Problem D: I've Got Your Back(gammon)

A friend of yours is working on an AI program to play backgammon, and she has a small problem. At the end of the game, each player's 15 pieces are moved onto a set of 6 board positions called *points*, numbered 1 through 6. The pieces can be distributed in any manner across these points: all 15 could be on point 3; 5 could be on point 6, 2 on point 5, 3 on point 4 and 5 on point 2; etc. Your friend wants to store all these possible configurations (of which there are 15504) into a linear array, but she needs a mapping from configuration to array location. It seems logical that the configuration with all 15 pieces on point 1 should correspond to array location 0, and the configuration of all 15 pieces on point 6 should correspond to the last array location. It's the ones in between that are giving her problems. That's why she has come to you.

You decide to specify a configuration by listing the number of pieces on each point, starting with point 6. For example, the two configurations described above could be represented by $(0, 0, 0, 15, 0, 0)$ and $(5, 2, 3, 0, 5, 0)$. Then you can order the configurations in lexicographic ordering, starting with (0,0,0,0,0,15), then $(0, 0, 0, 0, 1, 14), (0, 0, 0, 0, 2, 13), \ldots, (0, 0, 0, 0, 14, 1), (0, 0, 0, 0, 15, 0), (0, 0, 0, 1, 0, 14)$, $(0, 0, 0, 1, 1, 13)$, etc., ending with $(15, 0, 0, 0, 0, 0)$. Now all you need is a way to map these orderings to array indices. Literally, that's all you need, because that's what this problem is all about.

Input

Each test case will consist of one line, starting with a single character, either `'m'` or `'u'`. If it is an `'m'` it will be followed by a configuration and you must determine what array index it gets mapped to. If it is a `'u'` then it will be followed by an integer array index i, $0 \leq i < 15504$, and you must determine what configuration gets mapped to it. A line containing the single character `'e'` will end input.

Output

For each test case, output the requested answer – either an array index or a configuration. Follow the format in the examples below.

Sample Input

```
m 0 0 0 0 0 15
u 15503
e
```

Sample Output

```
Case 1: 0
Case 2: 15 0 0 0 0 0
```

A

Problem H: Trees

Source: `trees.{c,cpp,java}`

A graph consists of a set of vertices and edges between pairs of vertices. Two vertices are connected if there is a path (subset of edges) leading from one vertex to another, and a connected component is a maximal subset of vertices that are all connected to each other. A graph consists of one or more connected components.

A tree is a connected component without cycles, but it can also be characterized in other ways. For example, a tree consisting of n vertices has exactly n-1 edges. Also, there is a unique path connecting any pair of vertices in a tree.

Given a graph, report the number of connected components that are also trees.

Input

The input consists of a number of cases. Each case starts with two non-negative integers **`n`** and **`m`**, satisfying **`n ≤ 500`** and **`m ≤ n(n-1)/2`**. This is followed by m lines, each containing two integers specifying the two distinct vertices connected by an edge. No edge will be specified twice (or given again in a different order). The vertices are labelled **`1`** to **`n`**. The end of input is indicated by a line containing **`n = m = 0`**.

Output

For each case, print one of the following lines depending on how many different connected components are trees (T > 1 below):

```
Case x: A forest of T trees.
Case x: There is one tree.
Case x: No trees.
```

`x` is the case number (starting from 1).

Sample Input

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

Sample Output

```
Case 1: A forest of 3 trees.
Case 2: There is one tree.
Case 3: No trees.
```

Problem B: Who wants to live forever?

Digital physics is a set of ideas and hypotheses that revolve around the concept of computable universe. Maybe our universe is just a big program running on a Turing machine? Is the state of the universe finite? Will the life of the universe end? We can only theorize.

In order to help advance the current state of knowledge on digital physics, we ask you to consider a particular model of the universe (which we shall call Bitverse) and determine whether its life comes to a conclusion or continues evolving forever.

Bitverse consists of a single sequence of n bits (zeros or ones). The universe emerges as a particular sequence, in an event called the "Bit Bang", and since then evolves in discrete steps. The rule is simple—to determine the next value of the i-th bit, look at the current value of the bits at positions $i-1$ and $i+1$ (if they exist; otherwise assume them to be 0). If you see exactly one 1, then the next value of the i-th bit is 1, otherwise it is 0. All the bits change at once, so the new values in the next state depend only on the values in the previous state. We consider the universe dead if it contains only zeros.

Given the state of the universe at the Bit Bang, answer the following fundamental question: will Bitverse live forever, or will it eventually die?

Input

The first line of the input contains the number of test cases T. The descriptions of the test cases follow:

Each test case is a string of at least 1 and at most 200 000 characters **0** or **1**.

