Problem A

(Program filename: A.CPP, A.DPR, A.PAS or A.java)

House Numbers

NarmakSung has a hardware shop that makes new digit plates for house numbers. If a house number is 195, for example, he has to create one plate for digit 1, one for digit 9, and one for digit 5. But, the orders are not always that simple. He may get orders to make digit plates, for example, for all houses in one side of a street.

Since making several plates of the same digit costs much less than making all digits for each house one by one, he wants to know, for a big order he receives, how many of each digit plate he has to make.

Input (filename: A.IN)

The first number in the input line, t ($1 \le t \le 10$) is the number of orders. Following this, t orders are written in the input file. Each order starts with a line containing a street name, an arbitrary string of length at most 50 characters. The second line contains a single integer N ($1 \le N \le 10$), the number of sub-orders, followed by N lines of sub-orders. Sub-orders are of three kinds:

- A single house number: in this case, the sub-order line contains only a single integer n ($1 \le n \le 9999$)
- A series of house numbers: in this case, the sub-order line starts with a `'+'`, followed by three integer numbers a, b, c ($1 \le a, b, c \le 9999$). This means that NarmakSung has to make plates for house numbers from a up to b with distance of c. That is, digit plates have to be made for house numbers a, $a+c$, $a+2c$, ..., b. We assume that $a < b$, $b - a$ is a multiple of c, and $c \le b - a$.
- A series of house numbers to be excluded: this kind of sub-orders specifies that a series of house numbers *should not* be made. In this case, the sub-order line starts with a `'-'`, followed by three integer numbers with exactly the same conditions as in the previous case.

Note that if a house number is ordered more than once in two separate sub-orders, it is counted only once if it is not excluded at all (like number 100 in the second test case in the sample input). Also, if a house number is excluded somewhere in the test case, it cancels any order for that number, even if it appears later in the test case (like number 500 in the second sample). Note that it is possible to exclude some numbers that do not appear in other orders at all. In this case, these numbers are ignored (like 900 in the second sample).

Output (filename: A.OUT)

One set of output data is written for each input order consisting of 13 lines. Each set starts with one line containing the street name exactly as appeared in the input order. The next line must be of the form C `addresses` where C is the total number of house numbers to be made. In the special case of $C = 1$, the output line should be `1 address`. The next 10 lines should be of the following form: Line i should contain the number of digit plates needed to be made for digit i. These 10 lines are on the format "`Make` X `digit` Y" where X is how many copies of digit Y they need to make. The last line states the total number Z of digits needed, on the format "`In total` Z `digits`". If there is only one digit to produce, it should say, "`In total 1 digit`", in order to be grammatically correct. The output should be case-sensitive.

Sample Input

```
2
Azadi Street
2
+ 101 103 2
275
Sharif Highway
3
100
- 500 900 100
+  100 800 100
```

Sample Output

```
Azadi Street
3 addresses
Make 2 digit 0
Make 3 digit 1
Make 1 digit 2
Make 1 digit 3
Make 0 digit 4
Make 1 digit 5
Make 0 digit 6
Make 1 digit 7
Make 0 digit 8
Make 0 digit 9
In total 9 digits
Sharif Highway
4 addresses
Make 8 digit 0
Make 1 digit 1
Make 1 digit 2
Make 1 digit 3
Make 1 digit 4
Make 0 digit 5
Make 0 digit 6
Make 0 digit 7
Make 0 digit 8
Make 0 digit 9
In total 12 digits
```

Problem B

(Program filename: B.CPP, B.DPR, B.PAS or B.java)

Rotten Ropes

Suppose we have n ropes of equal length and we want to use them to lift some heavy object. A tear-off weight t is associated to each rope, that is, if we try to lift an object, heavier than t with that rope, it will tear off. But we can fasten a number of ropes to the heavy object (in parallel), and lift it with all the fastened ropes. When using k ropes to lift a heavy object with weight w, we assume that each of the k ropes, regardless of its tear-off weight, is responsible for lifting a weight of w/k. However, if $w/k > t$ for some rope with tear-off weight of t, that rope will tear off. For example, three ropes with tear-off weights of 1, 10, and 15, when all three are fastened to an object, can not lift an object with weight more than 3, unless the weaker one tears-off. But the second rope, may lift by itself, an object with weight at most 10. Given the tear-off weights of n ropes, your task is to find the weight of the heaviest object that can be lifted by fastening a subset of the given ropes without any of them tearing off.

