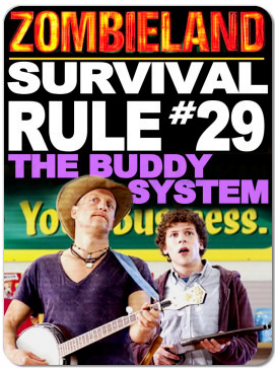
Problem A Input file: a.in Output: monitor/stdout/cout/System.out

Always Follow the Rules in Zombieland



Welcome to Zombieland! For the next five hours, your team will battle wits with hordes of the undead. Do you have the cunning, will, and stamina to survive? Follow the rules, and you may live to tell the tale.

In Zombieland, there are a fundamental set of rules you must follow to stay alive. Rule 8 is "Get a kick&@% partner", rule 18 is "Limber up", rule 29 is "The buddy system", and rule 22 is, "When in doubt, know your way out".

If you intend to survive for long in Zombieland, you'll need to know which rule number goes with which quote. Write a program to display the correct quote given the rule number.

Have fun during your stay in Zombieland, and remember rule 32, "Enjoy the little things".

Input

Input will begin with an integer `q`, $0 < q \leq 50$, on its own line signifying the number of quotes. On the following lines will be the quotes, one per line. No quote will be greater than 65 characters. The first quote will be rule 1, the second quote rule 2, etc. Following these quotes will be an integer `r`, $0 < r \leq 50$ on a line of its own signifying the number of rules to look up quotes for, followed by a sequence of `r` rule numbers, one per line.

Output

For each rule number, display "`Rule`", followed by a single space, then the rule number, then a "`:`" and a single space followed by the appropriate quote. If a rule number falls outside the range of quotes, display the message "`No such rule`", instead of a quote.

Sample Input	Sample Output
4 Cardio The double tap Beware of bathrooms Seatbelts 5 2 -10 5 1 4	Rule 2: The double tap Rule -10: No such rule Rule 5: No such rule Rule 1: Cardio Rule 4: Seatbelts

Problem B Input file: b.in Output: monitor/stdout/cout/System.out

Zombie Containment

Associated with the zombie apocalypse is the notion of a critical mass of zombies: once the number of zombies exceeds a certain threshold T, all is lost. Your city is in a precarious state: its current population is $3T$. Even worse, you suspect someone in the city may be infected, although you have no idea who it may be. Once the symptoms manifest, it will be too late for that person, as well as for everyone else in the city that person can reach.

To prevent the possibility of reaching critical mass, you are going to divide the city into 3 regions via a set of walls. The city is triangular, and so you are going to divide the city into 3 smaller triangles by choosing a single splitting point and erecting straight walls between that point and the three vertices of the outer wall. This must be done immediately, i.e., there is no time to relocate people. Can you find a splitting point such that the three resulting regions each have exactly T people?

Input

The input consists of multiple test cases. Each test case is formatted as follows:

- Line 1: An integer n, $3 \le n \le 30000$, denoting the number of people. n is always a multiple of 3.
- Lines 2 to 4: Two numbers x_i and y_i denoting the coordinates of the i^{th} corner of the city, with $-10 \le x_i, y_i \le 10$. The three corners are specified in counterclockwise order.
- Lines 5 to $n + 4$: Two numbers x_i and y_i denoting the coordinates of the i^{th} person, with $-10 \le x_i, y_i \le 10$. All people lie in the interior of the city triangle.

Input is followed by a single line with a 0, which should not be processed.

It is guaranteed that no two people will be collinear with any of the corners of the city. In particular, it is guaranteed that for any two people I and J and any corner A, the angle IAJ will be at least 10^{-7} rad.

Output

For each test case, print out a single line with two numbers x and y denoting the splitting point of the city. Each person must lie in the interior of one of the three generated triangles. Assume a person is a single point with zero radius, and that a wall is a line with zero thickness. It is guaranteed that there exists a splitting point; in fact, it is guaranteed that there exists a splitting point such that if the point were moved by 10^{-7} in any direction, it would still be a splitting point. Any splitting point that partitions the people into 3 sets of equal size will be judged correct.

