Important Contest Instructions!!

Please *read the following instructions carefully. They contain important information on how to run your programs and submit your solutions to the judges. If you have any questions regarding these instructions, please ask a volunteer before the start of the competition.*

Program Input
Most programs will require input. You have two options:

1) Your program may read the input from a file. The input data will be in the local directory in the file **probXX.txt**, where 'XX' is the problem number.

2) Your program may read the input from the keyboard (standard in). You may type everything on the keyboard, or you may copy the data from **probXX.txt** into the standard in. ***Tip:*** *Type 'Ctrl-Z <return>' to signal the end of keyboard input.*

   Note: An easy way to enter keyboard data is by redirecting the contents of a file to your program. For example, if you are executing prob01, the input file **prob01.txt** can be redirected to the standard in of your program using syntax like this (examples are shown for each of the allowed languages):

```
%> java prob01 < prob01.txt
%> java –jar js.jar prob01.js < prob01.txt
%> python prob01.py3 < prob01.txt
%> prob01.exe < prob01.txt
```

   Your program will behave exactly as if you were typing the input at the keyboard.

Program Output
All programs must send their output to the screen (standard out, the default for any print statement).

Submitting your Programs

Interpreted Programs (Java, JavaScript, Python) Your program must be named probXX.java / probXX.js / probXX.py2 / probXX.py3, where 'XX' corresponds to the problem number. For Python, use the extension that matches the Python version you are using. Please submit only the source (.java, .js, .py2 or .py3). For java, the main class must be named probXX. Note there is no capitalization. All main and supporting classes should be in the default (or anonymous) package.

Native Programs (C, C++, etc.) Your program should be named probXX.exe, where 'XX' corresponds to the problem number.

You are <u>strongly</u> encouraged to submit solutions for Problems #0 and #1 (see next pages) <u>prior</u> to the start of the competition to ensure that your build environment is compatible with the judges' and that you understand the Input and Output methods required.

Problem 0 Rocky Mountain Hello [1 point]

NOTE – this is the 1st of two problems that can be solved and submitted before the start of the CodeWars competition. Teams are **strongly** encouraged to submit this problem **prior** to the start of the competition – hey, it's basically a free point!

Summary

The sole purpose of this problem is to allow each team to submit a test program to ensure the programs generated by their computer can be judged by our judging system. Your task for this program is a variation on the classic "Hello World!" program by saying hello to our newest CodeWars site – Fort Collins, Colorado. In tribute to the local university, Colorado State, all you have to do is print "Go Rams Go!" to the screen.

Output

```
Go Rams Go!
```

codewars 2017 years

Problem 1 All Talk [1 point]

NOTE – this is the 2nd of two problems that can be solved and submitted before the start of the CodeWars competition. Teams are **strongly** encouraged to submit this problem **prior** to the start of the competition – hey, it's basically a free point!

Summary

Without good communication skills, life will be more challenging for you. And if you don't know how to do Input and Output properly, the CodeWars competition will not go well for you. So, here's your chance for a little practice before the actual contest begins.

Write a program to introduce your team to someone by their first name.

Input

The input will be your new friend's first name:

```
Wilbur
```

Output

Greet your new friend with a friendly, creative greeting of some sort that includes the person's name:

```
Salutations, Wilbur! We are the Fighting Sandcrabs from Port Lavaca HS!
```

**If you're confused at this point regarding inputs and outputs, go back and re-read your contest instructions.*

codewars 2017 years

Problem 2 A Momentous Occasion [2 points]

Summary

Congratulations! You are about to become the first human to land on the surface of Mars! You will need to work out one important detail before landing: exactly how much thrust (or force) will be required for your spacecraft to slow down. In order to answer that question, first you'll need to know your spacecraft's momentum.

In physics, Momentum is the product of the mass and velocity of an object. Mathematically speaking, it is also the same as the product of force and time. Newton's second law states that the change in an object's momentum is proportional to the force applied to that object. For example, a massive, quickly moving spacecraft has a large momentum and it would take a large (or prolonged) force to accelerate the spacecraft to that speed. In the same way, it would take a large (or prolonged) force to stop the spacecraft's motion. If the spacecraft were lighter or moving more slowly, then the amount of force required for those actions would also be smaller.

Mathematically, the momentum `p` is calculated by multiplying the mass `m` and the velocity `v`.

```
p = m * v
```

Write a program to calculate the momentum of a spacecraft given its mass and velocity.