Output

Print the answers to the test cases in the order in which they appear in the input. For each test case, print **LIVES** if the universe lives forever, and **DIES** otherwise.

Example

Input	Output
3 01 0010100 11011	LIVES DIES LIVES

The first example universe will never become a sequence of zeros (it will continue flipping: 01 10 01 ...). The second one will die in a few steps (0010100 0100010 1010101 0000000). The third one does not change.

C

– LCM Pair Sum –

One of your friends desperately needs your help. He is working with a secret agency and doing some encoding stuffs. As the mission is confidential he does not tell you much about that, he just want you to help him with a special property of a number. This property can be expressed as a function $f(n)$ for a positive integer n. It is defined as:

$$f(n) = \sum_{\substack{1 \le p \le q \le n \\ lcm(p,q)=n}} (p+q)$$

In other words, he needs the sum of all possible pairs whose least common multiple is n. (The least common multiple (LCM) of two numbers p and q is the lowest positive integer which can be perfectly divided by both p and q). For example, there are 5 different pairs having their LCM equal to 6 as (1, 6), (2, 6), (2, 3), (3, 6), (6, 6). So $f(6)$ is calculated as $f(6) = (1+6)+(2+6)+(2+3)+(3+6)+(6+6) = 7+8+5+9+12 = 41$.

Your friend knows you are good at solving this kind of problems, so he asked you to lend a hand. He also does not want to disturb you much, so to assist you he has factorized the number. He thinks it may help you.

INPUT

The first line of input will contain the number of test cases T ($T \le 500$). After that there will be T test cases. Each of the test cases will start with a positive number C ($C \le 15$) denoting the number of prime factors of n. Then there will be C lines each containing two numbers P_i and a_i denoting the prime factor and its power (P_i is a prime between 2 and 1000) and ($1 \le a_i \le 50$). All the primes for an input case will be distinct.

OUTPUT

For each of the test cases produce one line of output denoting the case number and $f(n)$ modulo 1000000007. See the output for sample input for exact formatting.

INPUT EXAMPLE

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

OUTPUT EXAMPLE

```
Case 1: 41
Case 2: 117
Case 3: 16
```

D

Greater New York Programming Contest
Stony Brook University
Stony Brook, NY

event sponsors

A • Hailstone HOTPO

The *hailstone sequence* is formed in the following way:

- If **n** is even, divide it by **2** to get **n′**
- if **n** is odd, multiply it by **3** and add **1** to get **n′**

It is conjectured that for any positive integer number **n**, the sequence will always end in the repeating cycle: `4, 2, 1, 4, 2, 1, …` Suffice to say, when **n == 1**, we will say the sequence has ended.

Write a program to determine the largest value in the sequence for a given **n**.

Input

The first line of input contains a single integer ***P***, (**1 ≤ *P* ≤ 100000**), which is the number of data sets that follow. Each data set should be processed identically and independently.

Each data set consists of a single line of input consisting of two space separated decimal integers. The first integer is the data set number. The second integer is **n**, (**1 ≤ n ≤ 100,000**), which is the starting value.

Output

For each data set there is a single line of output consisting of the data set number, a single space, and the largest value in the sequence starting at and including **n**.

Sample Input	Sample Output
4 1 1 2 3 3 9999 4 100000	1 1 2 16 3 101248 4 100000

Greater New York Programming Contest
Stony Brook University
Stony Brook, NY

event sponsors

This page intentionally left blank.

F

Problem A: Good versus Evil, Who Will Win?

Middle Earth is about to go to war. The forces of good will have many battles with the forces of evil. Different races will certainly be involved. Each race has a certain 'worth' when battling against others. On the side of good we have the following races, with their associated worth:

Hobbits - 1
Men - 2
Elves - 3
Dwarves - 3
Eagles - 4
Wizards - 10

On the side of evil we have:

Orcs - 1
Men - 2
Wargs - 2
Goblins - 2
Uruk Hai - 3
Trolls - 5
Wizards - 10

Although weather, location, supplies and valor play a part in any battle, if you add up the worth of the side of good and compare it with the worth of the side of evil, the side with the larger worth will tend to win.

Thus, given the count of each of the races on the side of good, followed by the count of each of the races on the side of evil, determine which side wins.

1 Input

The first line of input will contain an integer greater than 0 signifying the number of battles to process. Information for each battle will consist of two lines of data as follows.

First, there will be a line containing the count of each race on the side of good. Each entry will be separated by a single space. The values will be ordered as follows: Hobbits, Men, Elves, Dwarves, Eagles, Wizards.

