until 2007, and you are curious about what could happen in the final book of the series, you have decided to do some guessing by yourself. After rereading the series for some days, you have divided every piece of information you currently have into three categories:

A. *Extracts*. Extracts are taken from the previous published books in the series, and you don't need to check for their validities;
B. *Assumptions*. Assumptions are based on different interpretations of other assumptions / extracts from the books. An assumption is plausible if and only if some of the assumptions (or extracts from the book) this assumption relies on are interpreted in ways that support this assumption. No assumption requires itself to be plausible, whether directly or indirectly.
C. *Theories*. Theories are based on extracts and assumptions. If all the extracts and assumptions one theory depends on are believable and interpreted to support this theory, then we say this theory can be argued as believable.

Note that extracts and assumptions may have many different exclusive explanations leading to other theories and/or assumptions.

Because of personal preferences, you want to see some scenes in the next book come true much more than the others, thus you have assigned different values to different theories, each being an exponent of 2, less or equal to $2^{(\text{number of theories}-1)}$. You want to know what the maximal possible value of theories that can be argued as believable at the same time is; furthermore, since the less guesswork you do, the more reliable your theories are, you are interested in the minimum number of explanations you must make to make all those theories acceptable. Can you accomplish this seemingly impossible task?

Input

The input consists of several cases, each followed by a blank line.

Each test case starts with three integers N, M and K ($1 \leqslant N \leqslant 1000$, $1 \leqslant M \leqslant 5000$, $1 \leqslant K \leqslant 100$), the number of extracts from the book, the number of assumptions and the number of theories, respectively.

The following part describes extracts and assumptions. Extracts are always described before assumptions. Each description starts with one string S, the name of the *extract / assumption*, and an integer T in the next line, the number of different explanations you have come up with; T lines follow, on each line there is a string X, the explanation which leads to another *assumption / theory*.

The next part of test case consists of K descriptions of theories. Each description starts with one string S, the name of the *theory*, followed by one integer P, the value which you have assigned to this *theory*.

Two successive descriptions are separated by a line with one single character '-'. It is guaranteed that the total numbers of explanations which extracts and assumptions have does not exceed 50000, and you may assume every string's length is less or equal to 20, and only consists of letters from alphabet and numbers $0 \ldots 9$.

$N = 0$, $M = 0$, $K = 0$ indicates the end of input file and should not be processed by your program.

Output

For every test case, output the maximum value followed by the minimum number of explanations you have to make in the format as indicated in the sample output. Print all the plausible theories (with the maximum total value) on the following lines. Separate the result of two successive inputs with one blank line.

Sample Input

```
2 3 1
Extract1
2
Assumption1
Assumption2
-
Extract2
2
Assumption2
Assumption3
-
Assumption1
2
Assumption2
Theory1
-
Assumption2
0
-
Assumption3
1
Theory1
-
Theory1
1

0 0 0
```

Output for the Sample Input

```
Case 1:
1
4
Theory1
```

Explanation for Sample Input / Output

In order to make `Theory1` believable, `Extract1`, `Extract2`, `Assumption1`, and `Assumption3`'s explanations must be `Assumption1`, `Assumption3`, `Theory1` and `Theory1`, respectively. Thus the total number of explanations we must make is 4.

GUARDING ZION

Input File: guard.in

It is the 22th century, and machines with supreme artificial intelligence have conquered the world. Humans are kept alive by a computer controlled program, the Matrix, for no other purpose than providing energy to the machines. Only a small group of freedom fighters have escaped the Matrix and built a secret resist base, Zion, near the core of the earth. However the machines have discovered this and send down millions of sentinels to destroy Zion.

The underground world is actually the sewers of the previous cities in ruin — gigantic tunnels connecting the ground, Zion, and various other intersections. However all sewers have been destroyed when the machines first conquered the mankind. Because of limited time, the sewers have been repaired by people from Zion such that there will always be no more than one path from one intersection to another. No tunnel connects an intersection to itself. The best weapon we have is EMP (electromagnetic pulse) charge, an extremely powerful weapon to use against machines. Blowing an EMP charge will cause any electronic device within its range to stop functioning, therefore destroying the machines within range completely. However it can also cause another EMP charge within the range to malfunction, so the distance between any two EMP charges must be no less than their range.

The council has decided to put EMP on certain intersections to minimize the possibility of the machines destroying Zion. Since our resources are limited, we can only afford to put EMP charges on certain intersections of the underground tunnels. Deploying an EMP charge on a certain intersection has a non-negative cost. What is the maximal distance you can cover with EMP charges, and what's the minimum cost to achieve that maximal distance?