Input

The input consists of two positive integer values: the mass and velocity of a spacecraft. The maximum of either value is 16384.

```
14392 9334
```

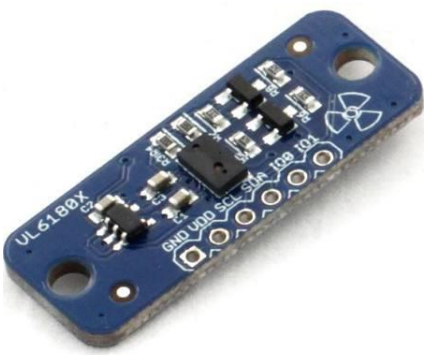
Output

The program must print the momentum of the spacecraft.

```
134334928
```

codewars 2017 years

Problem 3 — Time of Flight Sensor [3 points]



Summary

You probably already know that light travels really, really fast. Really. However, you may not know that you can use light to measure distances by timing how long it takes a pulse of light to reflect from a surface. A working TOF sensor requires really high-speed electronics because light travels 0.299792 millimeters per picosecond. The most amazing part is you can buy a Time-of-Flight sensor online for less than $20. It's true. No, really. It's pretty cool, too. Look it up after the contest.

So, anyway, given a number of picoseconds elapsed for a reflection, you can calculate the distance to the reflecting object and back (in millimeters) by simple multiplication. Write a program to calculate distances for a Time of Flight sensor.

```
distance = time * 0.299792
```

Input

The program input consists of three positive integers, each on a separate line. These numbers are reflection times ranging from 100 picoseconds to 1,000,000 picoseconds.

```
150
917203
32767
```

Output

The program must print the calculated distances, in millimeters, for each input value. The result must be accurate to within +/-1 millimeter.

```
44.9688
274970.121776
9823.284464
```

codewars 2017 years

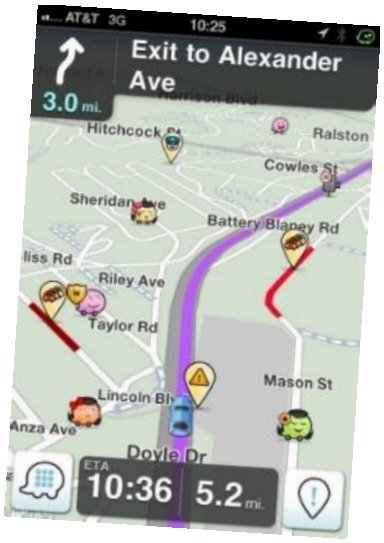
Problem 4 Alternate Route [3 points]

Summary

A large truck carrying bismuth subsalicylate (a common antacid medication sold as a pink liquid) has collided with a passenger vehicle. Both drivers escaped unharmed, but the vehicles caught fire and the freeway has been closed while the fire is extinguished and the roadway is cleaned. The cleaning is expected to take a long time because bismuth subsalicylate leaves behind a shiny metal oxide slag after being burnt. Think about that the next time you have heartburn.

Meanwhile, freeway traffic has been diverted to an alternate route for about 40 miles. Unfortunately for the motorists, the alternate route is under construction and all traffic must merge into a single lane. As you might imagine, the congestion behind this merger is causing several hours of delay for travelers. Vehicles alternate between being fully stopped and driving forward at very slow speeds. As the day progresses, more and more vehicles enter this mess and those coming at the end spend more time caught in the diversion than those at the beginning.

Write a program to calculate the average speed of a vehicle given a number of miles traveled and the number of minutes elapsed.



Input

The first line of input indicates how many vehicle records the program must read (up to a maximum of ten). Each vehicle record appears on a separate line and includes the number of miles traveled and the number of minutes elapsed.

```
4
12.7 258
8.6 162
9.9 176
6.5 98
```

Output

For each vehicle record, the program must print the average speed in miles per hour. Answers must be accurate to within +/- 0.01 miles per hour.

```
2.953488372
3.185185185
3.375
3.979591837
```

Problem 5 Space Race [4 points]

Summary

The annual Inter-Planetary Explorer League Triad Race is an amazing competition in which trios of pilots must navigate asteroid fields and endure high-G pulls from gravity-assisted slingshot maneuvers. The race ends with each spacecraft rounding a narrow turn and then launching an identity transceiver at a hyper-dimensional portal. It is strictly forbidden for the transceivers to have any type of propulsion system, so the pilot team must have really good aim to shoot the transceiver through the portal. The team whose transceiver enters the portal first is named the winner of the race.

Write a program that can determine the winner of the Triad Race given a set of transceiver trajectories.

Input

The first line of input indicates the number of teams (up to a maximum of seven) that survived the race long enough to launch their transceivers at the portal. Poorly-aimed transceivers that will miss the portal are not included. Each input line afterward includes the team name, the distance to the portal in kilometers, and the speed of the transceiver in kilometers per second.

```
3
Dragons 914.8 168.2
ChartMonkeys 1273.1 193.5
SolarWind 1144.6 181.3
```

Output

The program must print the name of the winning team and the number of seconds that elapsed until the transceiver entered the hyper-dimensional portal.

