icpc.foundation
icpc International Collegiate Programming Contest
TWO SIGMA north america sponsor
JET BRAINS programming tools sponsor
2018 ICPC Southern California Regional Contest

2018/2019 Southern California Regional International Collegiate Programming Contest Problem Set

2018/2019 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 1
Backup Orphans

GigantoCorp has a problem. They have removed all tape drives from their data center and are doing all of their backups to disk storage. However, they seem to have used more disk space than their backup software indicates should be in use. They think that there are files on the backup storage that are not in the backup software index. They would like your team to write a program to determine if there are "orphan" files on the backup storage that are not in the backup software index—or "orphan" index entries that have no corresponding files on the backup storage.

Input to your program is structured in two blocks. The first block is the index of backup images, one per line. Each backup image name consists of 1 to 32 printable ASCII characters (no spaces). There may be up to 100,000 images. Each backup image name is unique, and the image names may occur in any order. The block ends with an empty line.

The second block contains a list of backup file names, one per line. A backup file name has the format *imagename_time_type*, where *imagename* is a backup image name in the format described above, *time* is an integer in the range 0 to 2,000,000,000 inclusive representing the time in seconds since January 1, 1970 at 00:00 GMT (the "UNIX epoch"), and *type* is a string of one to six uppercase letters. There may be up to 300,000 backup file names, and they may occur in any order. Each backup file name is unique. There may be multiple files that correspond to the same backup image. The block ends with the end-of-file.

Your program is to print the list of files with no corresponding index entries, one per line, by printing the character "F", a single space, and the file name. After all the orphan files are printed, print the orphan index entries one per line by printing the character "I", a single space, and the backup image name. Entries in each list are to be printed in ASCII lexicographic order. No leading or trailing whitespace is to be printed on an output line.

If every file has an index entry and every index entry has one or more files, print a single line containing only the string *"No mismatches."* (without the quotes, but with the period).

Sample Input

```
payroll.xls
projects.doc
employees.dat
products.txt

payroll.xls_1539199053_INCR
employees.dat_1539199053_INCR
payroll.xls_1539112653_INCR
employees.dat_1539112653_INCR
payroll.xls_1539026253_FULL
employees.dat_1539026253_FULL
customers.dat_1539026253_FULL
```

Problem 1
Backup Orphans (continued)

Output for the Sample Input

```
F customers.dat_1539026253_FULL
I products.txt
I projects.doc
```

2018/2019 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 2
Lost is Close to Lose

Better Documents Inc. is contemplating the next generation of word processors. Now, nearly every word processor includes a Spell Checker. BDI, however, is looking forward to replacing that with a true Typo Checker. We've all been caught, after relying on a spell checker, by typing mistakes ("typos") that just happen to wind up as a correctly spelled word, just not the word we intended. BDI hopes to use AI to determine when a word's context suggests that it is out of place and probably should have been a different, but similarly spelled, word.

As a first step in this process, they want to see how common such similar words really are in ordinary text. Write a program that reads paragraphs of text and produces a list of similarly spelled words occurring in that text.

For the purpose of this program, a *word* is any maximal string of non-whitespace characters containing at least one alphabetic character. *Whitespace* can be either blanks or line terminators. The *core* of a word is what you have left after removing any non-alphabetic characters and replacing any upper-case letters with their lower-case equivalents.

Two words are considered to be *similarly spelled* if the core of one word can be converted to the core of the other word by a single application of any one of the following transformations:

- Delete a single character.
- Insert a single alphabetic character.
- Replace a single character by a different alphabetic character.
- Transpose any two adjacent characters.

Input consists of 1 to 100 lines of text, followed by an end of input marker in the form of a line containing only the string "***". Each line of text will contain 0 to 80 ASCII characters (not counting line terminators).

For each word core in the text that has one or more similarly spelled words, print a line consisting of

- That word core
- A colon followed by a blank
- A list of all similarly spelled word cores (with no duplicates and not containing the core to the left of the colon), in alphabetic order, separated by a single blank.

The lines printed should be in alphabetic order of the word cores to the left of the colon. If there are no similarly spelled words in the input, print a single line containing only the string "***".

No leading or trailing whitespace is to appear on an output line.