Input

There are multiple test cases in the input file. Each test case starts with three integers N, M and D, $(2 \leqslant N \leqslant 300,\ 1 \leqslant M \leqslant 3000)$, the number of intersections, the number of edges, and the range of each EMP charge, respectively. Intersections are numbered from 0 to $N-1$. The following line consists of N integers, the cost of deploying an EMP charge on intersection i. The next M lines each consists of three integers S, T, and C, $(0 \leqslant S, T \leqslant N-1,\ 1 \leqslant C \leqslant 20000)$, meaning that intersection S and T are connected by a tunnel whose length is C. Every test case ends with one blank line indicating the end of test case.

$N = 0$, $M = 0$, $D = 0$ indicates the end of input file and should not be processed by your system.

Output

For each test case, output two integers in the format as indicated in the sample output: the maximum distance EMPs can cover and the minimum cost you've found on a single line.

Sample Input	**Output for the Sample Input**

`2 1 3`
`5 6`
`1 0 6`
 	
`0 0 0`
	`Case 1:  6 11`

Harmony forever

Input File: harmony.in

We believe that every inhabitant of this universe eventually will find a way to live together in harmony and peace; that trust, patience, kindness and loyalty will exist between every living being of this earth; people will find a way to appreciate and cooperate with each other instead of continuous bickering, arguing and fighting. Harmony — the stage of society so many people dream of and yet it seems so far away from now . . .

Fortunately, the method of unlocking the key to true Harmony is just discovered by a group of philosophers. It is recorded on a strange meteorite which has just hit the earth. You need to decipher the true meaning behind those seemingly random symbols. . . More precisely, you are to write a program which will support the following two kinds of operation on an initially empty set S:

1. $B\ X$: Add number X to set S. The K^{th} command in the form of $B\ X$ always happens at time K, and number X does not belong to set S before this operation.
2. $A\ Y$: Of all the numbers in set S currently, find the one which has the minimum remainder when divided by Y. In case a tie occurs, you should choose the one which appeared latest in the input. Report the time when this element is inserted.

It is said that if the answer can be given in the minimum possible time, true Harmony can be achieved by human races. You task is to write a program to help us.

Input

There are multiple test cases in the input file. Each test case starts with one integer T where $1 \leqslant T \leqslant 40000$. The following T lines each describe an operation, either in the form of "$B\ X$" or "$A\ Y$" where $1 \leqslant X, Y \leqslant 500000$.

$T = 0$ indicates the end of input file and should not be processed by your program.

Output

Print the result of each test case in the format as indicated in the sample output. For every line in the form of "$A\ Y$", you should output one number, the requested number, on a new line; output `-1` if no such number can be found. Separate the results of two successive inputs with one single blank line.

Sample Input	Output for the Sample Input

```
5
B 1
A 5
B 10
A 5
A 40
2
B 1
A 2
0
```
	```
	Case 1:
	1
	2
	1

	Case 2:
	1
	```

Interesting Yang Hui Triangle

Input File: interesting.in

Harry is a Junior middle student. He is very interested in the story told by his mathematics teacher about the Yang Hui triangle in the class yesterday. After class he wrote the following numbers to show the triangle our ancestor studied.

```
                              1
                        1           1
                    1         2           1
                1       3           3         1
            1       4         6           4       1
        1       5       10          10        5       1
    1       6       15        20          15      6       1
1       7       21      35          35        21      7       1
                            ......
```

He found many interesting things in the above triangle. It is symmetrical, and the first and the last numbers on each line is 1; there are exactly i numbers on the line i.

Then Harry studied the elements on every line deeply. Of course, his study is comprehensive.

Now he wanted to count the number of elements which are the multiple of 3 on each line. He found that the numbers of elements which are the multiple of 3 on line 2, 3, 4, 5, 6, 7, . . . are 0, 0, 2, 1, 0, 4, . . . So the numbers of elements which are not divided by 3 are 2, 3, 2, 4, 6, 3, . . . , respectively. But he also found that it was not an easy job to do so with the number of lines increasing. Furthermore, he is not satisfied with the research on the numbers divided only by 3. So he asked you, an erudite expert, to offer him help. Your kind help would be highly appreciated by him.

Since the result may be very large and rather difficult to compute, you only need to tell Harry the last four digits of the result.

Input

There are multiple test cases in the input file. Each test case contains two numbers P and N, ($P < 1000$, $N \leqslant 10^9$), where P is a prime number and N is a positive decimal integer.

$P = 0$, $N = 0$ indicates the end of input file and should not be processed by your program.

Output

For each test case, output the last four digits of the number of elements on the $N + 1$ line on Yang Hui Triangle which can not be divided by P in the format as indicated in the sample output.