Input (filename: B.IN)

The first line of the input file contains a single integer t ($1 \le t \le 10$), the number of test cases, followed by the input data for each test case. The first line of each test case contains a single integer n ($1 \le n \le 1000$) which is the number of ropes. Following the first line, there is a single line containing n integers between 1 and 10000 which are the tear-off weights of the ropes, separated by blank characters.

Output (filename: B.OUT)

Each line of the output file should contain a single number, which is the largest weight that can be lifted in the corresponding test case without tearing off any rope chosen.

Sample Input

```
2
3
10 1 15
2
10 15
```

Sample Output

```
20
20
```

Problem C

(Program filename: C.CPP, C.DPR, C.PAS or C.java)

Optimal Keypad

Optimus Mobiles produces mobile phones that support **SMS** messages. The Mobiles have a keypad of 12 keys, numbered 1 to 12. There is a character string assigned to each key. To type in the n^{th} character in the character string of a particular key, one should press the key n times. Optimus Mobiles wishes to solve the problem of assigning character strings to the keys such that for typing a random text out of a dictionary of common words, the average typing effort (i.e. the average number of keystrokes) is minimal.

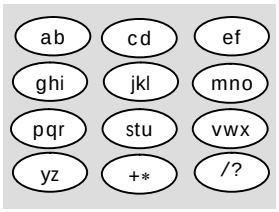


Figure 1

To be more precise, consider a set of characters {a, b, c,…, z, +, *, /, ?} printed on a label tape as in Fig. 2. We want to cut the tape into 12 pieces each containing one or more characters. The 12 labels are numbered 1 to 12 from left to right and will be assigned to the keypad keys in that order.

a	b	c	…….	z	+	*	/	?

Figure 2

You are to write a program to find the 11 cutting positions for a given dictionary of common words. The cutting positions should minimize the average number of keystrokes over all common words in the dictionary. Your output should be a string of 11 characters, where character i in this string is the first character of the $(i+1)^{th}$ label.

Input (filename: C.IN)

The first line contains a single integer t ($1 \le t \le 10$), the number of test cases. Each test case starts with a line, containing an integer M ($1 \le M \le 10000$), the number of common words in the test case. In each M subsequent line, there is a common word. Each common word contains at most 30 characters from the alphabet {a, b, c,…, z, +, *, /, ?}.

Output (filename: C.OUT)

The output contains one line per test case containing an optimal cut string. Obviously, there may be more than a single optimal cut string, so print the optimal cut string which is the smallest one in lexicographic order.

Sample Input

```
2
2
hi
ok
5
hello
bye
how
when
who
```

Sample Output

```
bcdefghijko
bcdefhlnowy
```

Problem D

(Program filename: D.CPP, D.DPR, D.PAS or D.java)

Hey, Pay Day!

An automatic machine is used in a factory for controlling entrance and exit of workers. Each worker has an electronic identification (ID) card that has to be inserted into the machine every time the worker enters or leaves the factory. The machine reads the name (say *S*) from the ID card, and the current date (say, D) and time (say, T) from its internal timer, and generates a new record, denoted by D–S–T, in its database. The entrance time is no sooner than 8:00:00, and the exit time is no later than 20:00:00.

The factory also has a watchman who is always present and carefully records the entrances and exits of all workers and of the clients. He creates a separate list each day that contains the names of the people (workers or clients) who enter or exit the factory during that day. The list is in the same order as the people enter or exit. The names of workers on this list are exactly the same strings as recorded by the automatic machine. Note that clients do not have ID cards.