Sample Input

```
Lost is Close to Lose

"Better Documents Inc. wants to add Typo Checking in to the
next generation of word processors," he said.
***
```

Problem 2
Lost is Close to Lose (continued)

Output for the Sample Input

```
close: lose
he: the
in: inc is
inc: in
is: in
lose: close lost
lost: lose
the: he
```

2018/2019 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 3
Collusion on Two Wheels

Two bicycle courier services have been competing in Metro City for many years, stealing customers back and forth from one another. Recently, they have come to realize that they would be better off if they could attract new customers instead. A problem is that, over time, each company's customers have become so widely scattered across the city that some of their delivery times are unacceptably slow, leading to customer dissatisfaction and a poor reputation.

The companies would like to divide up their current customers so that each company could run ads saying "We guarantee delivery in no more than M minutes", hoping to attract new customers. The streets of Metro City are laid out on a grid, with buildings at integer coordinates of the grid. The couriers must travel along those roads—they cannot cut through buildings. It takes one minute to travel one unit of distance in the x or y direction.

Your team is to write a program to divide up the current customer base. The objective is to minimize the longest delivery time required by either company to have a courier travel from one customer of that company to any other customer of the same company.

- A delivery is considered to have been completed when the courier reaches the (x, y) address of the customer. No time is counted for wandering the hallways of the customer's building.
- It is acceptable that a courier traveling from one customer to another will pass by customers of the same or of the other company. No extra time delay is accrued for riding past a customer.
- If it should happen that one company winds up with only a single customer, that company puts someone on site to deliver messages within that one building. This is considered a zero delivery time.

Input to your program starts with a line containing an integer N, the number of customers to be divided up. $2 < N \leq 1{,}000$. This is followed by N lines, each containing a pair of integers x and y denoting the position of one customer. $0 \leq x, y \leq 1{,}000$.

Your program is to print a single line containing the longest delivery time required by the two companies (i.e., the maximum of the two longest delivery times offered by each company separately).

Sample Input

```
6
1 1
4 1
1 5
10 10
10 8
7 10
```

Output for the Sample Input

```
7
```

2018/2019 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 4
Floor Tiling

A construction crew has to cover the floor of a room with square tiles. The tiles come in different sizes, the largest of which is $N \times N$ inches, where N is a power of 2. Other available sizes are $N/2 \times N/2$, $N/4 \times N/4$, and so on, down to 1×1. The room is not necessarily rectangular, but it is guaranteed to be a convex quadrilateral.

The workers were instructed to cover as much as possible with $N \times N$ tiles, then switch to $N/2 \times N/2$ and cover as much of the remaining area with those tiles, and so on, reducing the size of the tiles used at each step. The foreman of the crew realized that the optimal tiling pattern satisfying these constrains was too difficult to determine, so he decided to simplify the tile placement algorithm to something that would be easy for his crew to follow and that seemed to produce "pretty good" results.

The foreman's algorithm is as follows:

- Pick a point somewhere inside the room to be the origin of a coordinate system with east-west (X) and north-south (Y) axes.
- Place $N \times N$ tiles aligned to the X and Y axes, while making sure that the coordinates of the tile corners are multiples of N.
- For each smaller size $K \times K$ the tile coordinates must be multiples of K.

Write a program that computes the number of tiles of each available size that will be needed to cover a room. Figure 4-1 below shows an example that corresponds to the sample input.

