SPRING 13

PROGRAMMING COMPETITION

MARCH 23, 2013

ARIZONA STATE UNIVERSITY - WOMEN IN COMPUTER SCIENCE

Table of Contents

I. Mr. Koplu's Extravaganza

Mr. Koplu is preparing an extravagant experiment to showcase at the next World's Fair. The experiment involves a certain number of different liquid chemical substances, which have been mixed together and are currently reacting with each other. There are n different types of liquid substances. However, each substance appears in k different variants called isomers, and Mr. Koplu cannot distinguish between the different isomers...

We can assume that time is discrete and measured in minutes. During each minute, each possible reaction is described as follows: a b ratio type, meaning that ratio of liquid a (i.e., part of total amount, no matter which isomer) transforms into liquid b. If the reaction type is normal, then isomer i of liquid a transforms into isomer i of liquid b. If reaction type is special, then isomer i transforms into isomer $i+1$.

At time 0, each substance is in its initial isomer form, having number 1. For the special threshold value k, once isomer k of any substance passes through a special reaction, it transforms into solid state matter and precipitates from the mixture, no longer taking part in reactions (so, for all practical purposes, we can say it disappears). Moreover, during each minute, a small part of each liquid disappears (transforms into gaseous state and evaporates). From Mr. Koplu's point of view, the evaporation happens just before the reactions in any given minute.

Starting from time 0, exactly once a minute, Mr. Koplu takes note of the amount of each liquid in the mixture. He does this for a very, very long time, and then he sums the obtained amounts for each liquid (i.e., for each liquid, he adds the amount there was at time 0, at time 1, at time 2, etc.). The resulting n numbers are the result of the experiment. However, Mr. Koplu would like to know the results of experiment beforehand, so he asks you for a little help. Compute the results of his experiment, knowing that initially he just mixes the same amount (1 unit) of isomer 1 of each substance.

Description of Input

First, $1 \leq \mathbf{n} \leq 100$, and then $1 \leq \mathbf{k} \leq 1000$. Then, **n** floating-point numbers follow, describing the proportion of liquid of each type that evaporates in each minute. Then, one integer $\mathbf{m} < 10000$, the number of reactions, followed by m reaction definitions. Each definition consists of integers $0 \leq \mathbf{a} < \mathbf{n}$, $0 \leq \mathbf{b} < \mathbf{n}$, floating point $0 < \mathbf{r} \leq 1$, and integer $\mathbf{s} = 0$ or 1, meaning that part r of the total amount of liquid a transforms into liquid **b** every minute (in the normal way if $\mathbf{s} = 0$, and in a special reaction, otherwise). No two pairs (**a**,**b**) are the same between definitions, i.e., always **a** != **b**.

Description of Output

Exactly **n** numbers, representing the measured results of the experiment for each substance. Precision up to at least 6 digits after the decimal point is required.

Sample Case

Input

```
2
10
0.1 0.1
2
0 1 0.5 1
1 0 0.5 1
```

Output

```
8.6556936725 8.6556936725
```

II. Optimizing AstroNautilus

RIOT has sent AstroNautilus to space, a robot that would gather information from outer space and deliver it to us. AstroNautilus has been in outer space for over two years and has been sending us information. Through the years, AstroNautilus' messages have been encoded in way that only the designers of AstroNautilus would only be able to decrypt the message. Unfortunately, the people who design AstroNautilus were fired from RIOT. As former worker of RIOT, it's your duty to decrypt his message and optimize it in a way that is understandable.

Assume you have been given a string c of length l, and a set T of n sample string(s). We will reduce/optimize the string c by using the set T in the following--way:

- Wherever T_i appears as a consecutive substring of the string c, you can delete (or not) it.
- After each deletion, you will get a new string c by joining the part to the left and to the right of the deleted substring.

By that way, try to reduce the given string c to get a new string of minimum length. You can do delete for unlimited times.

Description of Input

- The first line contains the string **c**.
- The second line contains the integer **n**.
- Within the last n lines, the i^{th} line contains the string $\mathbf{T}_i$.

Description of Output

Output on a single line an integer that is the minimum length found.

Sample Case

Input

aaabccd
3
abc
ac
aaa

Output

2

III. Can I haz Calculator!

Through the years, calculators and computers have been simplifying the lives of many students and adults. Some teachers have said that calculators and computers have its disadvantages. Even graduate students tend to exhibit a lack of calculating ability. Many of them can't even calculate a simple problem such as 4 * 6 mentally or even 14 * 15 using pencil and paper. We all know, but who cares?