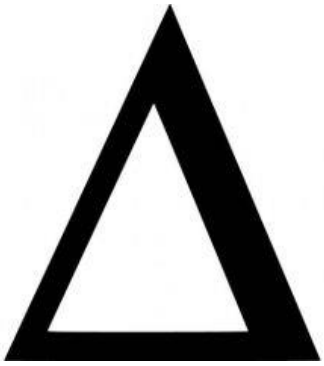
```
Dragons 5.438763377
```

Problem 6 Delta Inversion

[5 points]

Summary

Given a sequence of integers, write a program to 1) compute the deltas (differences between values), 2) invert the deltas, and 3) compute a new sequence based on the new deltas. For example, if you started with the following sequence:



```
33 47 48 19 86 6
```

The deltas would be:

```
14 1 -29 67 -80
```

The inverted delta values would be:

```
-14 -1 29 -67 80
```

Finally, applying these inverted deltas and starting with the first number of the original sequence gives this new sequence:

```
33 19 18 47 -20 60
```

Input

The first line of input indicates the number of integer sequences (up to 3). The first integer on each subsequent line indicates the number of values in a sequence (up to 32) and is followed by the integer sequence. The integer values in a sequence may be between -999 and 999 (inclusive).

```
2
7 -321 524 12 0 924 -658 -2
5 246 -651 -650 -650 31
```

Output

The program must print the new integer sequences, one per line.

```
-321 -1166 -654 -642 -1566 16 -640
246 1143 1142 1142 461
```

Problem 7 Iterative Solutions [6 points]

Summary

In mathematics, an iterative process uses the result of a function and substitutes that result back into the same function. The process is repeated until the process converges to a solution or the process diverges. The process is said to converge if the difference between two successive values is less than an acceptable error value. If the process does not converge after a large number of iterations, then the function is said to diverge.

Here's a simple example of an iterative function:

```
x[n+1] = 2 + 1 / x[n]
```

The starting value is often called `x[0]`. If we started evaluating the above example with `x[0]` equal to 3 and an acceptable error value 0.001, the iterative process would look like this:

```
x[1] = 2 + 1 / 3 = 2.333333             e = | 2.333333 - 3 |         = 0.666666
x[2] = 2 + 1 / 2.333333 = 2.428571      e = | 2.428571 - 2.333333 | = 0.095238
x[3] = 2 + 1 / 2.428571 = 2.411765      e = | 2.411765 - 2.428571 | = 0.016806
x[4] = 2 + 1 / 2.411765 = 2.414634      e = | 2.414634 - 2.411765 | = 0.002869
x[5] = 2 + 1 / 2.414634 = 2.414141      e = | 2.414141 - 2.414634 | = 0.000493
```

The calculated error value for `x[5]` is less than the acceptable error value, so this function converges on the value 2.414.

For this problem, we will represent iterative functions in the following general form:

```
x[n+1] = C + ( A * x[n] + M ) / ( B * x[n] + N )
```

Input

The first line of input is the number of iterative functions that the program must solve. Each line after the first represents a single iterative function with the values for x[0], A, B, C, M, N, and the acceptable error e, in that order.

```
2
3 0 1 2 1 0 0.001
2 1 0 5 0 2 0.01
```

Output

The program must print the solution for each iterative function on a separate line. If the function does not converge after 100 iterations, then the program must print the word DIVERGES. The result must be accurate to within the acceptable error value.

```
2.414141
9.999999
```

Problem 8 Parity Gap [6 points]

Summary

The parity of a number is useful for things such as error detection in computer science. A number has even parity if the count of ones (1) in its binary representation is even. A number has odd parity if the count of ones in its binary representation is odd.

The columns below represent a number's decimal representation; its binary representation; and whether the number has even or odd parity.

```
0  : 0000 : Even
1  : 0001 : Odd
2  : 0010 : Odd
3  : 0011 : Even
4  : 0100 : Odd
5  : 0101 : Even
6  : 0110 : Even
7  : 0111 : Odd
8  : 1000 : Odd
9  : 1001 : Even
10 : 1010 : Even
```

For the numbers between 0 and 10, 6 of them have even parity:

```
0 (0000)
3 (0011)
5 (0101)
6 (0110)
9 (1001)
10 (1010)
```

For the numbers between 5 and 10, 4 of them have even parity:

```
5 (0101)
6 (0110)
9 (1001)
10 (1010)
```

Given two integers A and B, determine how many integers between A and B (inclusive) have even parity.

Input

Each test case will consist of a line containing two integers A and B ($0 \le A \le B \le 100000000000$).
Input will end with a line containing 0 0.

```
0 4
0 10
0 17
5 10
9 9
867 5309
123 100000000000
314 159265
0 0
```

Output

For each test case, output the number of integers between A and B (inclusive) that have even parity.

```
2
6
9
4
1
2222
49999999939
79476
```

codewars 2017 years

Problem 9 Cairo Skyline [6 points]

Summary

Centuries before Houston was founded, the ancient Egyptians built fascinating skylines. Your task is to draw the outline of the Cairo skyline. I've never been there, but I believe it's made up of nothing but pyramids.

Input

Each line of input consists of two integers, the x-axis center of the pyramid (the center is to the right of this value), and the height of the pyramid. The pyramids are listed in nearest to furthest order. Maximum pyramid height will be 12. There will be no more than 10 pyramids in the list.