The next line will contain the count of each race on the side of evil in the following order: Orcs, Men, Wargs, Goblins, Uruk Hai, Trolls, Wizards.

All values are non-negative integers. The resulting sum of the worth for each side will not exceed the limit of a 32-bit integer.

2 Output

For each battle, print "`Battle`" followed by a single space, followed by the battle number starting at 1, followed by a "`:`", followed by a single space. Then print "`Good triumphs over Evil`" if good wins. Print "`Evil eradicates all trace of Good`" if evil wins. If there is a tie, then print "`No victor on this battle field`".

Sample Input	**Sample Output**
3 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 10 0 1 1 1 1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0	Battle 1: Evil eradicates all trace of Good Battle 2: Good triumphs over Evil Battle 3: No victor on this battle field

– Bits Equalizer –

You are given two non-empty strings S and T of equal lengths. S contains the characters '0', '1' and '?', whereas T contains '0' and '1' only. Your task is to convert S into T in minimum number of moves. In each move, you can

1. change a '0' in S to '1'
2. change a '?' in S to '0' or '1'
3. swap any two characters in S

As an example, suppose S = "01??00" and T = "001010". We can transform S into T in 3 moves:

- Initially S = "01??00"
- Move 1 – change $S[2]$ to '1'. S becomes "011?00"
- Move 2 – change $S[3]$ to '0'. S becomes "011000"
- Move 3 – swap $S[1]$ with $S[4]$. S becomes "001010"
- S is now equal to T

INPUT

The first line of input is an integer C ($C \leq 200$) that indicates the number of test cases. Each case consists of two lines. The first line is the string S consisting of '0', '1' and '?'. The second line is the string T consisting of '0' and '1'. The lengths of the strings won't be larger than 100.

OUTPUT

For each case, output the case number first followed by the minimum number of moves required to convert S into T. If the transition is impossible, output -1 instead.

INPUT EXAMPLE

```
3
01??00
001010
01
10
110001
000000
```

OUTPUT EXAMPLE

```
Case 1: 3
Case 2: 1
Case 3: -1
```

G

H

J - Joint Venture

Liesbeth and Jan are building a robot for a course project and have discovered that they need to fit two pieces of Lego into an opening.

The opening is x centimetres wide and the sum of the lengths of the two pieces has to be *precisely* equal to the width of the opening, or else the robot will break during the project demonstration, with catastrophic consequences for the grades of the two students.

Photo by Alan Chia

Luckily, Liesbeth and Jan were able to sneak into the physics laboratory late one night to measure the lengths of their remaining Lego pieces very accurately. Now they just need to select two pieces that will fit the opening perfectly.

Input

For each test case, you get:

- a line containing one positive integer: x, denoting the width of the opening in centimetres, with $1 \leq x \leq 20$.
- a line containing one non-negative integer: n, denoting the remaining number of Lego pieces Liesbeth and Jan have access to, with $0 \leq n \leq 1000000$.
- n lines containing positive integers ℓ, denoting lengths of Lego pieces in nanometres. Liesbeth and Jan have told you that no piece of Lego is longer than 10 centimetres, or 100000000 nanometres.

Output

For each test case, a row containing the word '`danger`' if no two pieces of Lego exist that precisely fit into the opening, or '`yes` ℓ_1 ℓ_2', with $\ell_1 \leq \ell_2$, should two such pieces of lengths ℓ_1 and ℓ_2 exist.

In case multiple solutions exist, a solution maximising the size difference $|\ell_1 - \ell_2|$ must be printed.

Example

input	output
1 4 9999998 1 2 9999999	yes 1 9999999

Problem A: Kingdoms

Several kingdoms got into serious financial troubles. For many years, they have been secretly borrowing more and more money from each other. Now, with their liabilities exposed, the crash is inevitable...

There are n kingdoms. For each pair (A, B) of kingdoms, the amount of gold that kingdom A owes to kingdom B is expressed by an integer number d_{AB} (we assume that $d_{BA} = -d_{AB}$). If a kingdom has negative balance (has to pay more than it can receive), it may bankrupt. Bankruptcy removes all liabilities, both positive and negative, as if the kingdom ceased to exist. The next kingdom may then bankrupt, and so on, until all remaining kingdoms are financially stable.

Depending on who falls first, different scenarios may occur—in particular, sometimes only one kingdom might remain. Determine, for every kingdom, whether it can become the only survivor.