Professor Martin does care. Professor Martin is an old fashioned guy. He decided to give his students some training in calculating without calculators and computers by creating a set of calculation problems, (like 2100 - 100 = ...). To simplify grading the problems, he constructed them so that almost all of them had 2000 as an answer. Not all of them, of course. His students would be smart enough to catch the pattern, and fill in 2000 for all the problems.

Unfortunately Professor Martin's printer turned out to be even more old-fashioned than the professor, and it couldn't interface with his new printer. Inspecting the printed problems, he soon recognized the pattern: none of the operations was transmitted to the printer.

A problem like:
2100-100=

Was printed as:
2100100=

Fortunately, all the digits and the equal sign were still printed. To make this bad situation much worse, Professor Martin's source file had disappeared. So Professor Martin has another problem: what were his original problems?

Given the fact that the answer (most likely) should be 2000, the line 2100100= could have been any of the lines:

2100-100=
2*100*10+0=
2*100*10-0=
2*10*0100=
2*-100*-10+0=

Professor Martin does remember a few things about how he wrote the problems:

- He is sure that whenever he wrote down a number (other than 0), it would not start with a zero. So 2*10*0100= could not have been one of his problems.
- He also knows he never wrote the number zero as anything but 0. So he would not have a problem like 2*1000+000=.
- He used only binary operators, not the unary minus or plus, so 2*-100*-10+0= was not an option either.
- He used the operators +, - and * only, avoiding the operator / (after all, they were first year students).
- He knew all problems followed the usual precedence and associativity rules.

You are to help Professor Martin recover his problem set by writing a program that when given a row of digits, insert one or more of the operators +, - and * in such a way that the value of the resulting expression equals 2000.

Description of Input

The input consists of one or more test cases. Each test case is a single line containing n digits ('0'–'9'), $1 \leq \mathbf{n} \leq 9$, followed by an equal sign. There will not be any blanks embedded in the input, but there may be some after the equal sign. The last test case is followed by a line containing only the equal sign. This line should not be processed.

Description of Output

For each test case, print the word Problem, then the number of the case, then all possible ways of inserting operators in the row of digits such that the resulting expression has the value 2000, subject to Professor Bartjens' memory of how he wrote the problems. Use the format shown below. If there is more than one possible problem, write the problems in lexicographic order. Each possible problem should be on a new line, indented 2 spaces. If there is no solution the answer IMPOSSIBLE should be printed, indented 2.

Sample Case

Input

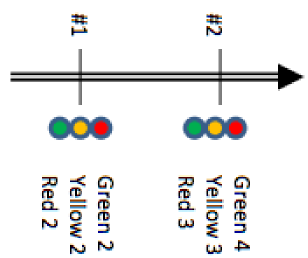
```
2100100=
77=
```

Output

```
Problem 1

  2 * 100 *10 + 0 =
  2 * 100 *10 - 0 =
  2100 - 100 =

Problem 2

  IMPOSSIBLE
```

IV. Surviving Arizona

A produce warehouse in Arizona has a number of Air Conditioning units installed in a successive manner, all in a row. Each AC unit has a light indicator which identifies is current operation status. Green light is turned on for a fixed number of seconds which indicates the unit is currently operating, then the unit turns Yellow for a fixed number of seconds indicating it is currently in temporary cool off state to prevent overheating, then turns Red for a fixed number of seconds indicating it has overheated. To prevent an overload on the main power supply this cycle is repeated for each AC unit, which have their own set of timings. When all units are initially turned on, they start at the beginning of their cycle with a Green light. Given N AC units, with their respective timings, compute the earliest time (in seconds) when all lights show Red.



In the example above, AC unit #1 turns Red at the 5th second, but when this happens, AC unit #2 is Yellow. At the 8th second, AC Unit #2 is Red but AC unit #1 has cycled through and is Green again. You can verify that in the 18th second, both units are Red. You can assume the total maximum cycle (adding Green, Yellow and Red times) is 99 seconds. You can assume $0 < N < 5$ and each Green, Yellow and Red time period is at least 1 second long.

Description of Input

Your program will read from standard input. The first line will contain a positive integer on a line by itself representing the number of AC units, **N**. The next **N** lines represent the lights. Each line of input will contain three positive integers separated by spaces – **G**, **Y** and **R** – representing the number of seconds that light is Green, then Yellow, then Red.

Description of Output

You must output a single positive integer on a line by itself representing the earliest time (in seconds) when all N lights show red.

Sample Case

Input

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

Output

18

V. Tweet Tweet

Two brothers, Marlon and Steve like playing their favorite game Hide and Seek. Today it's raining outside so they want to play something completely new and luckily they came across Twitter on their father's computer.