The list of pyramids ends with `0 0`

```
9 5
15 4
15 6
0 0
```

Output

Draw the resulting skyline from the input, with the pyramids in order, nearest to furthest (some may be completely invisible). Your output must include a numbered ruler line at the bottom (as shown). The completed skyline will have a maximum width of 60 characters at its base.

```
              /\
         /\  /  \
        /  \/ /\ \
       /     \/  \ \
      /         \  \ \
     /           \  \ \
1234567890123456789012345678901234567890
```

codewars 2017 years

Problem 10 Revenge of the Giant Palindromes [7 points]

Summary

Just when you thought it was safe to forget everything you learned about Palindromes, they have returned! They're back and they are tougher than ever! But do not fear: we've gathered intelligence on these new palindromes and we believe we can defeat them. First, remember that a palindrome is a sequence of two or more letters and/or numbers that read the same backward and forward, ignoring spaces, punctuation, and capitalization in the original string. And now they have a new strategy. They have learned to disguise themselves by infiltrating normal strings. This means the devious palindromes could be hiding in plain sight within longer non-palindromic strings. Sometimes the palindromes operate alone, and sometimes they work together. But we have confronted several of these palindromes and the leader is always the longest palindrome in a group. Take out the leader and the rest will scatter.

We need you, brave programmer, to write a program that can locate the single longest palindrome in a series of input strings. Your program will be used by the palindrome extermination team and they will remove the palindromes. Now get busy!

A nut for a jar of tuna

Input

The first line of input will tell you how many lines of strings follow after it. The lines are numbered for later reference. The line length will not exceed 78 total characters. Read each line and search for palindromes!

```
4
HANNAH's RADAR LASER NAAN GRILL
PALINDROMES ARE FUN
A NUT? FOR A JAR OF TUNA!
A ham a yam a can of spam bananama
```

Output

For each input line, your program must print the reference number and the longest palindrome in that line. The program must preserve the interior spaces, punctuation, and capitalization of the original line so that the palindrome exterminators will know what they are looking for. If there is no palindrome, then the program must print the message "NO PALINDROME".

```
HANNAH
NO PALINDROME
A NUT? FOR A JAR OF TUNA
am a yam a
```

codewars 2017 years

Problem 11 Process of Elimination [7 points]

Summary

Cryptography is hyuge [sic]. The state of the art is very advanced and the algorithms are based on, like, mathematics or something. But what if you don't want to use all that fancy 1024-bit encryption technology? What if you want to obscure a message just enough to prevent the casual observer from deciphering it? Well, if that's your aim then you're in luck, because here at CodeWars we specialize in quirky algorithms that are technically interesting but only marginally beneficial.

The algorithm presented here is a two-pass procedure. First your program must count the number of occurrences of each letter in a message. On the second pass, it should eliminate letters that have N or more occurrences, where N is a pre-arranged value called the noise threshold. At this stage a human should be able to read the message, but for this problem we'll give your program some additional word length hints. Now that's security. Am I right?

One word of caution: as with all encryption algorithms, you'll still need a reliable process for wiping your server clean. Like with a towel. Don't get eliminated!

Input

The message is presented as a grid of letters. The first line of input indicates the number of rows and columns of letters and the noise threshold. The second line begins with the number of words in the solution and then the number of letters in each word. The letter grid follows. Letters are separated by spaces. There will be no more than 100 letters in the grid.

Example 1:

```
3 15 4
4 2 3 3 9
F L Y W B E G A L K R U B E T
L H G E C K Y U B H L U G A F
K Y F M P U B K F I G O N S Y
```

Example 2:

```
4 13 6
7 4 1 4 3 5 1 4
P I M L H A K E D P I L A
M E H D E L P R I C A T H
L I G R H P E D L E T I P
H A D G H L O I D O N P D
```

Output

The program must print the secret message as words on a single line.

```
WE ARE THE CHAMPIONS
MAKE A MEER CAT GREET A GOON
```

Problem 12 Tutors [7 points]

Summary

At Rushmore Academy, all of the top computer science students help with tutoring in the computer lab after class. The tutoring program at the school pairs higher-skilled tutors with lower-skilled tutees. A tutor can only be paired with a tutee if the tutor's computer science skill is greater.

Given a list of tutors, tutees, and their respective skills, tell how many valid pairs exist. Solutions should complete all test cases in less than 30 seconds. (Be sure to run with the student input files for large test cases.)

Input

Input will begin with a single integer N (N ≤ 10) representing the number of test cases.
Each test case will consist of two lines representing the tutors and tutees, respectively.
Each line will begin with an integer T (2 ≤ T ≤ 100000) representing the number of tutors or tutees on the line.
The following T numbers represent the skill S (2 ≤ S ≤ 100000) of each tutor or tutee. Skills will be in non-decreasing order.

