Problem A. Medium Size Summations

Source file name:	A.c, A.cpp, A.java, A.py
Input:	`Standard`
Output:	`Standard`

R2-D2 (our well known friendly robot) needs to perform some operations quickly to save his space ship. These operations require computing long summations and a division. Moreover, he needs to find the exact solution and he is required to present a report with the results. For that, he needs to simplify his solution as much as possible.

We assume that there is an array available ($X1$, $X2$, ...) of 99999999 elements.

The array has the peculiar property that the average of the first K numbers is equal to the average of the index K and the number 1.

R2-D2 needs to do the following: Given a natural number N less than 99999999, his assignment is to compute the function :

$F(N) = F1(N)/F2(N)$ where:

$F1(N) = N \times \left(\sum_{1 \leq k \leq N} \frac{k^4}{X_k}\right)$

$F2(N) = \left(\sum_{1 \leq k \leq N} \frac{k^3}{X_k}\right) \times \left(\sum_{1 \leq k \leq N} \frac{k^2}{X_k}\right)$

That is, $F1(N) = N \times \left(\frac{1}{X_1} + \frac{16}{X_2} + \ldots + \frac{N^4}{X_N}\right)$ and

$F2(N) = \left(\frac{1}{X_1} + \frac{8}{X_2} + \ldots + \frac{N^3}{X_N}\right) \times \left(\frac{1}{X_1} + \frac{4}{X_2} + \ldots + \frac{N^2}{X_N}\right)$

Since R2-D2 needs an exact solution, we ask him to report the following: The solution needs to be given as a pair of relative prime numbers a,b such that $F(N) = a/b$ if the solution is not an exact integer. Otherwise just give the exact integer. The numbers processed by R2-D2 were of eight digits (99999999). Remember that R2-D2 was built long long time ago. His circuits are not that fast but he is clever. R2-D2 was able to perform one of these operations in less than one second. Can you do this assignment as fast as R2-D2 did it?

Input

You will receive an input line with natural numbers, one per line. Each number is less than 99999999. You will receive no more than 20 numbers.

Output

You need to give a sequence of lines each one with the solution of the corresponding input case. The solution is either a pair of natural numbers separated by the symbol / representing the pair a,b mentioned above (when the division is not exact) or just one natural number (when the division is exact). Notice that these numbers could require more than 8 digits.

Example

Input	Output

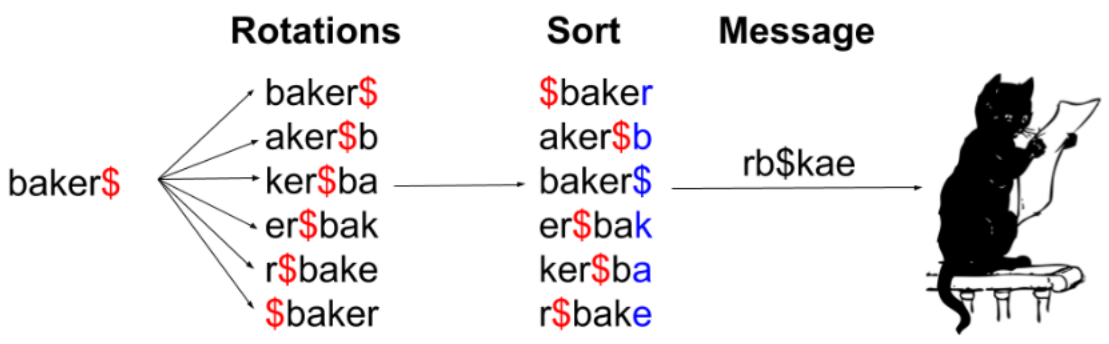
`1`
`2`
	`1`
	`6/5`

Problem B. Baker$

Source file name:	B.c, B.cpp, B.java, B.py
Input:	`Standard`
Output:	`Standard`

Baker and Iris are really good friends. Last month Iris moved out of the city and now their only way to communicate is using letters. As both Baker and Iris are cats, their each of their letters contains a single string of N characters. In order to create a more secure way of communication, Iris decided that she will encrypt the messages that sends to Baker using the following procedure:

First add the character $ at the end of the string, next she will create all rotations of the string in lexicographical order.Consider the character $ is the lowest lexicographically. The encrypted string is made taking the last character of each rotation.



Your task is to help Baker to decrypt the letter Iris has sent.

Input

The input consists of several test cases. Each test case consists of a line with a string S that contains only lower case characters and the $ symbol.

- $1 \leq |S| \leq 1000$

Output

For each test case print in one line the decrypted message.