They saw it for the first time, but were already getting bored to see a bunch of sentences having at most 140 characters each. The only thing they liked to play with it is, closing and opening tweets.

Lets assume there are X number of tweets on the page and each tweet can be easily opened by clicking on it, to see some statistics related to that tweet. Initially all the tweets are closed. Clicking on an open tweet closes it and clicking on a close tweet opens it. There is also a button/key to close all the open tweets. Given a sequence of Y clicks by Marlon, Steve has to guess the total number of open tweets just after each click. Please help Marlon in this game.

Description of Input

First line contains two integers **X Y**, the number of tweets (numbered **1** to **X**) and the number of clicks respectively ($1 \leq \mathbf{X},\ \mathbf{Y} \leq 1000$).

Each of the following Y lines has one of the following:

- CLICK T , where **T** is the tweet number ($\mathbf{1} \leq \mathbf{T} \leq \mathbf{X}$)
- CLOSEALL

Description of Output

Output **Y** lines, where the i^{th} line should contain the number of open tweets just after the i^{th} click.

Sample Case

Input

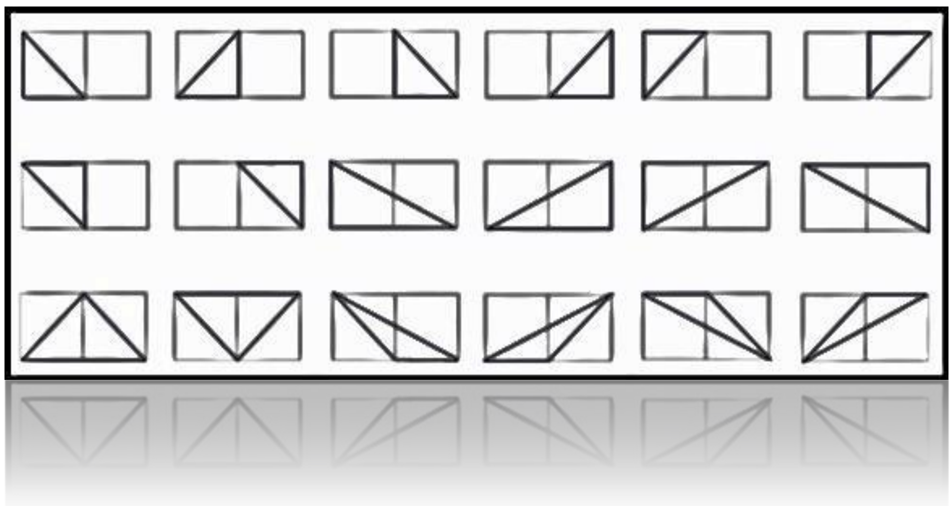
3 6
CLICK 1
CLICK 2
CLICK 3
CLICK 2
CLOSEALL
CLICK 1

Output

1
2
3
2
0
1

VI. Meh!

Everyone knows triangles are formed by having three points, however special triangles called lattice triangles are formed by having vertexes at integer coordinates in a coordinate plane. Your job is given M x N grid find all the possible lattice triangles within the given grid. Below is an example of a grid along with all the possibilities.



Description of Input & Output

You must be able to read in a file. The format of the file is as follows:

- The following lines will be two integer corresponding to M and N(0<=M,N<=200)

The output of the program should be the number of triangles for each case. This will be one single integer value.

Sample Case

Input	**Output**
1 1	4
1 2	18
2 2	76

VII. Graves Grieves!

In Alaska it is always the military officer's main worry to order his soldiers on parade correctly. Luckily, ordering soldiers is not really such a problem. If a platoon consists of X men, all of them have different rank (1 - lowest to X - highest) and on parade they should be lined up from left to right in increasing order of rank.

Sounds easy, doesn't it? Well, Sgt. Graves thought the same, until one day he faced with a new command. He soon discovered that his elite commandos preferred to do the fighting, and leave the thinking to their superiors. So, when at the first roll call the soldiers lined up in fairly random order it was not because of their lack of discipline, but simply because they couldn't work out how to form a line in correct order of ranks. Sgt. Graves was not at all amused, particularly as he soon found that none of the soldiers even remembered his own rank. Over the years of service every soldier had only learned which a true military challenge. After a moment's thought a solution of brilliant simplicity struck him and he issued the following order: "Men, starting from the left, one by one, do : (Step forward; go left until there is no superior to the left of you; get back in line)." . This did indeed get the men sorted in a few minutes. The problem was solved... for the time being.

The next day, the soldiers came in exactly the same order as the day before, and had to be rearranged using the same method. History repeated. After some weeks, Sgt. Graves managed to force each of his soldiers to remember how many men he passed when going left, and thus make the sorting process even faster.

