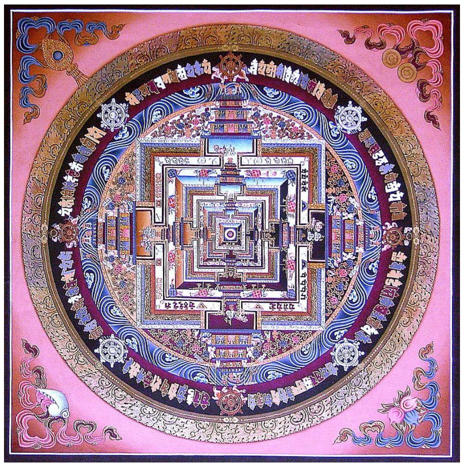
Problem A: The Mandala Effect

Input File: proba.in

As Wikipedia defines it, a "Mandala is a spiritual and ritual symbol in Indian religions, representing the universe." In India, mandalas are major cultural icons commonly found in clothing, books, architecture and etc.



These ornate illustrations are very hard to make because the symmetry must be perfect and the coloration must match exactly. Artists typically spend years training under a master to draw or weave Mandalas. Because an illustration's value comes from the complexity it projects to the eye, the creation of a new pattern is extremely tedious and time consuming. As you can probably tell, there is a faster way of making Mandalas using computer programming with just a quarter of a whole mandala. Figure out how!

Input

The input consists of one mandala. The mandala starts with a line with two integers **m** > 0 and **n** > 0, which indicates how many rows and columns the mandala quarter has, respectively. This is followed by **m** lines, each containing **n** characters, which forms the mandala quarter.

Output

The output consists of the full mandala. A blank line should appear after the mandala.

Sample Input

```
4 10
----@----*
---@----*-
--@----*-|
~~----*-|+
```

Sample Output

```
----@----**----@----
---@----*--*----@---
--@----*-||-*----@--
~~----*-|++|-*----~~
~~----*-|++|-*----~~
--@----*-||-*----@--
---@----*--*----@---
----@----**----@----
```

Problem B: In Case of Emergency

Input File: probb.in

The Philippines is no stranger to natural disasters. We may not get too many tornados or forest fires here but we get more than our fair share of earthquakes, typhoons and floods. Hence, many Filipinos are used to the idea of emergency evacuation and evacuation centers, and in times of calamity, the separation of families is a common event. Due to the sheer volume of evacuees and the chaotic nature of disasters, it is not uncommon for a family to get evacuated to different centers, sometimes even in the same center they can get separated. In times like these, one needs their family more than anything. This is why the "In Case of Emergency" entry in ID cards is important, it can be used to locate your family. Usually, children list their parents, spouses list each other, elderly people list their children and even friends sometimes list each other, in other words, these link family members together. In times of emergency, these are the people they want to be with. This is how families are defined and in emergency situations it is very important to the relief efforts since the amount of resources needed to run an evacuation center is counted in terms of the number of families it contains. Obviously, it takes time to count how many families there are in terms of their emergency contact information, so a computer program that would do it automatically and quickly is of great benefit to disaster response and relief operations, and also to ease the anxiety of people fearing for their missing loved ones.

Input

The input consists of lines of emergency contact persons of persons in an evacuation center. Each line consists of two names separated by a '-'. This means that the person on the left has listed the person on the right as their emergency contact person. You may assume that each name is unique to one person. A person with no emergency contact has themselves as their emergency contact.

Output

The output is a single integer that is the number of families in the evacuation center.

Sample Input

```
Jack - Monty
Lulu - Monty
Beth - Lulu
Edric - Chandra
Chandra - Nissa
Drake - Drake
```

Sample Output

```
3
```

Problem C: A Tail of 0 or 1

Input File: probc.in

If you are familiar with the game called "pin the donkey's tail" where the player is blindfolded and asked to pin the tail of the donkey at its rear, then you might find some similarities with the tail system. The tail system is a mechanism to somehow hide the data being passed over the network just like blindfolding, preventing others to see its real content. Not only that, it doesn't just hide the data but it also ensures that the recipient knows that the data received is the correct data. So the tail system does two functions, to hide the data by transforming it into a different form and when received by the recipient, the system provides a way to check the accuracy of data received.

So how does the tail system work? Transforming data using this method simply pins ***a tail of 0 or 1*** at the rear end of the data then converting the entire data set into its base 10 equivalent. This way, it is the base 10 numerical equivalent that is presented in case data is intercepted. Once data is received by the intended recipient, who of course knows the data format, he/she retransforms the data to its original state by simply dropping off the tail bit. But before that, the recipient must first check if each line of data was received accurately by counting either the number of bit 1's or 0's in the bit series to know if the count results to an even or odd number.