Example

Input	Output
`rb$kae` `annb$aa`	`baker` `banana`

Problem C. Binomial Powers

Source file name:	C.c, C.cpp, C.java, C.py
Input:	`Standard`
Output:	`Standard`

In algebra, one of the classical examples of algebraic expansions is the binomial power $(x + y)^n$, where $n \geq 0$ is an integer.

Here are a couple examples of these expansions:

$(x + y)^2 = x^2 + 2xy + y^2$

$(x + y)^3 = x^3 + 3x^2y + 3xy^2 + y^3$

As you can see, the number of terms in the expansion is always $n + 1$ and the exponents on x decrease from n to 0 while the exponents on y increase from 0 to n along these terms.

The only thing that is not so obvious is the values of each term's coefficient. An interesting result from algebra is that coefficients are very easy to calculate.

If you sort the terms in the ordering described above (x exponents decreasing), the value of the k-th term's coefficient is ${}_nC_k$, which is usually called "n choose k" and is defined as follows:

${}_nC_k = \frac{n!}{(k!(n-k)!)}$

Fortunately, a recursive definition exists for the binomial coefficients and is calculated as follows:

For n, $k > 0$; ${}_nC_k =_{(n-1)} C_{(k-1)} +_{(n-1)} C_k$

For $n \geq 0$; ${}_nC_0 = 1$; ${}_nC_n = 1$

Your job here is to expand a bunch of binomial powers as explained.

Input

The input starts with a line with a single integer T, followed by T test cases. Each test case is written in a single line and contains a single integer n.

- $1 \leq T \leq 100$
- $0 \leq n \leq 100$

Output

For each test case you need to calculate the expansion of $(x + y)$.

The output must be written as in the sample output, with exponents written after a '^' symbol and no spaces between any characters. Terms must be separated by a '+' character. Also remember that the terms must be sorted as explained before.

If a variable in a term has exponent 1, you must not write the exponent nor the '^' symbol. If the exponent is 0 you must also not write the variable itself. Coefficient should only be written if it is different from 1, or if both variables have exponents 0.

Examples:

- $3x^2y^0$ should be printed as 3x^2
- $1x^3y^1$ should be printed as x^3y
- $2x^0y^2$ should be printed as 2y^2

- $1x^0y^0$ should be printed as 1

Example

Input	Output
2 2 3	x^2+2xy+y^2 x^3+3x^2y+3xy^2+y^3

Problem D. Digital roots

Source file name:	D.c, D.cpp, D.java, D.py
Input:	`Standard`
Output:	`Standard`

The digital root of a positive integer is found by summing the digits of the integer. If the resulting value is a single digit then that digit is the digital root. If the resulting value contains two or more digits, those digits are summed and the process is repeated. This is continued as long as necessary to obtain a single digit.

For example, consider the positive integer 24. Adding the 2 and the 4 yields a value of 6. Since 6 is a single digit, 6 is the digital root of 24. Now consider the positive integer 39. Adding the 3 and the 9 yields 12. Since 12 is not a single digit, the process must be repeated. Adding the 1 and the 2 yields 3, a single digit and also the digital root of 39.

Input

The input file contains several test cases. Each test case contains a line with a single positive integer number N. The end of the input will be in a case where $N = 0$, this case should not be processed.

- $1 \leq N < 2^{64}$

Output

For each test case in the input, print a single line with the digital root of N.

Example

Input	Output
24 39 0	6 3

Problem E. Keeping thins in order

Source file name:	E.c, E.cpp, E.java, E.py
Input:	`Standard`
Output:	`Standard`

In this problem, you are given a series of lists containing words and numbers. Your task consists on sorting them so that the words are in alphabetical order, and the numbers in numerical order. There is one catch though: if the k-th element of the list is a number, it must remain a number in the sorted list.

Input

The input will contain multiple lists, one per line, there will be no more than 10 lists in the input. Each element will be separated by a comma followed by a space, and the list will be terminated by a period. No list will contain more than 100 elements. The input will be terminated by a line containing only a period.

Output

For each list in the input, output the sorted list, separating each element of the list with a comma followed by a space, and ending the list with a period.