Sample Input	Sample Output
3 0.0 0.0 10.0 0.0 0.0 10.0 4.0 4.0 1.0 4.0 4.0 1.0 0	3.0 3.0

Problem C Input file: c.in Output: monitor/stdout/cout/System.out

Mesmerization of The Count

"Sesame Street's" The Count has gone bad. He has escaped the hard-scrabble New York streets where he gained his fame and now haunts Oregon, surprising and killing unwary programmers throughout the state. Only one defense has been found: he is mesmerized by equations of the form

```
(a/b)^3 + (c/b)^3 = n
```

for natural numbers `a`, `b`, `c`, and `n`. For instance,

```
(415280564497/348671682660)^3 + (676702467503/348671682660)^3 = 9
```

stopped him in his tracks for a good 20 minutes, giving mathematician Henry Dudeney enough time to escape a certain death. Unfortunately, large numbers like this are too hard to memorize, so shorter ones such as

```
(2/1)^3 + (1/1)^3 = 9
```

are better. Each such equation is only good for one mesmerization, so your job is to write a program that will generate new such equations. In particular, you will be given `n`, and your job is to generate values for the natural numbers `a`, `b`, `c` that satisfy the first equation. When multiple solutions exist, you should report the one with the minimum possible sum `a + b + c + b`, such that the value of `a/b` is greater than or equal to the value of `c/b`. You may assume such a solution is unique. If you cannot find three natural numbers `a`, `b`, `c` such that `a + b + c + b` is less than 4,000, you should print "`No value.`"

Input

The input will be a sequence of lines; each line will contain a single natural number less than 10,000. Input is terminated with a `0`, which should not be processed.

Output

For any valid equation you find, print the equation with the appropriate values for `a`, `b`, and `c`. A single space should precede and follow the + and the = in the equation. When no valid equation exists, print "`No value.`"

Sample input	Sample output
1 9 7 6000 0	No value. (2/1)^3 + (1/1)^3 = 9 (5/3)^3 + (4/3)^3 = 7 (370/21)^3 + (170/21)^3 = 6000

Problem D Input file: d.in Output: monitor/stdout/cout/System.out

Zombie Swallows

In the movie, Monty Python and the Holy Grail, a very crucial scene centered around the question "What is the airspeed velocity of an unladen swallow?" Perhaps the more important question for an undead ornithologist, "What is the swallowing capacity of a zombie swallow?"

It turns out that in order to control a zombie swallow, you must control what the zombie swallow swallows. After rising from its grave, a zombie swallow has an empty stomach. To keep a zombie swallow alive, you must feed it immediately. Furthermore, each zombie swallow must swallow enough insects to meet its minimum energy requirements, yet at the same time, no more insects than can fit inside its stomach. Thus, when presented with a set of insects to feed upon, the swallow does its best to choose which insects it will swallow so that it meets its minimum energy requirement of C_{mini} micrograms of insects and does not exceed its stomach capacity of C_{maxi} micrograms of insects. If a swallow finds enough micrograms of insects to meet these requirements, the swallow will survive to swallow another day. Whenever a swallow starts to feed, it attempts to find a subset of insects that are as close to the midrange of its feeding requirements as possible.

Input

The first line of input will be S, the number of swallows that need to swallow insects. This will be followed by S lines where $S \leq 30$, and each of these lines will contain the feeding information for a single swallow:

- The first two integers $0 \leq C_{mini} < C_{maxi} \leq 2^{26}$ respectively represent the swallow's minimum energy requirements and maximum feeding capacity.
- An integer n_i where $0 \leq n_i \leq 150$ denotes the number of insects available to the swallow.
- Finally, a list of n_i positive integer weights $\leq 2^{26}$, where each weight (micrograms) is the weight of a specific insect.