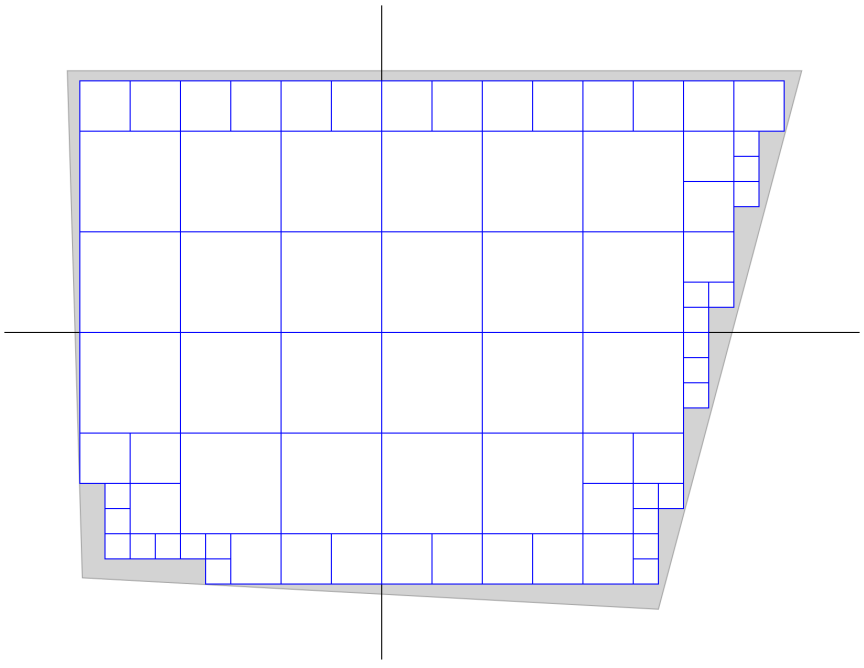

Figure 4-1. Tiling example corresponding to the sample input.

Problem 4
Floor Tiling (continued)

The input will consist of one or more lines, each describing a test case. The format of each line will be the following:

`N` x_1 y_1 x_2 y_2 x_3 y_3 x_4 y_4

where $1 \leq N \leq 1024$ is the largest tile size available, and (x_i, y_i) are the four corners of the room in counter-clockwise order. All x and y coordinates will be in the range $[-10^6, 10^6]$ and the origin $(0, 0)$ is guaranteed to be inside the room.

For each line in the input the output should contain one line in the following format:

C_n C_{n-1} ... C_0

where $N = 2^n$ and C_k is the number of tiles of size 2^k needed. Values are to be separated from each other by single spaces. No leading or trailing whitespace should appear on an output line.

Sample Input

```
4 11 -11 16.7 10.4 -12.5 10.4 -11.9 -9.75
```

Output for the Sample Input

```
22 31 22
```

2018/2019 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 5
Playing the Slots

The small nation of Erratica prides itself on defying conventions established by the more "boring" countries around the world. One of their more obvious distinctions lies in the design of their coinage. Believing that a person should be easily able to identify the value of a coin in a pocket or change purse by touch alone, Erratica designs its coins as polygons. For stability, the coins are convex—there are no notches cut into the coins. But, the overall shapes themselves can be quite irregular. Curiously, Erratica national law states that all coins must be the same thickness.

Erratica Vending, the only manufacturer of vending machines in the nation, has been sent the design for a new coin to be issued. Their machines are designed so that coins enter through a slot into a channel, with the slot being the open face of the channel. From the outside of the machine, the slot appears as a narrow rectangular hole, wide enough for the standard thickness of the coin, but long enough to accomodate the entire coin.

The company wants to know what would be the smallest slot length they will need so that the new coin can be slipped, after some rotation, into the slot.

Input begins with a line containing N, the number of sides to the polygonal coin ($3 \leq N \leq 20$). This is followed by N lines, each containing two floating point numbers, x and y, $0.0 \leq x, y \leq 100.0$, that are the coordinates of a vertex of the polygon. The two floating point numbers are separated by at least one space. All N vertices will be distinct, and the vertices will be presented in an order proceeding clockwise around the perimeter of the coin.

Print a single line with a floating point number, rounded to and printed with two decimal places, denoting the minimum slot length allowing the coin to pass through. No leading or trailing whitespace is to appear on the output line.

Sample Input

```
3
0 0
0.71 3.54
4.21 4.21
```

Output for the Sample Input

```
2.00
```

2018/2019 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 6
All Your Base

A magician has been invited to perform mental tricks at a math conference. For her final trick she picks two volunteers from the audience and places them in separate booths. She then proceeds to transmit a number using "thought projection" to both volunteers. Each volunteer writes the first number between 1 and $2^{64} - 1$ that comes to mind on a piece of paper. When the volunteers come out of the booths and show their numbers to the audience the two numbers match! Well, sometimes they do... Sometimes the numbers look different, but the magician claims that the two numbers are simply the same number written in two numeral systems with different bases. Your task is to write a program to verify the magician's claims.

In general, a sequence of digits

$$a_m a_{m-1} ... a_1 a_0$$

in base b represents the number

$$a_m b^m + a_{m-1} b^{m-1} + ... + a_1 b + a_0$$

For example, $101_2 = 5_{10}$.