Example

Input

```
0.
banana, strawberry, OrAnGe.
Banana, StRaWbErRy, orange.
10, 8, 6, 4, 2, 0.
x, 30, -20, z, 1000, 1, Y.
50, 7, kitten, puppy, 2, orangutan, 52, -100, bird, worm, 7, beetle.
.
```

Output

```
0.
banana, OrAnGe, strawberry.
Banana, orange, StRaWbErRy.
0, 2, 4, 6, 8, 10.
x, -20, 1, Y, 30, 1000, z.
-100, 2, beetle, bird, 7, kitten, 7, 50, orangutan, puppy, 52, worm.
```

Problem F. Teleporters

Source file name:	F.c, F.cpp, F.java, F.py
Input:	`Standard`
Output:	`Standard`

Mankind has evolved into a pretty awesome state. We have flying cars. We now live in planets other than Earth. But many people's favorite invention is the teleporter. Teleporters are way too expensive to be owned by every single planet. However, there are some rich planets which do have them.

The problem with the current teleporting technology is that they only work in pairs. In other words, one teleporter can only teleport people to ONE specified exit. They are bidirectional, so an "entry" teleporter can also work as an "exit" teleporter. Traveling between teleporters takes no time.

Your job as a travel planner is to come up with the best route between planets. The best route is the one that takes the less amount of time. Obviously, you can use teleporters to do this.

Input

First you will be given an integer N, the number of test cases. N maps, for which you have to calculate best routes, will follow.

For each of the N maps, the first line contains two integers, $1 \leq V \leq 50$, and $T \geq 0$, where V is the number of planets in the system, and T is the number of pairs of teleporters.

After these 2 numbers, V lines will follow. The first integer in these lines represents a planet, and the next pairs of numbers represent planets which can be reached from the current one, as well as the time it takes to reach them. Note that even if there is a path from planet A to planet B, that does not necessarily mean there is a path from B to A. The numbering of the planets starts with 0.

After that, there is a line with $T * 2$ integers, which represent the teleporters. These should be read pairwise, that is, if the line looks like this:

0 1 2 3

that means there is a teleporter pair between planets 0 and 1, and another pair between planets 2 and 3. If T is zero, then this line is omitted.

Finally, there is a line with two integers, the first represents the starting planet, and the second the goal.

Output

If there is a path from the starting planet to the goal planet, print one line with the format "Case #X: time" (X starts in 1), followed by a line containing all the planets visited, in order. Each of these planets will be separated by a single whitespace.

In case there exists no path, print "Case #X: Path does not exist", and omit the next line.

Note that the minimum path will always be unique.

Also, teleporter pairs can only be used once. So, if you already used a teleporter going from planet A to B, then you cannot use that same teleporter again, not even if going from B to A.

Example

Input

```
3
4 2
0 1 10
1 2 10
2 3 10
3 0 10
3 2 2 0
1 0
3 0
0 1 10 2 20
1
2 1 2
0 2
3 0
0 1 10 2 20
1
2 1 2
1 2
```

Output

```
Case #1: 10
1 2 0
Case #2: 20
0 2
Case #3: Path does not exist
```

Problem G. Prime gap

Source file name:	G.c, G.cpp, G.java, G.py
Input:	`Standard`
Output:	`Standard`

A prime gap is the difference between two successive prime numbers. The n-th prime gap, denoted $g(n)$ is the difference between the $(n+1)$-th and the n-th prime numbers, i.e.

$g(n) = p_{n+1} - p_n$

For the first five prime numbers (2,3,5,7,11) We have $g(1) = 1$, $g(2) = 2$ and $g(4) = 4$ the sequence of prime gaps has been extensively studied.

The first 30 prime gaps are :

$1, 2, 2, 4, 2, 4, 2, 4, 6, 2, 6, 4, 2, 4, 6, 6, 2, 6, 4, 2, 6, 4, 6, 8, 4, 2, 4, 2, 4, 14$

In this problem your task is to provide the maximum prime gap between two arbitrary numbers a and b.

Input

The input consist in several cases no more than 100, each case consist of two positive integers a and b, The cases will end with a line with the integers 0 0.

- $1 \leq a, b \leq 1300000$

Output

For each case you must print the maximum prime gap inside the range defined by the two positive integers inclusive, or NULL if such gap does not exist.

Example

Input	Output

1 1
6 12
1 30
0 0
	NULL
	4
	6

Problem H. In a hurry

Source file name:	H.c, H.cpp, H.java, H.py
Input:	`Standard`
Output:	`Standard`

David hates to wait at stop signs, yield signs and traffic signals while driving. To minimize this aggravation, he has prepared maps of the various regions in which he frequently drives, and measured the average delay (in seconds) at each of the various intersections in these regions. He wants to find the routes between specified points in these regions which minimize his delay at intersections (regardless of the total distance he has to drive to avoid delays), and has enlisted your assistance in this effort.