Note: Student sample input / output contains a test case where T ~ 100000 for both the tutor and the tutee, which contains too much text for the printed problem packet.

```
3
3 10 11 12
3 7 8 9
4 2 3 4 5
5 5 6 7 8 9
9 24 27 35 38 40 47 77 79 85
8 14 23 29 38 45 53 64 89
```

Output

Output for each test case should be a single integer representing the number of valid pairs of tutors and tutees.

```
9
0
40
```

Problem 13 Duke Wordwalker [7 points]

Summary

Duke Wordwalker has risen to power to bring about a new word order. Read his proclamation and despair.

It is my humble opinion that standard text is boring. Therefore I introduce a new writing standard, walking words!

The Wordwalker algorithm is a way of printing text that changes directions when the characters U, D, R, or L are encountered (either upper or lower case). These are, of course, the first letters of the words up, down, right, and left. Whenever the algorithm encounters one of these letters, the direction of printing changes to follow the indicated direction. For example, the phrase, "Lexically Dangerous Error" would be printed like this:

```
D yllacixeL
a rror
n E
g
e s
rou
```

If the text collides with existing text, the new character overwrites the previous character. The default text direction is right until a direction letter has been encountered.

Input

The input is a single line of text, up to 140 characters in length.

```
It is my humble opinion that standard text is boring. Therefore I introduce a new writing
standard, walking words!
```

Output

The program must print the input text using Duke Wordwalker's algorithm.

```
dnats taht noinipo el
a                   b
rd                  m
          It is my hu
 t                      riting stand
 e                      w           a
 x                                  rd
 t                      w            ,
                        e
 i                      n            w
 s                                   a
                        a   rdw gnikl
 b                           s
 o                      e    !
 ring. Therefore I introc
                        u
```

Problem 14 Pathfinder [8 points]

Summary

Kryten needs your help navigating through an unusual room that he's trying to leave.

The room is configured as an X by Y grid of squares. Kryten is in the square in the upper left corner of the room. The exit controls are in the square in the bottom right corner of the room.

```
Kryten -> ....
          ....
          ....
          .... <- Exit
```

Kryten can only move down or right to squares adjacent to his current position at any time. There are some squares in the room that are filled with mining ore, and Kryten is unable to move on these squares. These squares are identified with a '#' character.

For a given room, give the number of distinct paths Kryten can take from his initial position to the square with the exit controls.

Input

Input for each test case will begin with a pair of integers X and Y (between 4 and 20, inclusive) representing the number of rows and columns in a grid.
The next X lines will consist of strings of length Y containing only the '.' and '#' characters.
A line with 0 0 will mark the end of input.

```
4 4
....
....
....
....
4 4
....
....
....
...#
4 4
.#..
.#..
.#..
.#..
20 20
....................
....................
....................
....................
....................
....................
....................
....................
....................
....................
....................
....................
....................
....................
....................
....................
....#...............
....................
....................
0 0
```

Output

Each line of output must contain the number of unique paths to the exit for a given grid.

```
20
0
0
35342103720
```

codewars 2017 years

Problem 15 Roll the Bones [9 points]

Summary

Ancient dice were made from the ankle bones of hoofed animals, such as sheep and oxen. These bones are roughly cube-shaped, although two of the sides are rounded and cannot be landed on. Thus, the phrase "roll the bones" refers to rolling dice or, metaphorically, to taking a risk.

Modern dice are often made of plastic (or some other non-bone material) and typically have pips on each face to denote the numbers one through six. When two such dice are rolled together, the most common sum of the pips is seven and the least common sums are two and twelve. We can write a simple program to analyze all thirty-six possible combinations and print a distribution table like this:

```
SUM     COUNT     PERCENT
 2        1        2.77%
 3        2        5.55%
 4        3        8.33%
 5        4       11.11%
 6        5       13.88%
 7        6       16.66%
 8        5       13.88%
 9        4       11.11%
10        3        8.33%
11        2        5.55%
12        1        2.77%
```

But creating such a table for standard dice is too trivial for a rock star programmer like you! For this problem, we'll imagine dice with a variety of number schemes. Each scheme is presented below with names that inspired the distributions:

```
* 1 2 3 4 5 6 Linear
* 0 1 1 2 3 5 Fibonacci
* 0 0 1 1 2 4 Tribonacci
* 1 2 2 3 3 4 Normal
* 1 1 1 2 2 3 Zipf
* 0 1 2 3 5 7 Prime
* 1 2 3 3 4 5 Gaussian
```

Write a program that can compute the distribution table for any combination of two or three dice schemes.