It is quite possible that some workers forget to insert their ID cards into the machine. But, we assume that each worker makes this mistake at most once each day. We also know that each worker that comes to work in the factory on a certain day, enters the factory exactly once and leaves it exactly once on that day.

At the end of the month, and on the pay day, the director of the factory notes that, with the information recorded by the automatic machine and the lists prepared by the watchman, it is not possible to find out exactly how many hours each worker has been present (working) in the factory in the past month. He, therefore, decides to pay each worker 100K Rials multiplied by the "average" time he/she has been present in the factory during these days. The "average" time is computed as the average of two numbers: one is the minimum possible and the other is the maximum possible time the worker has been present in the factory during the working days in the past month.

You are to write a program to find out the minimum and the maximum possible time each worker has been present in the factory.

Input (filename: D.IN)

In the first line of input, there is an integer t ($1 \le t \le 10$), the number of test cases, followed by the data for the test cases. Each line of a test case is a record of machine database or reports of watchman at the end of a day. Each line representing a machine record is like *D–S–T*, where *S* is a string of lower case letters, *D* is a date, formatted like *YY/MM/DD*, and *T* is a time stamp, formatted like *HH:MM:SS*. There is no blank character in this kind of input lines. Each line representing a watchman's report of a single day containing $1 \le k \le 40$ names like the following:

$$D \quad S_1\ S_2\ S_3 \ldots S_k$$

Which means that at date *D* (with format *YY/MM/DD*), first person named S_1 entered or left the factory, then the second person named S_2, and so on. The date and the names are separated by one blank character. In a test case, there cannot be more than one line of kind watchman's report for the same date. The period of interest is no more than 30 days. Person names are not more than 15 characters and no two persons have the same name. Each test case terminates with a line containing a single # character. The company has at least one employee and at most 15 employees. Two machine records of a certain date may have the same time stamp. In this case, the watchman may record the corresponding events in any order.

Output (filename: D.OUT)

The output file contains answers to the test cases with the i^{th} answer corresponding to the i^{th} test case. Each line of an answer is of the form:

$$S\text{-}H_1\ \ M_1\ \ S_1\text{-}H_2\ \ M_2\ \ S_2$$

Where *S* is the name of a worker, H_1, M_1 and S_1 represent the minimum presence time of the worker *S* during period of interest, and H_2, M_2 and S_2 represent the same for maximum presence time of the worker *S*. (H_i, M_i, and S_i specify hours,

minutes and seconds without any leading zeros.) The lines in each answer should be sorted on field *S* alphabetically (dictionary sort). Each answer terminates with a line containing a single # character.

Sample Input

```
2
99/10/03-sarah-09:00:00
99/10/03-sarah-14:00:00
99/10/04-ali-17:00:00
99/10/03-maryam-09:02:00
99/10/03-ali-10:20:15
99/10/04 ali reza sarah hamid ali reza sarah hamid
99/10/03-ali-12:00:00
99/10/03-maryam-09:04:00
99/10/04-sarah-18:00:00
99/10/03 sarah maryam maryam hamid hamid ali hassan ali sarah hassan
99/10/04-ali-13:00:00
#
99/10/03-sarah-09:00:00
99/10/03-sarah-14:00:00
99/10/04-ali-17:00:00
99/10/03-ali-10:20:15
99/10/04 ali sarah ali sarah
99/10/03-ali-12:00:00
99/10/04-sarah-18:00:00
99/10/03 sarah ali ali sarah
99/10/04-ali-13:00:00
#
```

Sample Output

```
ali-5 39 45-5 39 45
maryam-0 2 0-0 2 0
sarah-6 0 0-10 0 0
#
ali-5 39 45-5 39 45
sarah-6 0 0-10 0 0
#
```

Problem E

(Program filename: E.CPP, E.DPR, E.PAS or E.java)

Unfoldung

Ramtung, the senior Ph.D. student, has to propose a problem for the ACM programming contest this year. As he is highly involved in his Ph.D. studies, he cannot think about anything outside his research area; all about shortest paths in computational geometry.