Encoding data in the tail system requires that one must first choose if the bit to be evaluated would be the 1's or the 0's and if the count of the chosen bit 1's or 0's would either be an even or an odd count. Suppose we want to send the small letter **a** over the network. We then choose to evaluate all 1's in the bit series and decide that all 1's must be an even count. Thus, since the small letter **a** has an 8-bit ASCII code of 97 in base 10 which is **0110 0001** in base 2 – notice there are 3 bit 1's in the series - ; to make the count of bit 1's even, we must pin a tail of 1 at its rear, making it **0110 0001 1** (base 2).The 8-bit data now becomes a 9-bit data with the tail, which is further converted to the base 10 equivalent value **194**.

Your task is to simply create a **tail system reader** that will identify if a given set of data is accurate and if so, transform that data back to its original form. Data encoded is limited to all characters in the **8-bit ASCII code system.**

Input

The first line in the INPUT file contains **N** number of integers to be evaluated. The second line contains either zero(**0**) or one(**1**) to identify which bit needs to be counted. The third line contains either the capital letter **E** to denote that the count for the chosen bit must be even or the capital letter **O** to denote that the count must be odd. The succeeding lines contain the series of positive N integers each representing the 9-bit data (8 bits for the ASCII code with 1 tail bit) in base 10 format.

Output

The OUTPUT file contains the series of characters received and decoded. Each character is separated by a new line. If a character is an error, the word **ERROR** is reflected.

CONSTRAINTS

N>0 encoding exclusive for 9 bits only (8-bit ASCII code with 1 bit tail)

SAMPLE INPUT

```
5
1
0
145
202
152
216
158
```

SAMPLE OUTPUT

```
H
ERROR
L
ERROR
O
```

Problem D: p-Norms

Input File: probd.in

The concept of magnitude or absolute value is generalized to the concepts of norms for vectors and matrices. Given an n-dimensional vector

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix},$$

a general vector norm, denoted by $\|\boldsymbol{x}\|$, is a nonnegative norm defined such that:

1. $\|\boldsymbol{x}\| > 0$ when $\boldsymbol{x} \neq 0$ and $\|\boldsymbol{x}\| = 0$ iff $\boldsymbol{x} = \mathbf{0}$.
2. $\|k\,\boldsymbol{x}\|$ = $|k|$ $\|\boldsymbol{x}\|$ for any scalar k.
3. $\|\boldsymbol{x} + \boldsymbol{y}\| \ \leq \|\boldsymbol{x}\| + \ \|\boldsymbol{y}\|$.

Vector norms which are instances of p-norms, where p is an integer, $p > 0$, for an n-vector $\boldsymbol{x}$, is defined by

$$\|\boldsymbol{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}.$$

Important cases are:

- 1-norm:

$$\|\boldsymbol{x}\|_1 = \sum_{i=1}^{n} |x_i|,$$

  sometimes called the Manhattan norm because in the plane it corresponds to the distance between two points as measured in “city blocks”.

- 2-norm:

$$\|\boldsymbol{x}\|_2 = \left(\sum_{i=1}^{n} |x_i|^2\right)^{1/2},$$

  which correspond to the usual notion of distance in Euclidean space, hence this is called the Euclidean norm.

- ∞-norm:

$$\|\boldsymbol{x}\|_\infty = \max_{1\leq i\leq n} |x_i|,$$

  which can be viewed as a limiting case as $p \to \infty$.

For the vector $x = [-1.6, 1.2]^T$, $\|\boldsymbol{x}\|_1 = 2.8$, $\|\boldsymbol{x}\|_2 = 2.0$, and $\|\boldsymbol{x}\|_\infty = 1.6$.

Your task is to compute the 1-,2-, and ∞-norm, given an n-vector $\boldsymbol{x}$.

INPUT

Input consists of several lines. Each line consists a positive integer n, $1 \leq n \leq 1000$, followed by n real numbers representing the elements of the vector in the transposed form. Each element is separated by a single space.

OUTPUT

Output consists of several lines correspond to each line in the input. Each line in the output must consists three real numbers namely the 1-, 2- and ∞-norm of the vector rounded to six (6) decimal places. Each number must be separated by a single space.

SAMPLE INPUT

```
2 -1.6 1.2
3 1 2 3
```

SAMPLE OUTPUT

```
2.800000 2.000000 1.600000
6.000000 3.741657 3.000000
```

Problem E: Sum of Cubes
Input File: probe.in

Any positive integer x can be expressed as a sum of a finite number of cubes

$$x = d_1^3 + d_2^3 + \ldots + d_k^3$$

where $1 \le i \le k$ and d_i is a positive integer. Your task is to minimize the value of k.