Input

Each line of input has three upper-case letters that represent the dice to be analyzed, followed by three integers. Each letter will be either the first letter of a distribution name or an X. The input ends with three X's and three zeroes. Otherwise, there can be no more than one X per line. If a letter X is present with two other letters, it means that there are only two dice to be analyzed. The integer values are the sums of the pips that your program should use when printing the count and distribution.

```
L L X 6 7 8
T F L 5 7 9
P Z G 4 10 11
X X X 0 0 0
```

Output

For each line of input, the program must print the three letters on one line, followed by three more lines, one for each integer value. The program must print the correct rows of the distribution table for each of the integer sums. To simplify judging, the program must not print the other rows of the distribution table. Integer values must match exactly; percentages must match to within +/- 0.1%.

```
L L X
6   5    13.88%
7   6    16.66%
8   5    13.88%
T F L
5   25   11.5740%
7   31   14.35185%
9   20   9.259259%
P Z G
4   17   7.87%
10   17   7.87%
11   17   7.87%
```

Problem 16 Letter Cubes [10 points]

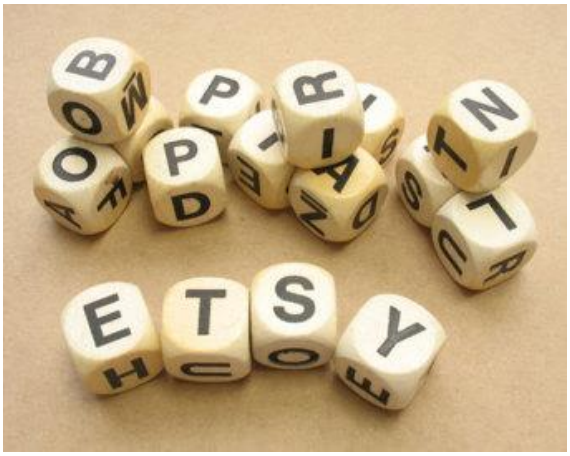
Summary

Word games are fun, right? There are lots of games in which players build words from letters on cubes. The goal of this problem is simple: given a set of letter cubes and a group of words, write a program to determine if each word can or cannot be constructed from the cubes. Your program may re-order and omit cubes as necessary to form words. Also, each of the cubes can only have one side facing up, so only one letter from each of the cubes may be used for any particular word.

Input

The first line of input indicates the number of cubes, N, from three to nine. The next N lines give the letters that appear on the cubes. The following line indicates the number of words, M, from three to ten. And then the next M lines have one word each.

```
4
S X O C A D
O S N Q K M
B G D E L R
U E W I F P
10
CODE
WARS
LONG
DOG
SING
BRING
GLUE
IN
BAD
FORK
```



Output

For each word, the program must print the word and say whether it can or cannot be formed using the cubes.

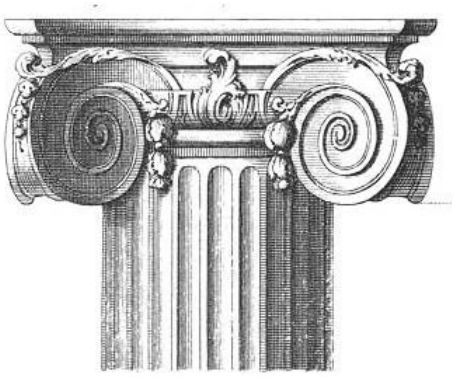
```
CODE can be formed
WARS can be formed
LONG CANNOT be formed
DOG can be formed
SING can be formed
BRING CANNOT be formed
GLUE CANNOT be formed
IN can be formed
BAD CANNOT be formed
FORK can be formed
```

codewars 2017 years

Problem 17 I, For One, Like Roman Numerals [11 points]

Summary

Find the largest value Roman numeral within a string of text. Ignore spaces, punctuation, and non-Roman numeral letters. In our slightly-modified Roman numeral system, a single letter S on the end represents a value 1/2.



Rules about Roman numerals:

- The numeric values of the letters are added up, unless a smaller value precedes a larger value, in which case it is subtracted.
- The symbols "I", "X", "C", and "M" can be repeated three times in succession, but no more. "D", "L", and "V" can never be repeated.
- "I" can be subtracted from "V" and "X" only. "X" can be subtracted from "L" and "C" only. "C" can be subtracted from "D" and "M" only. “S”, "V", "L", and "D" can never be subtracted.
- Only one small-value symbol may be subtracted from any large-value symbol.

Tricky aspects:

- string may contain valid numerals but invalid sequence, i.e., program must ignore invalid combinations of Roman numerals
  - IL is not valid; 49 is XLIX
- not looking for the longest sequence, but the largest value:
  - M is larger than XXIII
  - X is larger than IX

Roman Numeral Table

1	I	14	XIV	27	XXVII	150	CL
2	II	15	XV	28	XXVIII	200	CC
3	III	16	XVI	29	XXIX	300	CCC
4	IV	17	XVII	30	XXX	400	CD
5	V	18	XVIII	31	XXXI	500	D
6	VI	19	XIX	40	XL	600	DC
7	VII	20	XX	50	L	700	DCC
8	VIII	21	XXI	60	LX	800	DCCC
9	IX	22	XXII	70	LXX	900	CM
10	X	23	XXIII	80	LXXX	1000	M
11	XI	24	XXIV	90	XC	1600	MDC
12	XII	25	XXV	100	C	1700	MDCC
13	XIII	26	XXVI	101	CI	1900	MCM