One of the problems in this area is to find the shortest path between two given points on the surface of a polyhedron. A technique that helps finding such paths is *unfolding*. We cut the surface of the polyhedron along some line segments such that it can be unfolded onto a common plane. This flat surface allows us to apply more simple techniques to find the desired path. In the *unfoldung* problem (named after its author, Ramtung) you are to find whether the surface of a given polyhedron can be unfolded into a common plane when cut along a number of its edges.

To simplify your task, we consider as input the outer surface of a solid object composed of unit cubes glued to each other on their faces such that each cube is adjacent to at least one other cube (unless the object consists of a single cube). We say two cubes are adjacent if they have exactly one face in common. We want to unfold the outer surface of the object (i.e., we ignore the faces that are glued) to obtain a flat layout. The input to the problem is the description of the outer surface as well as a number of unit-edges of the surface that are to be cut. For the sake of simplicity, you may assume that the given object is such that every edge of the outer surface is adjacent to exactly two faces of the outer surface.

For example, Fig. a and Fig. b show how the outer surface of two glued cubes is unfolded onto a common plane. In Fig. a, dotted edges are uncut, and solid lines show the ones that are cut. Note that the face *efgh* is not part of the outer surface. The input data to this example is given in the first sample input. (The numbers inside faces of the right layout (Fig. b) are used to identify faces in the sample input data.)

You are to write a program to determine whether such a surface can be laid out onto a common plane after unfolding its faces. By unfolding we mean rotating a face around one of its edges until it becomes co-planar with the other face adjacent to that edge (so the angle made between the faces inside the surface will be 180º). Note that it is possible for the layout obtained after unfolding to overlap. If possible, one can rotate more than one face together.

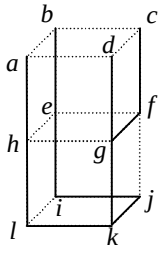


Figure a: A surface with its cut edges.

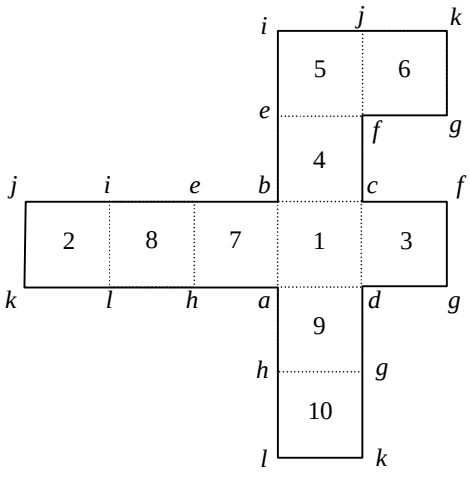


Figure b: The unfolded surface. The face labeled as 1 is the upper face in Fig. (a)

Input (filename: E.IN)

The first line of the input file contains a single integer t ($1 \le t \le 10$), the number of test cases, followed by the input data for each test case. The first line of each test case contains a single integer f ($6 \le f \le 10000$) which is the number of faces on the outer surface. Assume that the faces are numbered uniquely from 1 to f. The second line contains integer n, which is the number of unit edges between faces of the outer surface, followed by exactly n lines each containing a string of the form x+y or x-y forms. x and y are distinct integers ($1 \le x, y \le f$) representing two faces of the surface. Both forms specify face x is adjacent to face y along a common edge. The plus sign shows that the edge common to x and y is cut (solid lines in Fig. a) and the minus sign indicates that the edge is not cut (dotted lines). There is no blank character in a line and there are no empty lines in the input.

Output (filename: E.OUT)

There should be one line per test case containing a string indicating the output to the test case. The output should be the string `CAN UNFOLD` if one can unfold the given surface, `CANNOT UNFOLD` if the surface cannot be unfolded, and `DISCONNECTED` if the surface is separated into two or more pieces by the cut edges. Note that if the surface is disconnected, your output should be `DISCONNECTED` regardless of whether it can be unfolded or not. Be careful that the output is considered case sensitive.