If you know how many positions each man has to walk to the left, can you try to find out what order of ranks the soldiers initially line up in?

Description of Input

The first line of input contains an integer **t** <= 50, the number of test cases. It is followed by t test cases, each consisting of 2 lines. The first line contains a single integer **X** (1 <= **X** <= 200000). The second line contains **X** space separated integers wi, denoting how far the i-th soldier in line must walk to the left when applying Sgt. Graves's algorithm.

Description of Output

For each test case, output a single line consisting of **X** space separated integers - the ranks of the soldiers, given from left to right in their initial arrangement.

Sample Case

Input

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

Output

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

VIII. Kurt's Digitizer!

Kurt is a very studious and obedient boy. He has recently joined Digital Circuits Course in his school. He is absolutely fascinated by digital circuits, and stays back in lab after school hours to make circuits. What interests him is plugging wires and voltage sources on Bread Board and then lighting up an LED.

A Bread Board is a large plastic board, with lots of holes in it. The holes are arranged as following- Holes in a group of 5 shares a common metallic plate, i.e. if one of them gets a high voltage, then all 5 of them are at "High". The group is aligned vertically. There are several groups placed horizontally as well as vertically, depending on the dimensions of the Bread Board. For illustration, consider a Bread Board Having 6 Columns and 2 Rows-

```
A B C D E F
A B C D E F
A B C D E F
A B 1 D E F
A B C D 2 F
G H I J K L
G H I J K L
G H I J K L
G 3 I J K L
G H I J K L
```

All A's are in a group, and share same voltage. Notice the 3 above, it is basically in H group. How can we refer to the point 3 on Bread Board, using coordinate system? Candy for taking a guess! It is (2, 9), from Top Left.

Now if Kurt plugs one end of copper wire in 1 and the other in 2, then C and E will become a single group and start sharing voltage. Kurt can apply a high voltage to any hole, and that group becomes "high". The Bread Board is grounded, thus each group is initially at low voltage. When an LED is connected across two points, it glows up if there is a potential difference between the two points, i.e. when one end is "high" and the other is "low". The lab assistant must leave, but he allows Kurt to stay with one Bread Board, one LED, one Voltage sources (with several output points) and lots of wires. So Kurt gets down to work, he connects wires, applies (high) voltage, and occasionally connects the LED between two points. Sometimes he removes a voltage source as well. At times he connects a wire (or voltage source) to the point, which already has a wire (or voltage source) connected to it. This is not a good practice, but Kurt is excused because he is new to electrical stuff. And when he removes voltage source from a point, he removes only one of it at a time.

Since the number of connections is rather enormous and confusing, he requires your help to tell him whether the LED will light up each time he connects it. However Kurt's cryptography classes have made him rather irksome. He is telling you the row or column not as a number, but in a Base-52 notation of his own. In this notation, each number is represented as 2 characters, where each character can be either A-Z or a-z.

Here A = 0, B = 1, ... Z = 25, a = 26, ... z = 51. So 1 is represented as AB, 5 is AF, 97 is Bt.

And to top that he doesn't give any gap between row number and column number, or between pair of coordinates (column and row), used to indicate the endpoints of wire.

Description of Input

The first line contains 3 space separated integers, **N** ($\leq$ 1000000), **R** ($\leq$ 500), and **C** ($\leq$ 2500).

R is the number of rows and C is the number of columns in the Bread Board. Then N lines follow. Each line starts with one of the following characters-

W - It means a wire is attached in the Board. W is followed by the pair of coordinates, for the two endpoints of wire.

V - It means a voltage is applied on the board. V is followed by the coordinate where the Voltage is applied.

R - It means a voltage source is removed from the board. R is followed by the coordinate from where the source is removed.

L - It means an LED is attached on the Board. L is followed by the pair of coordinates, for the two endpoints of LED.

Note that there is no space between coordinates, and between column and row, for any type of query. See example for clarity.

Description of Output

For each input line starting with '**L**' output "**ON**" if the LED is on, and "**OFF**" otherwise, on separate line.