Sometimes a piece of paper shown to the audience has a mix of digits and English letters, or, in some cases, only letters. The magician has an explanation for that too—the base b is greater than 10. For $b > 10$ the letters of the alphabet are used to represent digits greater than 9. For example, in the hexadecimal system (base 16) it is common to use $A = 10, B = 11, ..., F = 15$. We can extend this notation to $G = 16, ..., X = 33, Y = 34, Z = 35$ to represent digits for numeral systems up to base 36. For example, $\mathrm{XYZ}_{36} = \mathrm{ABFB}_{16}$.

Input to your program will start with a number N on the first line, followed by N lines, each containing two alphanumeric strings separated by a space, representing pairs of numbers from two volunteers. Each string will consist of at most 64 characters.

The output should contain N lines, one for each pair of strings, containing only the word *"yes"* if there exist two bases b_1 and b_2, $2 \leq b_i \leq 36$, such that the first string in base b_1 represents the same number as the second string in base b_2, or only the word *"no"* if no such bases exist. No leading or trailing whitespace is to appear on an output line.

Sample Input

```
3
101 5
XYZ ABFB
Z1G L0L
```

Output for the Sample Input

```
yes
yes
no
```

2018/2019 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 7
Find Poly

Your team is to write a program that examines a set of geometric figures and counts how many are polygons. For the purposes of this problem:

- A geometric figure is a set of line segments connected directly or indirectly at their end points.
- A polygon is a geometric figure where all the connected line segments form exactly one loop.

Figure 7-1 has ten geometric figures of which a, b, e, and f are polygons. The dots represent the end points of the line segments. Note that b is self-intersecting, but that intersection is not at the end points of the intersecting line segments. Similarly, c and d as well as e and f intersect but are not connected.

The input is a series of lines terminated by end-of-file. Each line will have one or more line segments of the form:

$(x1, y1), (x2, y2);$

where $(x1, y1)$ is one end point and $(x2, y2)$ is the other end point. The separator characters, "(),;" may be preceded and followed by white space. A line will be at most 100 characters long. There will be at most 200 line segments.

A given line segment will appear only once in the input and no line segment will be of length 0. Each x and y will be integers ≥ 0 and ≤ 99. Line segments are not directed, so the order of the end points in the line segment is not significant. The order of the line segments in the input is also not significant.

Your program is to print a single line with the total number of geometric figures followed by a single space and the number of polygons found. No signs, leading zeroes, or leading or trailing whitespace are to appear on the line.

Sample Input

```
(84,84),(78,84);(68,60),(64,64);(20,85),(15,88);(0,0),(2,8);(30,60),(30,66);
(13,40),(18,38);(15,88),(15,95);(18,38),(8,38);(31,7),(25,10);(30,66),(26,70);
(40,14),(30,19);(5,85),(15,88);(48,20),(56,26);(84,84),(84,82);(66,82),(70,86);
(15,95),(25,90);(70,86),(66,88);(59,23),(50,27);(15,88),(5,80);(78,84),(74,82);
(60,80),(66,82);(5,85),(5,80);(25,10),(40,14);(20,85),(25,90);(20,60),(30,66);
(13,36),(14,30);(30,60),(20,60);(64,64),(60,60);(31,7),(30,19);(15,88),(25,90);
(68,60),(76,64);(8,38),(13,40);(5,85),(15,95);(0,0),(10,4);(10,30),(14,30);
(74,82),(70,86);(10,30),(12,43);(6,10),(10,4);(5,80),(20,85);(6,10),(2,8);
(60,80),(66,88);(84,82),(74,82);(12,43),(13,36);
```