Input

The first line of the input contains the number of test cases T. The descriptions of the test cases follow:

The description of each test case starts with a line containing the number of the kingdoms n, $1 \leqslant n \leqslant 20$. Then n lines follow, each containing n space-separated numbers. The j-th number in the i-th line is the number d_{ij} of gold coins that the i-th kingdom owes to the j-th one. You may assume that $d_{ii} = 0$ and $d_{ij} = -d_{ji}$ for every $1 \leqslant i, j \leqslant n$. Also, $|d_{ij}| \leqslant 10^6$ for all possible i, j.

Output

Print the answers to the test cases in the order in which they appear in the input. For each test case, print a single line containing the indices of the kingdoms that can become the sole survivors, in increasing order. If there are no such kingdoms, print a single number **0**.

Example

Input	Output
1 3 0 -3 1 3 0 -2 -1 2 0	1 3

I

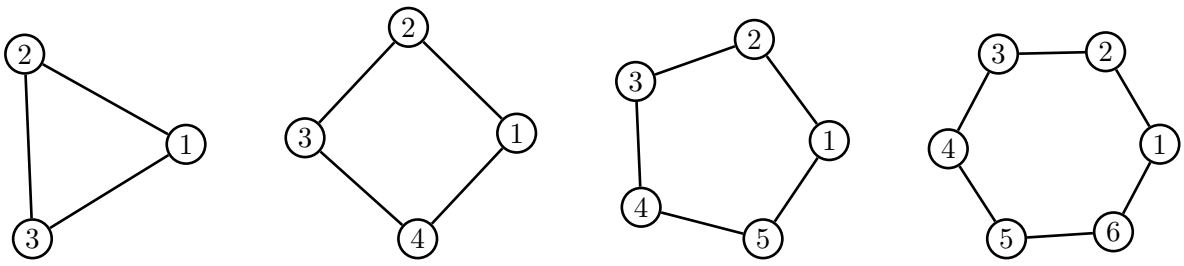
J

E - Edge Case

In graph theory, a *matching* or *independent edge set* in a graph $G = (V, E)$ is a set of edges $M \subseteq E$ such that no two edges in the matching M share a common vertex.

Recently you saw in the news that "The Sveriges Riksbank Prize in Economic Sciences in Memory of Alfred Nobel" (informally, the Nobel Prize in Economics) for 2012 was awarded to Alvin E. Roth and Lloyd S. Shapley for, amongst other things, their algorithm for finding a matching satisfying certain criteria in a bipartite graph. Since you have also heard that matchings in *cycle graphs* have applications in chemistry your thoughts centre around a plan for a beautiful future where your Christmas shopping is more luxurious than ever!

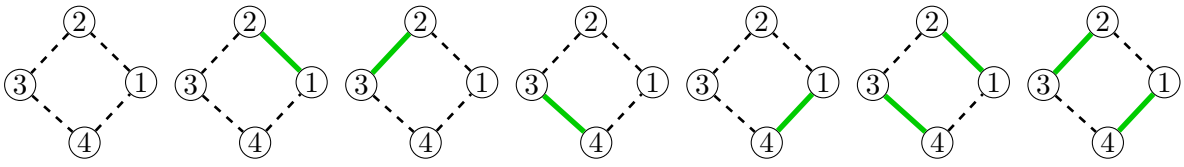
The cycle graph, C_n, $n \geq 3$, is a simple undirected graph, on vertex set $\{1, \ldots, n\}$, with edge set $E(C_n) = \{\{a, b\} \mid |a - b| \equiv 1 \bmod n\}$. It is 2-regular, and contains n edges. The graphs C_3, C_4, C_5, and C_6 are depicted in Figure 1.



(a) C_3 (b) C_4 (c) C_5 (d) C_6

Figure 1: The graphs C_3, C_4, C_5, and C_6.

Your first step towards Nobel Prize fame is to be able to compute the number of matchings in the cycle graph C_n. In Figure 2 the seven matchings of the graph C_4 are depicted.



(a) M_1 (b) M_2 (c) M_3 (d) M_4 (e) M_5 (f) M_6 (g) M_7

Figure 2: The matchings of C_4. The edges that are part of the respective matching are coloured green, while the edges left out of the matching are dashed. $M_1 = \emptyset$, $M_2 = \{\{2, 1\}\}$, $M_3 = \{\{3, 2\}\}$, $M_4 = \{\{4, 3\}\}$, $M_5 = \{\{1, 4\}\}$, $M_6 = \{\{2, 1\}, \{4, 3\}\}$, and $M_7 = \{\{3, 2\}, \{1, 4\}\}$.

Input

For each test case, you get a single line containing one positive integer: n, with $3 \leq n \leq 10000$.

Output

For each test case, a row containing the number of matchings in C_n.

Example

input	output
3 4 100	4 7 792070839848372253127