Sample Case

Input

```
9 2 10
WADAEAFAG
VAGAD
LAFAHAGAE
VAKAK
LAJAKAKAJ
LAKAIAGAB
LABABACAB
RAKAK
LAJAKAKAJ
```

Output

ON
ON
OFF
OFF
OFF

Further Explanation of Sample Input

The input is to be interpreted as

```
9 2 10
W 3 4 5 6
V 6 3
L 5 7 6 4
V 10 10
L 9 10 10 9
L 10 8 6 1
L 1 1 2 1
R 10 10
L 9 10 10 9
```

IX. Cho`Gath's Cupcake Dilemma!

Cho`Gath is planning on serving cupcakes for dessert. However, time is running out and the he has only enough time to bake one batch. Thankfully, the cupcake baking pan that he plans on using has barely enough spaces to ensure each guest receives one cupcake. To his dismay, some of the spaces in the cupcake pan are too close to each other. If cupcake batter is placed in two such spaces then they will bake together and neither will have the aesthetically pleasing round shape one expects from gourmet cupcakes. Not only that he also wants to impress his guests with this dessert, so he wants all cupcakes to be perfectly round. You must determine if it is possible for the desired number of cupcakes to be baked without any two baking in to each other..

Description of Input

The first line consists of a single integer T ≤ 30 denoting the number of test cases to follow. Each test case begins with a single line consisting of three integers **n,m**, and **g**. Here, **n** is the number of spaces in the cupcake pan, **m** is the number of conflicting pairs of spaces in the pan, and g is the number of guests. Following this line is m lines describing the conflicting pairs. Each line consists of two distinct integers i and j, both between 0 and n-1. This means spaces i and j are conflicting so batter may only be placed in at most one of them. The bounds are 1 ≤ **n** ≤ 1,000, 1 ≤ **m** ≤ 20,000, and 0 ≤ **g** ≤ **n**. Since the cupcake pan has barely more spaces than guests (if any), then we also have **n-g** ≤ 15.

Description of Output

The output for each test case consists of a single line containing "Possible" if Cho`Gath can bake all cupcakes perfectly round or "Impossible" if the he must settle for some cupcakes which are imperfect.

Sample Case

Input

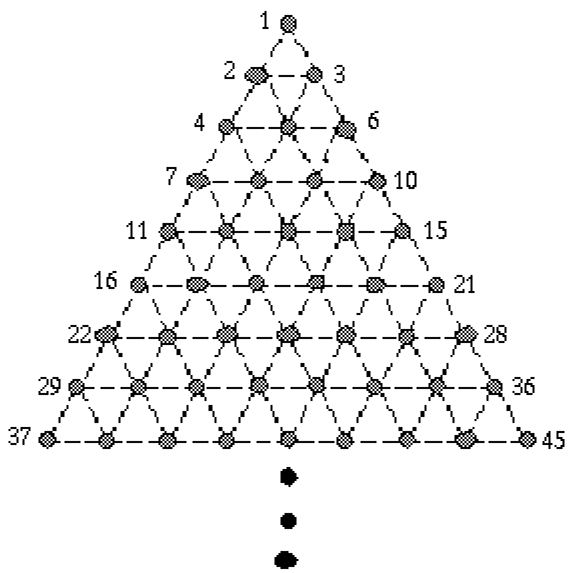
2
3 2 2
0 1
1 2
3 3 2
0 1
1 2
2 0

Output

Possible
Impossible

X. Triangular Tantrum!

Consider the many points on an infinite of equilateral triangles as shown in the figure below.



Note that if we number the points from left to right and top to bottom, then groups of these points form the vertices of certain geometric shapes. For example, the sets of points {1,2,3} and {7,9,18} are the vertices of triangles, the sets {11,13,26,24} and {2,7,9,18} are the vertices of parallelograms, and the sets {4,5,9,13,12,7} and {8,10,17,21,32,34} are the vertices of hexagons.

Write a program which will repeatedly accept a set of points on this triangular grid, analyze it, and determine whether the points are the vertices of one of the following "acceptable" figures: triangle, parallelogram, or hexagon. In order for a figure to be acceptable, it must meet the following two conditions:

1) Each side of the figure must coincide with an edge in the grid.

2) All sides of the figure must be of the same length.

Description of Input

The input will consist of an unknown number of point sets. Each point set will appear on a separate line in the file. There are at most six points in a set.

Description of Output

For each point set in the input file, your program should deduce from the number of points in the set which geometric figure the set potentially represents; e.g., six points can only represent a hexagon, etc. The output must be a series of lines listing each point set followed by the results of your analysis.

Sample Case

Input

1 2 3
11 13 29 31
26 11 13 24
4 5 9 13 12 7
1 2 3 4 5
47
11 13 23 25

Output

1 2 3 are the vertices of a triangle
11 13 29 31 are not the vertices of an acceptable figure
26 11 13 24 are the vertices of a parallelogram
4 5 9 13 12 7 are the vertices of a hexagon
1 2 3 4 5 are not the vertices of an acceptable figure
47 are not the vertices of an acceptable figure
11 13 23 25 are not the vertices of an acceptable figure