Sample Input	**Output for the Sample Input**

`3 4`
`3 48`
`0 0`
	`Case 1: 0004`
	`Case 2: 0012`

JOHN'S CANONICAL DIFFERENCE BOUND MATRICES

Input File: john.in

When John studied the timed automaton, he met the problem about how to trigger the machine. With the problem deeply studied, he found that it can be ascribed to the clock constraints of the timed automaton. The timed automation in question is described below:

The clock variables, or simply clocks, are variables whose values are integers. Of course, time passes at the same rate for all clocks, and any clock can be reset to zero. John uses C to denote the finite set of clocks, and defines the clock constraints for C as follows:

(1) All inequalities of the form $t\#c$ or $c\#t$ are clock constraints, where t is a clock, $\#$ is either $<$ or $\leqslant$, and c is an integer.

(2) If A_1 and A_2 are clock constraints, then $A_1 \wedge A_2$ is a clock constraint.

John notes that a clock constraint can define several regions in some multidimensional space. He wants to know such regions, so he defines the clock zones recursively as follows.

$$A = x < c \mid x \leqslant c \mid c < x \mid c \leqslant x \mid x - y < c \mid x - y \leqslant c \mid A_1 \wedge A_2$$

For simplicity, he let $C_0 = C \cup \{x_0\}$, where x_0 is a reference clock whose value is always 0. The clock zone A can be described by a Difference Bound Matrix D (called a *DBM*) which is a matrix (D_{ij}) of size $|C_0| \times |C_0|$. Each D_{ij} has the form $(d_{ij}, \#)$, where $d_{ij} \in Z \cup \{\$\}$, $\# \in \{<, \leqslant\}$. The value of D_{ij} can be evaluated in the following form:

For every inequality $x_i - x_j \# d_{ij}$ in clock zone A, let $D_{ij} = (d_{ij}, \#)$, where x_i and x_j are two clocks. If the bound of $x_i - x_j$ for x_i and x_j is unknown, let $D_{ij} = (\$, <)$.

For example, DBM of the clock zone given by $x_1 - x_2 < 2 \ \wedge \ 0 < x_2 \leqslant 2 \ \wedge \ 1 \leqslant x_1$ is shown below:

$$\begin{pmatrix} (0, <=) & (-1, <=) & (0, <) \\ (\$, <) & (0, <=) & (2, <) \\ (2, <=) & (\$, <) & (0, <=) \end{pmatrix}$$

The representation of a clock zone by a DBM is not unique. In this example, there are some implied constraints that are not reflected in the DBM. Since $x_1 - x_2 < 2$ and $x_2 \leqslant 2$, it must be the case $x_1 < 4$. Since $x_0 = 0$, the original $D_{10} = (\$, <)$ can be changed into $D_{10} = (4, <)$. Such adjusting operation is called the tighten operation.

Now John wants to do the similar adjusting operations of difference bounds for all clocks x_i and x_j until further application of this tighten operation does not change the matrix. John obtains the following new canonical difference bound matrix:

$$\begin{pmatrix} (0,<=) & (-1,<=) & (0,<) \\ (4,<) & (0,<=) & (2,<) \\ (2,<=) & (1,<) & (0,<=) \end{pmatrix}$$

Note that some clock zone may contain contrary conditions and has not canonical difference bound matrix.

But John can not obtain a canonical difference bound matrix for a complex clock zone. He asks for your help.

Input

The first line of the input file is a single integer T $(1 \leqslant T \leqslant 20)$, which is the number of test cases you must process, followed by T test cases:

Each test case consists of several lines. Four integers i, j, d and r are given on each line, representing a constraint $x_i - x_j < d$ or $x_i - x_j \leqslant d$ $(0 \leqslant i, j \leqslant m, -10000 < d < 10000)$. If $r = 0$, then this line represents an inequality in the form of $x_i - x_j < d$, otherwise it represents an inequality in the form of $x_i - x_j \leqslant d$. The maximal index m of clocks indicates that the indexes of the clocks are $0, 1, \ldots, m$, $(1 \leqslant m \leqslant 100)$. Note that you have to get the value of m by yourself.

A symbol # given on a single line indicates the end of a test case.

Output

For each test case, first output "`Case #:`" on a single line, where # is the case number starting from 1. Print a blank line after each test case.

For each test case, output the description of the canonical difference bound matrix. If it doesn't have a canonical difference bound matrix, print "`Canonical DBM does not exist.`" (without quotes); If it has a such canonical difference bound matrix, print the matrix in the format as indicated in the sample output. Every element D_{ij} of the matrix should be written in the form $(d_{ij}, \#)$, where # is either `<` or `<=`. If the bound of $x_i - x_j$ for x_i and x_j is unknown, print `($,<)` at the position (i, j). Two consecutive elements on each row should be separated by a single space.

Sample Input	**Output for the Sample Input**

```
1
1 2 2 0
0 2 0 0
2 0 2 1
0 1 -1 1
#
```
	```
	Case 1:
	(0,<=) (-1,<=) (0,<)
	(4,<) (0,<=) (2,<)
	(2,<=) (1,<=) (0,<=)
	```