Output for the Sample Input

```
10 4
```

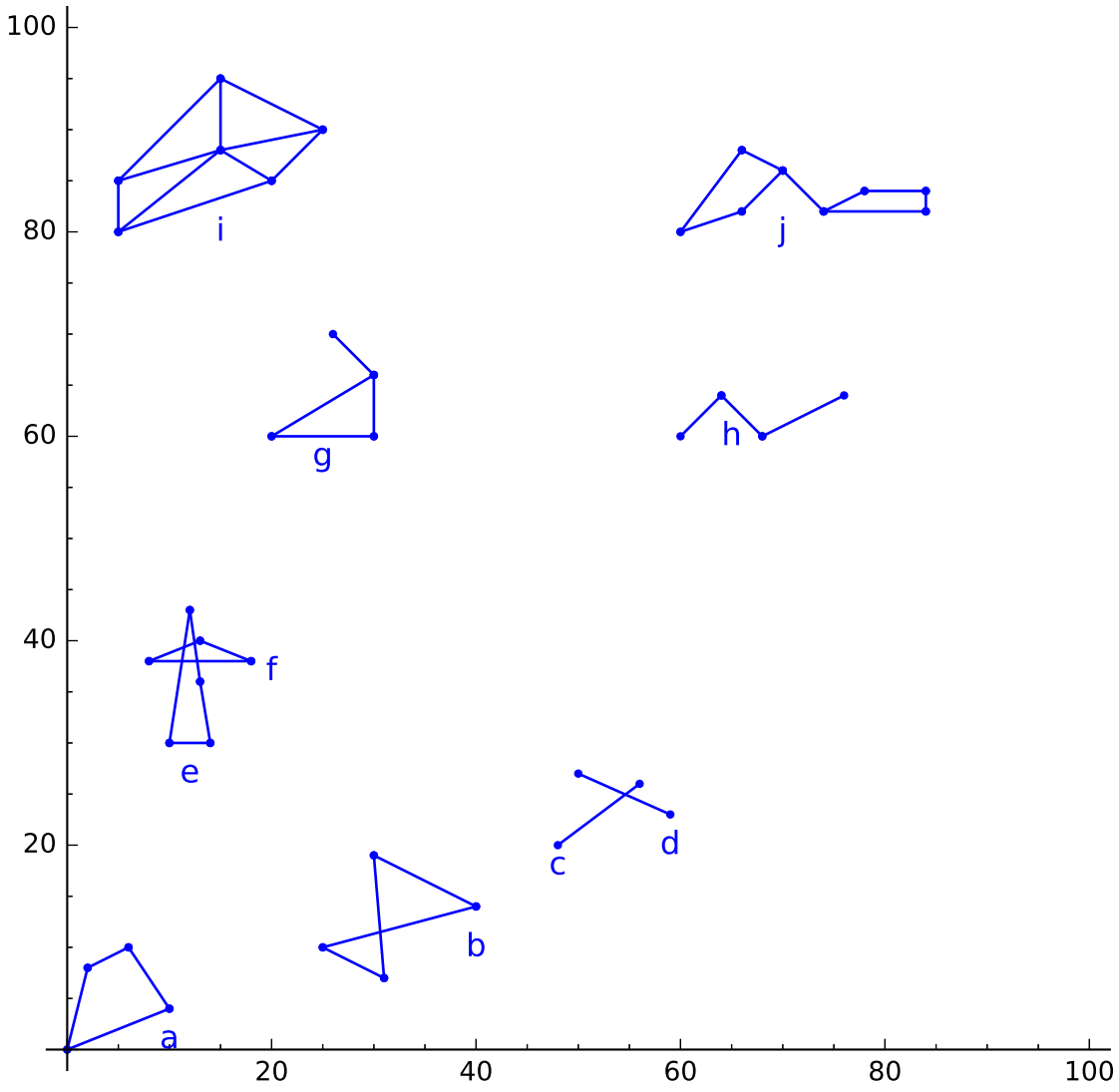

Figure 7-1. Figures in the Sample Input.

2018/2019 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 8
Capacity Planning

Capacity planning involves taking known quantities or rates, and projecting the consumption or outcomes based upon another factor. For example, one might ask *"How many megabytes are needed to store a 10 kilobit per second data stream that runs continuously for 52 weeks?"* Another question might be *"How fast is something falling, in meters per second, after accelerating from rest for 2 minutes?"* A compact way of writing the above questions is

```
10 kb|s,52 wk,MB
1 g0,2 min,m|s
```

where the vertical bar "|" is pronounced "over" or "per".