Because of the characteristics of zombie swallows, $1 \leq (C_{maxi} / (C_{maxi} - C_{mini})) \leq 60{,}000$.

Output

For each swallow, your program should determine whether or not that particular swallow's feeding requirements are achievable. If the swallow can feed according to its requirements (given the insects it is allowed to choose from), the program should output "`Sallow swallow swallows.`" However, if there is no combination of insects that satisfy the eating restrictions for that swallow, the program should output "`Sallow swallow wallows in dust.`", i.e., the swallow would be underfed or overfed regardless of its choice of insects to swallow.

Sample Input	Sample Output
2 8 11 2 3 5 299 300 9 1 2 3 4 5 60 130 260 270	Sallow swallow swallows. Sallow swallow wallows in dust.

Decomposing Fibonacci Numbers

Some Fibonacci numbers are immune to zombie attack — as prime numbers they can't be decomposed.

Fibonacci numbers are defined by the following recurrence:

$$F(n) = \begin{pmatrix} n = 0: \ 0 \\ n = 1: \ 1 \\ n > 1: \ F(n-1) + F(n-2) \end{pmatrix}$$

You will be given an indefinite number of integer ranges of numbers that can be represented as 64-bit signed integers. Your job is to report in increasing order the Fibonacci numbers that fall within that range, as well as their base-2 logarithm* and their *prime decomposition* — the prime numbers in increasing order which, when multiplied together, give the value of the Fibonacci number. If there is no Fibonacci number in the range, report that fact.

A reminder: the logarithm of zero is undefined, even though zero is the first Fibonacci number. Also note that, by definition, 0 and 1 have no prime factors, even though they are Fibonacci numbers.

Input

The input file contains an indeterminate number of lines consisting of two non-negative integers (`lo` and `hi`) separated by one space, given in hexadecimal format (as in `0x1a` meaning 26 in decimal). Each integer is guaranteed to fit within a 64-bit signed integer. The program terminates when it either encounters an end-of-file condition or when `lo` $\geq$ `hi`.

Output

For each range in the input file, print the range and the Fibonacci number information as shown in the sample output, with each range separated by a blank line. Note that the base-2 logarithm (`lg`) is reported with six digits to the right of the decimal point, and that the prime factors are separated by single spaces.

* Reminder: to calculate the base c logarithm, note that $\log_c(x) = \log(x) / \log(c)$, using on the right-hand side your favorite logarithm (common logarithm or natural logarithm).

Sample Input	**Sample Output**
0x0 0x8 0x9 0xc 0x9 0x40 0x0 0x0	Range 0 to 8: Fib(0) = 0, lg does not exist No prime factors Fib(1) = 1, lg is 0.000000 No prime factors Fib(2) = 1, lg is 0.000000 No prime factors Fib(3) = 2, lg is 1.000000 Prime factors: 2 Fib(4) = 3, lg is 1.584963 Prime factors: 3 Fib(5) = 5, lg is 2.321928 Prime factors: 5 Fib(6) = 8, lg is 3.000000 Prime factors: 2 2 2 Range 9 to 12: No Fibonacci numbers in the range Range 9 to 64: Fib(7) = 13, lg is 3.700440 Prime factors: 13 Fib(8) = 21, lg is 4.392317 Prime factors: 3 7 Fib(9) = 34, lg is 5.087463 Prime factors: 2 17 Fib(10) = 55, lg is 5.781360 Prime factors: 5 11

Pride and Prejudice and Zombies

Pride and Prejudice and Zombies, by Jane Austen and Seth Grahame-Smith, spices up the famous 1813 satire about marriage and social convention with intercalary appearances by zombies, skunks, ghouls, chipmunks and ninjas. Most literary critics praised PPZ (as it's known on Facebook) as "clever", "insightful", holding "indomitable appeal", although Macy Halford of The New Yorker condemned Mr. Grahame-Smith's retelling as "awful", and "one hundred per cent terrible".