Sample Input

```
2
10
20
1-4
1-3
1-7
1-9
4-5
3+6
7-8
9-10
5+2
6+2
8-2
10+2
7+9
9+3
3+4
4+7
5+8
8+10
10+6
6-5
6
12
1-2
2-3
3-4
4-1
1-5
2-5
3-5
4-5
1-6
2-6
3-6
4-6
```

Sample Output

```
CAN UNFOLD
CANNOT UNFOLD
```

Problem F

(Program filename: F.CPP, F.DPR, F.PAS or F.java)

A DP Problem

In this problem, you are to solve a very easy linear equation with only one variable x with no parentheses! An example of such equations is like the following:

$$2x - 4 + 5x + 300 = 98x$$

An expression in its general form, will contain a '=' character with two expressions on its sides. Each expression is made up of one or more terms combined by '+' or '-' operators. No unary plus or minus operators are allowed in the expressions. Each term is either a single integer, or an integer followed by the lower-case character x or the single character x which is equivalent to $1x$.

You are to write a program to find the value of x that satisfies the equation. Note that it is possible for the equation to have no solution or have infinitely many. Your program must detect these cases too.

Input (filename: F.IN)

The first number in the input line, t ($1 \le t \le 10$) is the number of test cases, followed by t lines of length at most 255 each containing an equation. There is no blank character in the equations and the variable is always represented by the lower-case character `x`. The coefficients are integers in the range (0…1000) inclusive.

Output (filename: F.OUT)

The output contains one line per test case containing the solution of the equation. If s is the solution to the equation, the output line should contain $\lfloor s \rfloor$ (the floor of s, i.e., the largest integer number less than or equal to s). The output should be `IMPOSSIBLE` or `IDENTITY` if the equation has no solution or has infinite solutions, respectively. Note that the output is case-sensitive.

Sample Input

```
2
2x-4+5x+300=98x
x+2=2+x
```

Sample Output

```
3
IDENTITY
```

Problem G

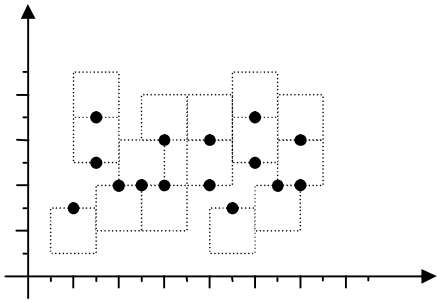
(Program filename: G.CPP, G.DPR, G.PAS or G.java)

Map Labeler

Map generation is a difficult task in cartography. A vital part of such task is automatic labeling of the cities in a map; where for each city there is text label to be attached to its location, so that no two labels overlap. In this problem, we are concerned with a simple case of automatic map labeling.

Assume that each city is a point on the plane, and its label is a text bounded in a square with edges parallel to *x* and *y* axis. The label of each city should be located such that the city point appears exactly in the middle of the top or bottom edges of the label. In a *good labeling*, the square labels are all of the same size, and no two labels overlap, although they may share one edge. Figure 1 depicts an example of a good labeling (the texts of the labels are not shown.)

Given the coordinates of all city points on the map as integer values, you are to find the maximum label size (an integer value) such that a good labeling exists for the map.



Input (filename: G.IN)

The first line contains a single integer t ($1 \le t \le 10$), the number of test cases. Each test case starts with a line containing an integer m ($3 \le m \le 100$), the number of cities followed by m lines of data each containing a pair of integers; the first integer (X) is the x and the second one (Y) is the y coordinates of one city on the map ($-10000 \le$X, Y≤ 10000). Note that no two cities have the same (x, y) coordinates.

Output (filename: G.OUT)

The output will be one line per each test case containing the maximum possible label size (an integer value) for a good labeling.