Input

For each region, David provides you with a map. The map data first identifies some number of intersections, NI. The regions never include more than 10 intersections. The intersections in each region are numbered sequentially, starting with the number one (1). For each intersection, in turn, the input then specifies the number of streets leading away from the intersection, and for each such street, the number of the intersection to which the street leads, and the average delay, in seconds, that David encounters at that intersection. Following the data for the last intersection in a region there appear the numbers associated with the intersections where David wants to start and end his drive. The entire input consists of a sequence of maps, followed by the single integer zero (0). Note: between each (node number, cost) pair there are 3 white spaces.

Output

For each region, in order, print a single line of output which contains the region number (they, too, are sequentially numbered, starting with 1), a list of the intersection numbers David will encounter in the route with minimum average delay, and the average number of seconds he will be delayed while travelling this route. A suitable format is shown in the example below.

Notes:

1. There will always be a unique route with the minimum average delay in each region.
2. A street from intersection I to intersection J is one-way. To represent a two-way street from I to J, the map must also include a route from intersection J to intersection I.
3. There will never be more than one route directly from intersection I to intersection J.

Example

Input

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

2
1   2 5
1   1 6
1 2

7
4   2 5   3 13   4 8   5 18
2   3 7   6 14
1   6 6
2   3 5   5 9
3   6 2   7 9   4 6
1   7 2
0
1 7