Here's a little known fact: the book's release was delayed by some 8 years due to several heated disagreements between Mr. Grahame-Smith and the editors at Quirk Books, the small publishing house in Philadelphia that eventually published the novel. The most impassioned of the arguments centered on the controversial "Vampire Number" chapter, where Mrs. Bennett — desperate to rally the English countryside's interest in matrimony — contrives a social event where 50 men and 50 women draw slips of paper from a large, feathery hat. "It is an axiom generally acknowledged, that a single man in possession of the complementary multiplicand, must be in want of a wife", says Mrs. Bennett in announcing the event. On each slip of paper is a single three-digit number. The goal of the event was for each of the women to find the man whose three-digit number, when multiplied by her own, produced a six-digit number that reproduced the digits of the two "multiplicands" (Mrs. Bennett's word) with the correct "distribution" (also her word) in some order.

Mr. Grahame-Smith insisted the chapter was vital to the story arc, whereas Quirk Books considered it forced, obscure, boring, and unnecessarily intellectual. Furthermore, Quirk Books was initially contemplating a limited New England release and argued that no one outside of the Pacific Northwest understood math. Mr. Grahame-Smith eventually conceded this to be true, and agreed to cut the chapter. The chapter's existence only surfaced recently in a follow-up article in The New Yorker where Ms. Halford noted the novel would have "benefitted substantially" had it been included.

Vampire Numbers

It is generally acknowledged that Vampire numbers are positive integers with an even number — we'll say $2n$ — of digits where the $2n$ digits can be distributed across two n-digit numbers such that their product equals the original $2n$-digit number. None of the three numbers can include leading zeroes, and neither of the two n-digit numbers can have consecutive zeroes anywhere.

For examples:

- 125460 = 204 x 615 (so 125460 is a Vampire number).
- 16758243290880 = 1982736 x 8452080 (so 16758243290880 is a Vampire number).
- 353 has an odd number of digits, so it can't be a Vampire number by definition.
- 3421 can't be subdivided properly, so it's not a Vampire number either.

Write a program that reads in a series of numbers (each at most 18 digits) and prints whether or not that number is a Vampire number.

Input

There will be an arbitrary number of inputs, one per line, with no leading zeroes or extraneous whitespace. Each number will have at most 18 digits. End of input is marked by a single `0` on its own line, for which no output should be produced.

Output

For each input, print the number, followed by a colon, followed by a space, followed by "`yes`" if the number is a Vampire number or "`no`" if it is not.

Sample Input	Sample Output
1260 6880 8680 102510 108135 110758 115672 116725 125248 12054060 13078260 46847902 46847921 1001795850 315987404670 472812953760 10174695862032 10174695862037 2512099504480801 8186379410403769 170147428389340249 189598345243224241 968781726110944201 968781726110944203 698781726110944201 0	1260: yes 6880: yes 8680: no 102510: yes 108135: yes 110758: yes 115672: yes 116725: yes 125248: yes 12054060: yes 13078260: yes 46847902: no 46847921: no 1001795850: yes 315987404670: yes 472812953760: yes 10174695862032: yes 10174695862037: no 2512099504480801: yes 8186379410403769: yes 170147428389340249: yes 189598345243224241: yes 968781726110944201: yes 968781726110944203: no 698781726110944201: no

Tales from DeCrypt

In newsgroups, lists, and other ways of publicly sharing information, one popular method of obscuring information without actually hiding it has been the ROT13 algorithm: alphabetic characters are simply rotated by 13 positions (modulo 26), so that the encryption and decryption algorithms are identical. Messages that are potentially offensive to some readers of the newsgroup or list are purposely posted in ROT13 form, on the theory that the **reader** is responsible for changing the offensive material into clear-text form, and so that the reader cannot complain about it.