Capacity planning can also be useful for unit conversion, such as *"How many fluid ounces in a gallon?"* Unit conversion also doubles as recipe scaling; for example *"If the recipe calls for 2 tablespoons of butter per pound of flour, how many tablespoons of butter are needed for 2.5 pounds of flour?"* A similarly compact way of writing the questions is

```
1 gal,1,floz
2 TBSP|lb,2.5 lb,TBSP
```

Your program will take a table of unit conversions, then perform capacity planning computations using the appropriate unit conversions where necessary. Input to your program is a series of lines with one conversion formula per line, a blank line, then capacity planning questions, one question per line, until end-of-file. The format of a conversion equation is

$\texttt{k}_1\ \texttt{u}_1\ =\ \texttt{k}_2\ \texttt{u}_2$

where $\texttt{k}_1$ and $\texttt{k}_2$ are numeric constants, $\texttt{u}_1$ and $\texttt{u}_2$ are unit expressions. Each element is separated by exactly one space. A unit expression contains an optional numerator, an optional fraction bar "|", and an optional denominator. The numerator and denominator, when present, contain one or more unit names. When more than one unit name is present, the names will be separated by "*". For example, standard gravitational acceleration, g0=9.80665 m/s^2, and 1 Hz=1/s would be written

```
1 g0 = 9.80665 m|s*s
1 Hz = 1 |s
```

The format of the capacity planning question is

$\texttt{k}_a\ \texttt{u}_a\texttt{,k}_b\texttt{[u}_b\texttt{],[k}_c\ \texttt{]u}_c$

where $\texttt{k}_a$, $\texttt{k}_b$, and $\texttt{k}_c$ are numeric constants, and $\texttt{u}_a$, $\texttt{u}_b$, and $\texttt{u}_c$ are unit expressions in the same format as $\texttt{u}_1$ and $\texttt{u}_2$ above. Note that $\texttt{u}_b$ and $\texttt{k}_c$ are optional. The optional $\texttt{k}_c$ is present to help scale measuring devices. For example, *"If all you have is a half-cup measuring cup, how many half-cups are necessary to measure 1 quart of milk?"* The compact notation of the question would be

```
1 qt,1,0.5 cup
```

A capacity planning question might require a computation or answer that is the inverse of the stated units. For example *"If the recipe calls for 2 TBSP of butter per pound of flour, and you only have 1.5 TBSP butter, how many pounds of flour should you use?"* The compact question would be

```
2 TBSP|lb,1.5 TBSP,lb
```

Problem 8
Capacity Planning (continued)

For each capacity question, print out a single number, rounded and printed to two decimal places. The number expresses units of $\mathtt{k}_c\mathtt{u}_c$ corresponding to the appropriate computation of

$\mathtt{k}_a\mathtt{u}_a \times \mathtt{k}_b\mathtt{u}_b$
or $\mathtt{1/(k}_a\mathtt{u}_a\mathtt{)} \times \mathtt{k}_b\mathtt{u}_b$
or $\mathtt{k}_a\mathtt{u}_a \times \mathtt{1/(k}_b\mathtt{u}_b\mathtt{)}$
or $\mathtt{1/(k}_a\mathtt{u}_a\mathtt{)} \times \mathtt{1/(k}_b\mathtt{u}_b\mathtt{)}$

There will be at most fifty entries in the unit conversion table. All conversions are internally consistent. All unit conversions are simple scalings; there will not be any offset conversions such as Fahrenheit-Celsius. All capacity planning questions can be answered. No capacity questions evaluate to zero.

No leading or trailing whitespace is to appear on an output line.

Sample Input

```
1 kB = 1024 B
1 MB = 1024 kB
1 B = 8 b
1000 b = 1 kb
1 min = 60 s
1 hr = 60 min
1 day = 24 hr
7 day = 1 wk
1 g0 = 9.80665 m|s*s
1 Hz = 1 |s
4 cup = 1 qt
4 qt = 1 gal
1 cup = 8 floz

10 kb|s,52 wk,MB
1 g0,2 min,m|s
1 gal,1,floz
2 TBSP|lb,2.5 lb,TBSP
1 qt,1,0.5 cup
2 TBSP|lb,1.5 TBSP,lb
```

Output for the Sample Input

```
37490.84
1176.80
128.00
5.00
8.00
0.75
```

2018/2019 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 9
Swamp County Toll Roads

Swamp County has just completed its first express toll roads. As is now the norm, there are no attended toll booths—all tolls are collected electronically. Rather than issue electronic tags or transponders to road users, toll collection is done based on the automated reading of license plates as vehicles pass through a toll plaza.

For a variety of reasons, sometimes the license plate images are of low quality. When this happens it's possible for license plates to be mis-read, particularly since some characters closely resemble others. The county has studied cases where this has happened and has come up with estimates of the likelihood that a given character will read correctly.