0
```

Output

```
Case 1: Path = 2 1 4; 8 second delay
Case 2: Path = 1 2; 5 second delay
Case 3: Path = 1 2 3 6 7; 20 second delay
```

Problem I. Intersecting line segments

Source file name:	I.c, I.cpp, I.java, I.py
Input:	`Standard`
Output:	`Standard`

In a 2-D Cartesian space, a straight line segment A is defined by two points $A_0 = (x_0, y_0)$, $A_1 = (x_1, y_1)$. The intersection of line segments A and B (if there is one), together with the initial four points, defines four new line segments. In Figure 1.1, the intersection P between lines B and C defines four new segments. As a result, the total amount of line segments after the evaluation of intersections is five.

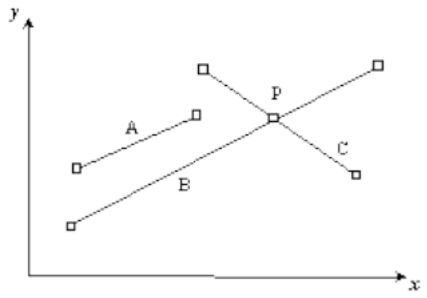


Figure 1.1 - Intersections of line segments

Given an initial set of lines segments, determine the number of line segments resulting from the evaluation of all the possible intersections. It is assumed, as a simplification, that no coincidences may occur between coordinates of singular points (intersections or end points).

Input

The input begins with a single positive integer on a line by itself indicating the number of test cases following, each of them as described below. This line is followed by a blank line, and there is also a blank line between two consecutive test cases.

For each test case the first line contains the integer number N of line segments. Each of the following N lines contains four integer values x_0, y_0, x_1, y_1, separated by a single space, that define a line segment.

- $1 \le N \le 20000$
- $0 \le x_0, y_0, x_1, y_1 \le 10^6$

Output

For each test case, the output must follow the description below. The outputs of two consecutive cases will be separated by a blank line. The integer number of lines segments after all the possible intersections are evaluated.

Example

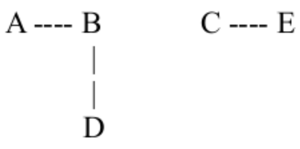
Input	Output

2
 	
5
3 1 3 8
4 1 4 8
2 4 9 4
8 7 5 7
5 6 10 1
 	
2
2 4 4 9
2 6 5 4
	11
	
	4

Problem J. Connected Components

Source file name:	J.c, J.cpp, J.java, J.py
Input:	`Standard`
Output:	`Standard`

A graph $G = (V, E)$ is connected if a path can be found in 0 or more intermediate steps between any pair of nodes in G. The following graph is not connected:



It contains, however, many connected subgraphs, which are: $\{A\}$, $\{B\}$, $\{C\}$, $\{D\}$, $\{E\}$, $\{A, B\}$, $\{B, D\}$, $\{C, E\}$, $\{A, B, D\}$.

A connected subgraph is maximal if there are no nodes and edges in the original graph that could be added to the subgraph and still leave it connected. In the previous example, $\{C, E\}$ and $\{A, B, D\}$ are maximal.

Your job is to find, for the given graphs, how many connected subsets there are.

Input

There will be many input cases, each of which will be separated by a blank line. The last input case will be followed by EOF.

Each of the input cases starts with a line that contains a single upper case alphabetic character, which represents the largest node name in the graph. Each successive line contains a pair of upper case alphabetic characters denoting an edge in the graph. The graph represented by the input is undirected.

Output

For each test case print a single line with the number of maximal connected subgraphs.

Example

Input	Output
E	2
AB	1
CE	3
DB	
B	
AB	
C	

Problem K. Typo sequences

Source file name:	K.c, K.cpp, K.java, K.py
Input:	`Standard`
Output:	`Standard`

As part of a study about typos on modern smartphone keyboards, a volunteer is given a list of words and he is asked to type them all without correcting if they make a mistake. You've been hired to produce a program to analyze the results and determine which kinds of typos are more common.

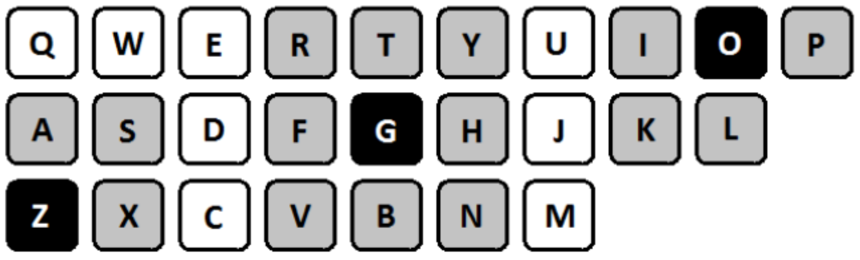
You're given a list of pairs of words, a reference word (the word the volunteer was supposed to type) and an output word (the word the volunteer actually typed).

For each pair of words, your job is to determine all the possible sequences of typos that the volunteer may have followed to produce the output word while attempting to type the reference word.

The volunteer attempted to type each letter of the reference word just once, and for each of those letters there are 5 possible outcomes when typing it:

1. OK (o): The volunteer typed the correct letter.
2. FORGOT (f): The volunteer forgot to type the letter.
   e.g. forgot to type an 'l' in 'hello', producing 'helo'.
3. WRONG (w): The volunteer typed an "adjacent" key instead of the correct one.
   e.g. typed 'r' instead of 'e' in 'hello' and produced 'hrllo'.
4. EXTRA ON LEFT (l): The volunteer pressed the correct key and an "adjacent" key together and the wrong letter was typed first.
   e.g. pressed keys for 'l' and 'i' when typing 'l' in 'hello', producing 'heillo'.
5. EXTRA ON RIGHT (r): The volunteer pressed the correct key and an "adjacent" key together and the correct letter was typed first.
   e.g. pressed keys for 'h' and 'y' when typing 'h' in 'hello', producing 'hyello'.

A key is "adjacent" to another if they share a border or a corner. See the image below.



Adjacent keys of 'z', 'g' and 'o'

As an example, if the reference word is 'normal' and the output word is 'nrtmsap', there is only one sequence of the above typos that could produce that output word ofrolw, because:

- 'n' is ok (o)
- 'o' is missing (f)

- 'r' followed by its adjacent 't' (r)
- 'm' is ok (o)
- 'a' preceded by its adjacent 's' (l)
- 'l' replaced by its adjacent 'p' (w)

There may be more than one sequence of typos that lead to the same output word.

As an example of such scenario if while typing 'allok', the volunteer produced 'allok' (yes, the same word, sequence ooooo), there can also be a sequence of typos that happen to produce the same word. ofowl, for example:

- 'a' is ok (o)
- 'l' is missing (f)
- 'l' is ok (o)
- 'o' replaced by its adjacent 'l' (w)
- 'k' preceded by its adjacent 'o' (l)

Input

The first line will only have a single positive integer K, smaller than 50. Then follow K pairs of words (each word in its own line), the first being the reference word and the second being the output word. Look at the sample.

The words are only formed by at most 15 lower case letters and there's always a sequence of typos that, when followed while attempting to type the reference word, can produce the output word.

While the reference word must have at least 1 character, the output word may be empty (since you can always forget to type all the letters).

Output

The output of each test case must be followed by a blank line. The output of a test case starts with a header on a single line, which consists of the reference word, followed by the output word, separated only by a single space, and finished with a colon (:). Look at the sample.

After the header, list on independent lines each possible sequence of typos that the volunteer could have made to type the output word while attempting to type the reference word.

Sequences must be ordered lexicographically.

Example

Input

```
3
allok
allok
empty

normal
nrtmsap
```

Output

```
allok allok:
ofolo
ofowl
ooflo
oofwl
ooofl
ooooo
ooorf
oorfo
oorwf

empty :
fffff

normal nrtmsap:
ofrolw
```