We can use a rotational cipher to hide information as well as obscure it. You are responsible for generating the decryption algorithm for the encryption algorithm described here. It is restricted to 7-bit ASCII/ANSI characters, and we will deal only with the printing characters — 0x20 (space) up to and including 0x7e ('~') for 95 characters. This way the encrypted text can still be dealt with as pure text for file manipulation and transmission purposes.

Encryption Algorithm

Select three numbers to encode all the necessary information for this linear congruential random number generator: the multiplier (`a`), the modulus (`m`), and the seed (`s`):

```
double r(in int a, in int m, inout int s)
   double val = s modulo m / double(m)
   s = ( a * s + 1 ) modulo m
   return val
```

The two integers for the random number generator and the initial seed (`s` in the pseudo code above) are contained within the file as the first line, which contains three white-space delimited 32-bit integers, given in the order "`a m s`". Range: $2 \leq$ number ≤ 65536. So a first line of "`12343 65536 11`" generates

- `a = 12343`
- `m = 65536`
- `s = 11`

The output file of the encryption program contains these three numbers, white-space delimited, on one line. The encrypted text begins on the next line. This constitutes the input to your decryption program.

Encrypted text is generated by the following algorithm:

```
for each character c in the input stream
    if the character is not in the range 0x20 through 0x7e
        pass it through to the output
    otherwise
        c = ( (c - 32) + ceiling( 95 - r(a,m,seed)*95) ) ) modulo 95 + 32
        send c to the output
```

Input

The input for your program is the output of the encryption program: three white-space delimited numbers on one line. The encrypted text begins on the next line, and continues to the end of file.

Output

Your output is the decryption of the text encrypted in the input file.

Sample Input	**Sample Output**
12343 65536 11 a[+d'x/vKmV<WP(+2:N]%CN+u#rjNQB vW'ecvzcK5E%F;^Qlo~pt\]kwGr*.yv| So|#p36LhPNM#"N<I|}2c[cGX5I3o!u m48rOK1&N=&8%Q-2Jq^[v&r;at"z#'C	aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa bbbbbbbbbbbbbbbbbbbbbbbbbbbbbb ccccccccccccccccccccccccccccc ddddddddddddddddddddddddddddd

A Smart Brain is a Tasty Brain

The zombies have cornered you and your team. There's no hope…

But wait! Suddenly, they all stop advancing and offer you a deal instead. As it turns out, the zombies are having a problem finding good tasting brains, which, to a zombie, are the smart brains (there is nothing more disgusting to a zombie than the brains of a communications major). While your brains would be really delicious, the zombies realize that they could be put to better use by helping them to find other smart brains. The deal is this: help the zombies determine whether a given brain is smart (and therefore tasty) or not, and they will let your team go (at least for now).

Being the big-brained team that you are, you quickly discover that brains can be determined to be smart or not if the brain can return a correct answer to a given Boolean expression. You quickly compile a list of Boolean expressions together and set off testing one expression on each brain.

Every Boolean expression is deterministic and is recursively defined as follows

1. *expression* = '`(`' + (*value* or *expression*) + *operation* + (*value* or *expression*) + '`)`'
2. *value* = '`t`' or '`f`'
3. *operation* = '`&`' or '`|`'
4. All values and expressions can be preceded with an optional '`!`' symbol.

There are no characters in the expression other than what's listed below. Here is each character's definition:

1. '`&`' represents `AND`. (`a&b`) evaluates to true if both `a` and `b` are true; false otherwise.
2. '`|`' represents `OR`. (`a|b`) evaluates to false if both `a` and `b` are false; true otherwise.
3. '`!`' represents `NOT`. `!(a)` evaluates to false if `a` is true; true otherwise.
4. '`(`' and '`)`' are the endpoints of expressions. Order of operations stipulates that all expressions inside parentheses must be evaluated first. There is always a corresponding end parenthesis '`)`' for every beginning parenthesis '`(`'.
5. '`t`' and '`f`' represent true and false, respectively.