MathATube.com

Input

Lines of text:

```
I locate Max McMillion
requirements: six virtual DVDs
She gave him a live Wham CD
Mad lexicons
```

Output

The largest valued Roman numeral of each line, along with its value:

```
MCMI 1901
MS 1000 1/2
MCD 1400
MDLXI 1561
```

codewars 2017 years

Problem 18 PonyMon [11 points]

Summary

PonyMon is a fun little game in which ponies of various types battle one another for virtual glory. There is a wide variety of ponies, each with their own unique characteristics:

- Name: PonyMon names may be quirky, cute, or intimidating (but in a family-friendly way).
- Type: the types are Earth, Air, Fire, Water, Battle, Nature, Light, Metal, Shadow, and Plasma. A PonyMon may have a weakness and/or resistance to attacks from certain other types.
- Weakness: if the pony is attacked by another pony of this type, the attack's damage is doubled.
- Resistance: if the pony is attacked by another pony of this type, the attack's damage is halved.
- Health Points, or HP: a positive integer (up to 200) that indicates how much damage the pony can receive before being defeated.
- Attacks: Everypony has two attacks. Each attack has three attributes:
  - a name
  - an integer amount (one to five) of power charges required to perform the attack.
  - an integer amount (up to 200) of the damage value for the attack.

At the beginning of the game, two ponies face off against each other in battle. Both ponies begin the game with no power charges and full health points. The game is played in turns. On each pony's turn, it gains one power charge and it can attack once. A pony may only use an attack for which it has sufficient power. The attack reduces the opponent's HP by the attack's damage value. Damage is cumulative and if a pony's health is at or below zero then it loses the battle. Otherwise, the other PonyMon begins its turn.

Write a program to simulate battles between various PonyMon ponies. Gotta catch everypony!

Input

The first line of input indicates the number of PonyMon ponies in the database, followed by one line for each pony. Everypony has a name (with no spaces), type, weakness, resistance, HP, and two attacks. Each attack has a name (no spaces), power requirement, and damage. If a PonyMon has an X for its weakness or resistance, that means the PonyMon does not have that attribute. The next line in the file indicates the number of battles to simulate. Each battle is described on one line, giving the names of the two competitors in the order in which they take their turns.

```
4
KayoTick M P B 130 Lash 3 60 Torment 4 100
Rhinopony B L N 110 Trample 2 30 Skewer 4 90
Valkysiren A M S 70 EnchantingSong 1 30 FlyBy 3 50
TwilightPony P P X 80 Gallop 1 20 PlasmaBeam 4 80
3
Rhinopony Valkysiren
TwilightPony KayoTick
Valkysiren TwilightPony
```

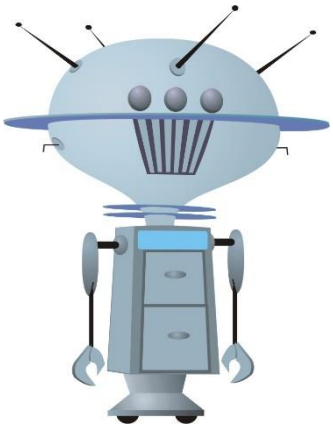
Output

For each battle, the program must print the winner and loser and the winner's remaining health points. For the purposes of this problem, if a pony has enough power charges for both its attacks, it should always use the attack that does the most damage.

```
Valkysiren defeats Rhinopony with 10 HP remaining.
TwilightPony defeats KayoTick with 20 HP remaining.
Valkysiren defeats TwilightPony with 30 HP remaining.
```

Problem 19 Robot Distance Sensor [12 points]

Summary



One of the cool things about robots is that they can have senses that we humans cannot. For example, an electro-optical proximity detector (EOPD) uses visible light to measure distance. The EOPD is not affected by ambient light because it measures reflected light from its own light source. The EOPD sensor is pretty nifty, but some math is required. That's where you come in.

When the robot scans the EOPD sensor, it reads a raw value between 0 and 65535, which represents the amount of source light reflected from a distant surface. Far objects result in a value of 0, and objects very close to the sensor give a reading of 65535. Theoretically, the sensor value v should be related to the distance s by the following formula:

```
v = ( k ^ 2 ) / ( s ^ 2 )
```

We should be able to calculate the distance to an object by solving the equation for s, which gives us `s = k / srqt( v )`. Due to the nature of the EOPD hardware, calculating the distance from the raw formula is slightly inaccurate. So we must correct for this inaccuracy by introducing an error factor into our equation, like this: `s = k / sqrt( v ) - e`.