Sample Input

```
1
6
1 1
2 3
3 2
4 4
10 4
2 5
```

Sample Output

```
2
```

Problem H

(Program filename: H.CPP, H.DPR, H.PAS or H.java)

Balanced Chemical Equations

One cumbersome problem in chemistry is the task of making the number of atoms balanced in a chemical equation. Our problem is concerned with this.

Chemists obey these rules when they present chemical equations:

1. Each element name is abbreviated by at most two letters. The first letter is always in upper-case and the second letter if exists, is a lower-case letter (e.g. Calcium is represented by *Ca*, Oxygen by *O*, and Chlorine by *Cl*).
2. Each molecule is composed of a number of atoms. To represent a molecule, we concatenate the abbreviated names of its composite atoms. For example, *NaCl* represents Sodium Chloride. Each atom name may be followed by a frequency number. For example, Calcium Chloride *CaCl*2 consists of one atom of Calcium and two atoms of Chlorine. If the frequency is not given, it is assumed to be 1 (so *HCl* is equivalent to *H*1*Cl*1). For the sake of simplicity, you may assume that the frequency of an occurrence of an atom is at most 9 (so we do not have *C*11*H*22*O*11 in the problem input). Note that there may be several occurrences of the same atom in the molecule formula, like *H* atom in *CH*3*COOH*.
3. In ordinary chemical reactions, a number of molecules combine and result in a number of other molecules. For example a well known sample of neutralization is:

   $$2HCl + CaO2H2 \rightarrow CaCl2 + 2H2O$$

   This means two molecules of chlorohydric acid (*HCl*) with one molecule of Calcium Hydroxide results in one molecule of Calcium Chloride (*CaCl*2) and two molecules of water.
4. In every chemical reaction, the total number of each atom in the right side of the equation equals the total number of that atom in the left side (that is why it is called an equation!).

Your task is to write a program to find appropriate coefficients x_1, x_2, …, x_M and y_1, y_2, …, y_N to balance an (imbalanced) equation like

$$A_1+A_2+A_3+ \ldots +A_M \rightarrow B_1+B_2+B_3+ \ldots +B_N$$

in the following way:

$$x_1\ A_1+ x_2\ A_2+ x_3\ A_3+ \ldots + x_M\ A_M \rightarrow y_1\ B_1+ y_2\ B_2+ y_3\ B_3+ \ldots + y_N B_N$$

Input (filename: H.IN)

The first line contains an integer t ($1 \le t \le 10$), the number of test cases. Each test case consists of a single line containing an expression like:

$$A_1+A_2+A_3+ \ldots +A_M = B_1+B_2+B_3+ \ldots +B_N$$

Each A_i or B_i is a formula of a molecule according to the rules given in items 1 and 2.

The input equations are given in a way that if they can be balanced, x_i and y_i coefficients can be in the range of 1 to 9. There are less than 10 molecules and there are less than 10 different types of atoms, in a given equation. Additionally, you may assume molecules contain no more than 3 different kinds of atoms. You may also assume that there is no blank character in the input file, and the maximum length of the input lines is 200 characters.

Output (filename: H.OUT)

The output will be one line per test case containing the list of required coefficients, separated by blank characters, in the following order:

$$x_1\ \ x_2\ \ \ldots\ x_M\ y_1\ \ y_2\ \ \ldots\ y_N$$

The coefficients should be integers in the range of 1..9. Obviously, there may be more than one answer for a test case. In such situations, print the answer which minimizes the number:

$$\overline{x_1\ x_2\ \ldots\ x_M\ y_1\ y_2\ \ldots\ y_N}$$

(This is an $(M+N)$-digit decimal number whose digits are x_i and y_i coefficients.) If the equation is impossible to balance, the output line should be `IMPOSSIBLE`.

Sample Input

```
3
HCl+CaO2H2=CaCl2+H2O
HCl+H2SO4=NaCl
HCl+NaOH=NaCl+H2O
```

Sample Output

```
2 1 1 2
IMPOSSIBLE
1 1 1 1
```