Input

The first line contains an integer `x` such that `0 < x ≤ 10000`. This is followed by `x` lines each containing a complete Boolean expression (up to 50 characters long) followed by one space, the equals symbol '=', another space, and the test brain's evaluation of the expression (either '`t`' or '`f`').

Output

On one line for each test brain, print the number of the brain followed by a colon and a space followed by either "`Good brain`" if the expression was evaluated correctly or "`Bad brain`" if the expression was not evaluated correctly. Once you are finished, I suggest you start running as fast as you can, as the zombies will only give you so much of a head start for your help.

Sample Input	**Sample Output**
4 (t|f) = f ((t|f)&((f|t)&f)) = f ((f&f)|(f|!(t&f))) = t (f|(f|t)) = f	1: Bad brain 2: Good brain 3: Good brain 4: Bad brain

Problem I Input file: i.in Output: monitor/stdout/cout/System.out

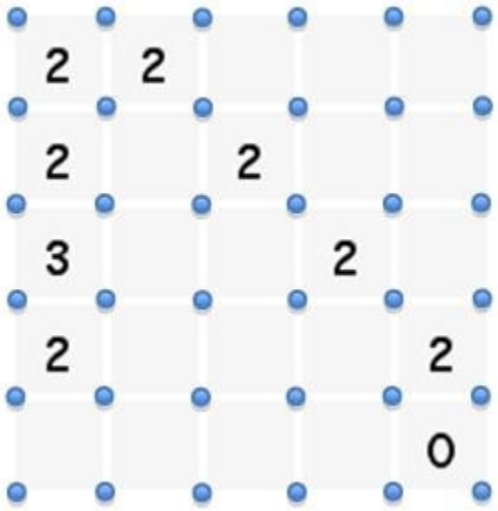
Containing Zombies and Respecting Building Codes

It's well documented that the only way to thwart zombies is to trap them in closed fences. Zombie hunters have for years been luring zombies into clear spaces and then erecting such fences at lightning speeds so they can't go very far.

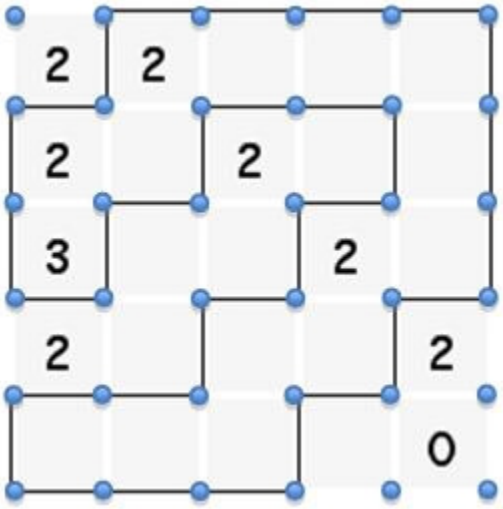
Unfortunately, government has its zoning laws, its taxes, and its building codes, and recently they started to enforce silly constraints on how these zombie-enclosing fences can be constructed. According to city codes, fence posts must be planted at regular intervals, and the fences themselves must respect arbitrary limits on how many walls can surround a unit square of land. We also want to build as long of an enclosing fence as possible, because the government has plenty of money to waste on such things.

Building Zombie Fences

Build walls between vertices to form a single enclosed fence without crossings or branches. The number indicates **exactly** how many walls — according to the crazy city building laws — must surround it (and a lack of number means there's no constraint.) So, presented with the following 5 x 5 grid of land squares:



the following fence could be constructed:



Problem I Input file: i.in Output: monitor/stdout/cout/System.out

The grid of lots is always `n x n` [$1 \leq n \leq 6$], and each lot is either a number (`0`, `1`, `2`, or `3`) to impose a constraint, or a blank if no constraint is being imposed. You're to output the length of the longest possible loop (or equivalently, the number of vertices in the loop), or `-1` if no loop exists. Note that loops of length 0 are invalid. A valid loop must enclose a non-zero amount of area.