The county would like your team to write a program that will, given the error estimates for individual characters as stated above, take a list of license plates as read and determine the probability that the entire license plate was read correctly.

Input to your program will begin with a list of accuracy estimates. Each estimate consists of the character, a single space, and its estimate of a correct read, in the range $0 < p < 1$. For example, the first line of the sample input indicates there is an 88% chance that an "I" read from a license plate is actually an "I". Any characters not in the list are taken as 100% ($p = 1$). You may assume that the reading of one character has no effect on the reading of another character (they are independent). This list ends with an empty line.

The remaining lines of input contain license plates as read, one per line, starting in the first column. Each license plate contains a string of one to eight upper-case letters or digits. The list of license plates ends with end-of-file.

For each license plate, your program is to print a line with the the probability that the license plate was read correctly as a value between zero and one. The value is to be rounded to and printed with three digits after the decimal point. No leading or trailing white space is to appear on an output line.

Sample Input

```
I 0.88
1 0.99
0 0.87
O 0.95
Q 0.87

PROGRAM
ICPC2018
2JKB843
1JKL893
SNU8J5
```

Problem 9
Swamp County Toll Roads (continued)

Output for the Sample Input

```
0.870
0.758
1.000
0.990
1.000
```

2018/2019 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 10
ABC Conjecture

The ABC Conjecture states that given three positive integers a, b, and c such that $c = a + b$, with a and b having no common prime factors, the product of the distinct prime factors of $a \times b \times c$ is usually not much smaller than c.

In order to understand the conjecture, we define the radical of an integer n (denoted as $rad(n)$) as the product of the distinct prime factors of n. Stated another way, this is the product of all the primes in n taken to the first power. For example, $rad(36) = rad(2 \times 2 \times 3 \times 3) = 2 \times 3$ since 2 and 3 are the distinct primes in 36.

The full conjecture says that $rad(a \times b \times c)^{\omega}$ is usually not smaller than c; and, if $\omega > 1$, even by just a little bit, there are only a finite number of exceptions. If $\omega = 1$ there are an infinite number of exceptions but they are still rare.

Of course the interesting cases are where $rad(a \times b \times c)$ is less than c. Your team's job is to write a program to explore this conjecture where $\omega = 1$.

The input is a series of lines terminated by end-of-file. Each line will have a test case of two positive integers, a and b, separated by spaces. Each integer will be no greater than 2,000,000.

For each test case, print the answer on a separate line. If a and b have at least one prime factor in common, print only the word "bad". Otherwise print only the word "less", "equal", or "greater" if $rad(a \times b \times c)$ is less than, equal to, or greater than c, respectively. No leading or trailing whitespace is to appear on an output line.

Sample Input

```
4 127
3 125
5 30
1 4374
1999979 1999993
2000000 1999999
122808 877200
52441 3125
161051 158949
57987 96645
1827904 906471
1192452 1770344
```

Problem 10
ABC Conjecture (continued)

Output for the Sample Input

```
greater
less
bad
less
greater
greater
bad
less
less
bad
less
bad
```

2018/2019 SOUTHERN CALIFORNIA REGIONAL INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 11
Picking Up the Dice

Two players are playing a game with a set of K six-sided dice. One player calls out a number in the range K through $6K$ inclusive and the other tries to roll that number with K dice. After the first roll, the rolling player is allowed to pick up any number (0 through K) of dice and re-roll them.

Given the number of dice, the target number the player wants to roll, and the set of numbers the player obtained on the first roll, what number of dice should the player pick up to maximize their chances of getting the target number on the second roll? Your team is to write a program to answer that question.

Input to your program begins with a line containing 2 integers separated by one or more spaces: K, the number of dice, and T, the target number. $2 \leq K \leq 24$, $K \leq T \leq 6K$.

The next line contains K integers, indicating the numbers that were rolled on each of the dice on the first roll. All will be integers in the range 1 through 6 inclusive. Input values will be separated from each other by one or more spaces.

Your program is to print a single line containing an integer denoting the number of dice that the roller should pick up and re-roll in order to maximize the chances of getting an overall sum of T. If there are multiple options for the number of dice to choose that have the same maximum chance of getting an overall sum of T, choose the smallest such number.

No leading zeroes, signs, or leading or trailing whitespace are to appear on the output line.

Sample Input

```
3 9
5 4 1
```

Output for the Sample Input

```
1
```