To illustrate, $3 = 1^3 + 1^3 + 1^3$ (k is 3), $28 = 3^3 + 1^3$ (k is 2), and $125 = 5^3$ (k is 1).

INPUT

The input starts with a positive number **n** followed by **n** positive integers. The positive integers are separated by whitespaces. Each positive integer x is less than or equal to a million.

OUTPUT

For each non-negative integer **x**, you have to output **x** and the minimum **k** as defined above. Each of the pair of values **x k** should appear in one output line.

SAMPLE INPUT

```
3
3 28 125
```

SAMPLE OUTPUT

```
3 3
28 2
125 1
```

Problem F: The Productive Juan Project – Phase 1

Input File: probf.in

Every Juan needs to be productive. They must learn how to manage time among others. Because of this, the national government has commissioned you and your team to help every Juan manage their schedule.

What the government wants, for a start, is a program that will maximize the number of tasks completed for every Juan in a day.

Now, off you go and help the government!

Input

Input starts with the number of test cases **N**. Succeeding lines are the **N** test cases. A test case starts with the number of time **T** to process. Following lines are the set of pairs for start times and end times. Time follows the 24-hour/military time format.

Output

For each test case, your program must display the number of tasks that a Juan can do in a day. Assume that a Juan can do one task at a time. For cases not defined by the specification, display 'unable to process'

Constraint

$0 < N <= 10$
$0 < T < 21$

Sample Input

```
4
4
0800 0900
0830 1000
0900 1200
5
1200 2000
1600 1800
0500 1200
0800 1200
1030 1230
10
0800 0900
0900 1000
1000 1100
1100 1200
1130 1230
1200 1300
1300 1400
1400 1500
1500 1600
1600 1700
2
800 900
0900
```

Sample Output

```
2
2
9
unable to process
```

Problem G: The Productive Juan Project - Phase 2

Input: probg.in

Every Juan needs to be productive even more. The government asks you to intensify the Productive Juan Project and commence with its second phase.

Goal is simple. Your program must provide the series of tasks that a Juan must perform in a day to be productive.

Now, off you go and work it out in an instant!

Input

Input starts with the number of test cases **N**. Succeeding lines are the **N** test cases. A test case starts with the number of tasks **T** to process. Following lines are the set of tasks, labeled by the letters of the alphabet, along with their respective start time and end time. Time follows the 24-hour/military time format.

Output

For each test case, your program must display the order of tasks that a Juan can do in a day. Assume that (a) a Juan can do one task at a time; (b) shorter tasks are preferred over the longer ones; and (c) earlier start times are preferred for conflicting tasks. For cases not defined by the specification, display 'unable to process'

Constraint

0 < N <= 10
0 < T < 20

Sample Input

```
4
4
A 0800 0900
B 0830 1000
C 0900 1200
5
A 1200 2000
B 1600 1800
C 0500 1200
D 0800 1200
E 1030 1230
10
A 0800 0900
B 0900 1000
C 1000 1100
D 1100 1200
E 1130 1230
F 1200 1300
G 1300 1400
H 1400 1500
I 1500 1600
J 1600 1700
2
A 800 900
B 0900
```

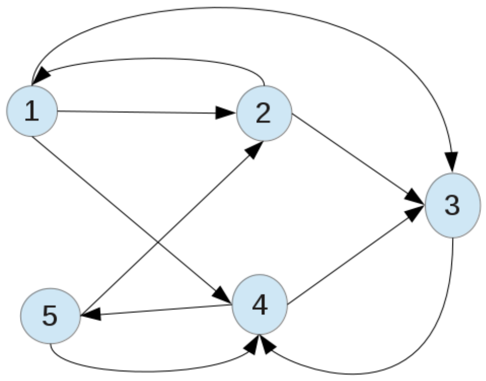
Sample Output

```
AC
EB
ABCDFGHIJ
unable to process
```

Problem H: Counting Paths

Input File: probh.in

Gio was given the graph in the figure below by his teacher. Given the starting vertex 1 and an ending vertex 4, he was asked to count the number of paths of length 3 from 1 to 4.



Gio answered that there are 4 paths:

1->2->1->4,
1->2->3->4,
1->4->3->4, and
1->4->5->4.

Gio was asked again to count the number of paths of length 3 from 4 to 3. He easily counted 3 paths:

4->3->4->3,
4->5->4->3, and
4->2->5->3

Now Gio's exercise can easily be solved using a computer.