Input

The input is a series of zombie fencing problems, expressed by `n`, the dimension of the problem, followed by an `n x n` grid ($1 \leq n \leq 6$) with the number constraints (with the - to represent no constraint), followed by a blank line. End of input is marked by a single `0`.

Output

For each fence problem, you should print the length of the longest fence loop that can be constructed for that problem while still respecting all constraints, or you should print `-1` if the problem has no such solution. There should be no extraneous white space, save for the newlines that separate each of the fence lengths.

Sample Input	**Sample Output**
2 22 22 3 222 222 222 5 ----0 2---2 3--2- 2-2-- 22--- 6 222222 2-22-2 22222- 22-2-2 2-22-2 222222 0	8 -1 32 46

Save the Python Programmers!

In post-apocalyptic California, only six teams of Python programmers remain: three who use CPython and three who use Jython. Unfortunately, C++-programming zombies roam the streets, so the programmers may only leave their safe houses under the guidance of their benevolent dictator, Guido.

The six teams control a network of safe houses, and through discussions on their social network Facelessbook, they have agreed to swap safe houses — that is, the CPython programmers will move to the houses occupied by the Jython programmers, and vice versa. Each night, Guido will guide one team from one safe house to a different, nearby one, and he will do this every night until the six teams have exchanged safe houses.

Each safe house is only large enough for one team, so no two teams can be in the same house at the same time. Tremendous distrust exists between the CPython and the Jython programmers, so they insist that Guido alternate between leading a CPython team and a Jython team (although on the first night he may pick either.)

The network of safe houses is well known, both how many exist, and which pairs of houses are close enough to travel between in a single night. Your job is to determine the minimum number of nights it will take to exchange the six teams, if it can be done at all.

There will be at most twenty safe houses, each identified by a single lower-case letter. The CPython teams start in houses a, b, and c; the Jython teams start in houses d, e, and f. At the end of the transfer, the CPython teams must end up in houses d, e, and f, and the Jython teams must end up in houses a, b, and c.

Input/Output

The input will be in a single line, which gives the connections between the houses. Each line will consist of space-separated "words". Each word indicates a connection between the house represented by the first character of the word, and the houses represented by every subsequent character of the word.

For each scenario, if there is a solution, print the minimum number of nights required for the move; if there is no way to make the move, print a single line containing "`No solution.`"

Examples

The first example puzzle, Figure 1, described by `gabd hbcf edf`, can be solved in 34 moves. (It is presented here in a size where you can try it by hand, by placing pennies on the top row and nickels on the bottom row, and swapping the two rows according to the constraints listed above.)

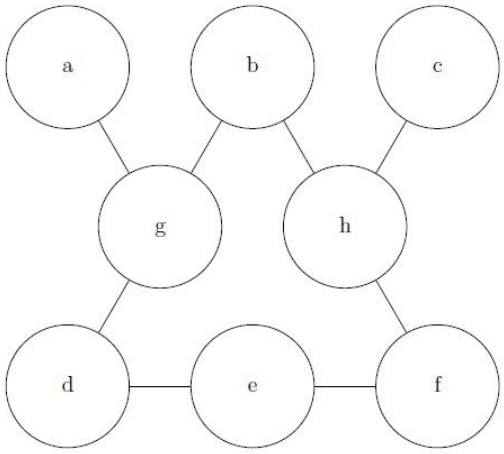


Figure 1: A moderately difficult puzzle

The next puzzle, shown in Figure 2, described by `bacgh dge feh`, requires 46 moves.