Input

The input data consists of several directed graphs and test cases for each graph. The first line in the input file is an integer 1 <= **n** <= 50 referring to the number of graphs. Each graph description is introduced by a line containing the letter "G". The next line contains two integers: the number of vertices 1 <= **v** <= 100 and the number of directed edges, or arcs 1 <= **a** < 9900. This is followed by a lines. Each of these lines refer to an edge and consists of 2 integer, the source vertex s of the edge and the sink vertex e.

The test cases for the graph is introduced by a line containing the character "T". The next line contains an integer **t** referring to the number of test cases. Each test case is in a line containing 3 integers: **x**, **y**, and **l**.

x is the starting vertex in the path, **y** is the ending vertex in the path, and **l** is the length of the path from vertex **x** to vertex **y**.

Output

For each test case, print the number of paths of length **l** from vertex **x** to vertex **y**.

Sample Input

```
2
G
5 10
1 2
1 3
1 4
2 1
2 3
3 4
4 3
4 5
5 2
5 4
T
5
1 4 3
4 3 3
1 1 4
2 5 5
2 4 1
G
4 8
1 2
1 4
2 1
2 3
3 2
3 4
4 1
4 3
T
2
2 4 2
1 1 4
```

Sample Output

```
4
3
2
6
0
2
8
```

Problem I: What's the Number?

Input File: probi.in

Clark and Leah got their happy ending last night. It was a blast! As part of the tradition, the next bride and groom must be chosen. To do this, they gave the eligible bachelors and bachelorretes pieces of paper with texts. Their task is simply to decode a secret number given the following instructions:

- The first number in the text is always the base number.
- If the succeeding number in the text is
    - preceded and succeeded by a vowel, subtract the number from the base number;
    - preceded by a consonant and succeeded by a vowel and vice-versa, add the number from the base; or
    - preceded and succeeded by a consonant, multiply the base by 2.
- For other numbers not defined by the rules above, ignore and proceed to the next.

The last bachelor and bachelorrete to guess the number gets to be the next bride and groom.

Constraint

base number >= 0

Input

Input has several lines of text.

Output

Display the secret number.

Sample Input

```
Just 100 smile for me and le20t the day begin
You are the sunshine that Lights
my he10art with in30
I'm sure t10hat you're an angel in disguise
Come take my ha10nd and together we will rise
On the wings of lov10e
Up and above the clo50uds
The only way to fly is
On the wings of love 1000
```

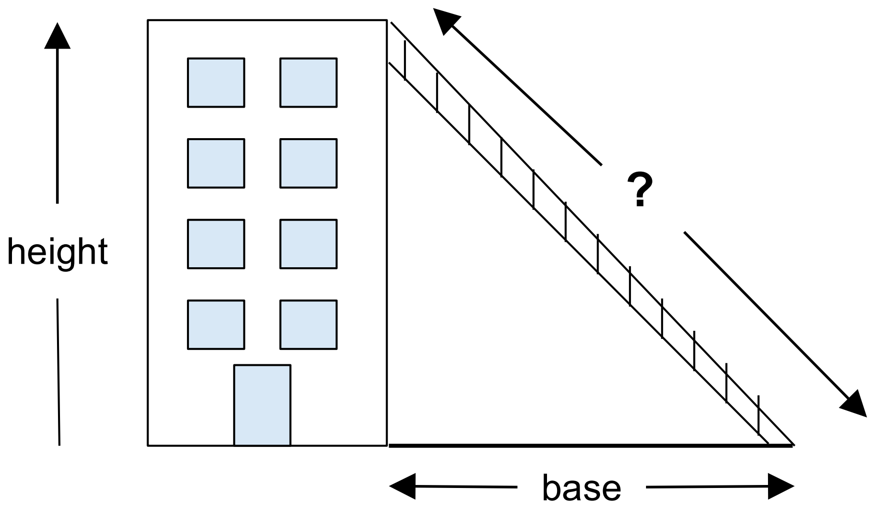
Sample Output

```
230
```

Problem J: Pyth's Magical Ladder

Input File: probj.in

Pyth has acquired a 5-foot magical ladder from a land far far away which can be used in any land structure of any given height and base imaginable. There's just one problem. It does not work and extend unless the exact length of the ladder is whispered or the height of the structure is less than or equal twice its length.



As his best buddy, you are tasked to help Pyth in this task.

Input

There are several lines of input. Each line contains the height of the structure and the width of the base.

Output

For each line of input, display the height of the ladder. Otherwise, display the original length of the ladder.

Constraint

height, base > 0

Sample Input

```
100 20
```

Sample Output

```
101
```