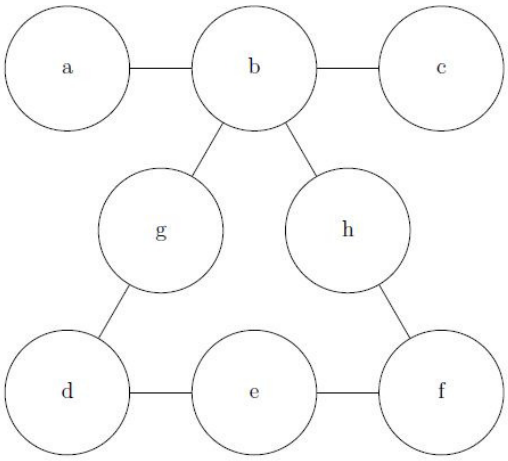


Figure 2: A harder puzzle

The situation depicted in Figure 3, `gab hbdf ibc edf`, requires 62 moves.

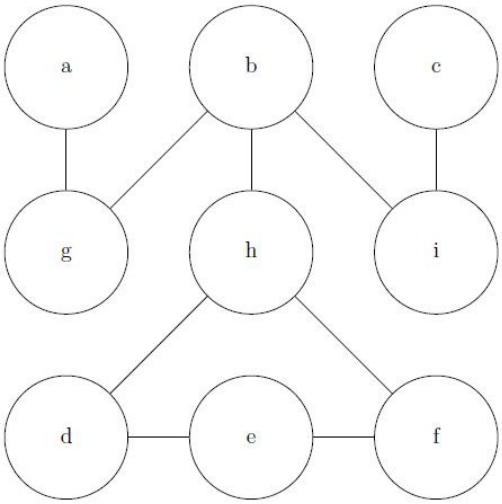


Figure 3: Good luck with this one!

Sample Input	**Sample Output**
gabcdef	No solution.
gabcdefh	No solution.
bac dce feg ga	No solution.
ade bef cfg gd	7
gab hbdf ibc edf	62

This puzzle was created by Bob Hearn at Dartmouth.

Zombie Blast!

Help!!! The zombies are marching! The zombie invasion has begun, and their legion is on the field, coming toward our last line of defense.

All hope is not lost though. In anticipation of the forthcoming doom, you have deployed a host of Adjustable Conflagration Mines (ACMs) on the battlefield. You can detonate all these mines simultaneously to a single, blast radius you control, and each mine will instantaneously incinerate any zombie within its blast radius.

You take a satellite image to give you a map of the situation. The map is a rectangular region divided into square cells of unit side length. Each cell is either empty, occupied by a zombie (`Z`), or occupied by a mine (`M`). For example, it could look like this:

M			Z
		Z	Z
M			Z

A zombie will be incinerated by a mine if the distance between the center of the cell it occupies and the center of the mine's cell is less than or equal to the blast radius. In order to minimize collateral damage, you have to set off the mines with the smallest blast size that will still incinerate all the zombies. For a given invasion scenario, what exactly is that radius?

Input

Input begins with a line containing a single integer `N`, the number of invasion scenarios (maps) for which you need to determine a blast radius. Each scenario begins with a line containing two space-separated integers, `w` and `h` (`1 ≤ w, h ≤ 2000`), indicating the width and height of the map. Then the map follows as `h` lines of text with `w` characters each, telling you what occupies each cell:

- '`Z`' denotes a zombie
- '`M`' denotes a mine
- '`.`' denotes an empty cell.

Every map will have at least one zombie (`Z`) and one mine (`M`) on it.

Output

For each invasion scenario, output on a single line a real number `r`, the smallest blast radius with you must set off the mines in order to incinerate all the zombies. Your output must be accurate to an error of no more than a factor of 10^{-6} relative to the exact answer. (For example, if the correct answer is 50, any answer between 49.99995 and 50.00005 will be accepted.)

Sample Input	**Sample Output**
2 4 3 M..Z ..ZZ M..Z 5 4 .ZZ.M Z.Z.. .Z.ZZ Z.Z.Z	3.16227766 5

Letter	Color
A	Green
B	Red
C	Yellow
D	Light Blue
E	White
F	Purple
G	Pink
H	Grey/Silver
I	Orange
J	Tan/Brown
K	Dark Blue