But what are k and e? They depend on the electrical characteristics of an individual sensor. To calibrate the sensor, we take multiple measurements of v at known distances. This gives us a list of values for v and s. Then we can calculate an approximate value for k from any two measurements, where one distance is far and the other is near.

```
k' = ( sFar - sNear ) / [ (1/sqrt(vFar)) - (1/sqrt(vNear)) ]
```

The final value for k is the average of all the approximations. Next we calculate the error factor for each pair of (v,s) like this:

```
e' = ( k / sqrt(v) ) - s
```

And finally, we calculate e as the average of all the error values e'. Write a program to analyze calibration values so that your robot can accurately convert raw sensor readings into distances.

Input

The first line of input indicates the number of calibration readings (up to 11) and the number of test values (up to 7). Each calibration reading appears on a separate line with the distance first (up to 99), then the raw sensor value. All the test values follow on one line. This data will be used to test your calibration. There may be extra spaces in the input.

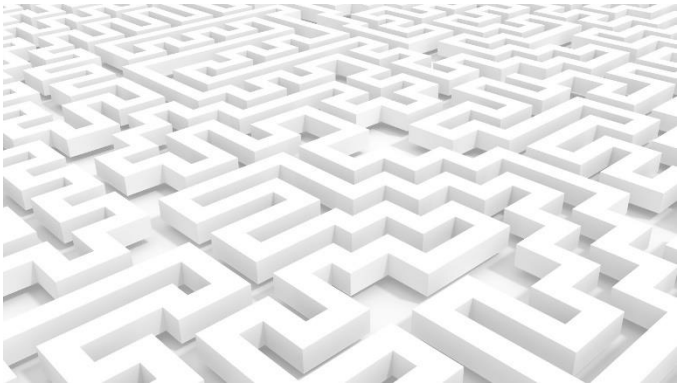
```
4 5
3 44721
5 16123
7 8189
9 4949
32267 7701 60944 12208 4190
```

Output

The program must print (and label) values for k and e. Then, for each sensor test value the program must print the test value and the calculated distance. Each value-distance pair must be printed on its own line. The distances must be accurate to within +/- 0.1.

```
k = 632.5136998722186
e = -0.011741087487370239
32267 3.5209694452114344
7701 7.206730355568299
60944 2.5620276016808723
12208 5.72402509096205
4190 9.769765905820604
```

Problem 20 Yet Another Maze Solver [12 points]



Summary

Here at CodeWars, we just love mazes. If you love mazes as much as we do, then you'll enjoy this problem. Write a program to solve a maze constructed from text characters.

The (non-passable) maze walls are represented by forward slash and backslash characters. Other solid (non-passable) areas are represented by periods. The start point is marked with the @ character and the goal point is marked with an X. Passable regions are represented by spaces. The path through the maze must connect adjacent spaces, where adjacent includes up, down, left, right, and diagonal.

Input

The first line of input gives the width and height of the text maze, up to a maximum of 77 wide and 55 high. The maze itself is given on the next lines.

```
26 9
../\/\..../\../\../\../\/\
..\@ /.../ /..\ \/  \.\  /
../  \../  \/\.\  /\ \/ /.
./ /\/./ /\  /./  \.\  /..
/ /.../  \.\ \/ /\ \.\/./\
\ \../ /\ \.\  /./  \../X/
.\ \/ /./ /./ /./ /\ \/ /.
..\  /./ /./ /..\ \.\  /..
...\/..\/..\/....\/..\/...
```

Output

The program must print the solved maze with the shortest possible solution marked with asterisk characters.

```
../\/\..../\../\../\../\/\
..\@ /.../ /..\ \/  \.\  /
../* \../**\/\.\  /\ \/ /.
./*/\/./*/\* /./**\.\  /..
/*/.../* \.\*\/*/\*\.\/./\
\*\../*/\ \.\**/./ *\../X/
.\*\/*/./ /./ /./ /\*\/*/.
..\**/./ /./ /..\ \.\**/..
...\/..\/..\/....\/..\/...
```

codewars 2017 years

Problem 21 Perfectly Defined [12 points]

Summary

The CodeWars Perfect Dictionary has definitions where the word being defined comes 1st, followed by a hyphen and then the meaning. The word being defined is formed from two words put together, and the meaning explains their relation. The definition is 'perfect', if the word being defined can be created from fragments of words (substrings) in its meaning, provided the fragments are at least two letters long. Here are some examples (the numbers under the definition are there to show the position/index of each letter, but aren't in the actual dictionary/input):

```
hayneedle - a needle lost in a haystack
012345678901234567890123456789012345678901
000000000011111111112222222222333333333344
```

This definition is perfect, because 'hayneedle' can be formed from the substrings at indices 19-21 (hay) and indices 02-07 (needle). Let's look at a different word with the same definition:

```
lockneedle - a needle lost in a haystack
012345678901234567890123456789012345678901
000000000011111111112222222222333333333344
```

This definition is also perfect, because 'lockneedle' can be formed from the substrings at indices 09-10 (lo), indices 25-26 (ck) and indices 02-07 (needle). Now let's look at one more word with the same definition:

```
clockneedle - a needle lost in a haystack
012345678901234567890123456789012345678901
000000000011111111112222222222333333333344
```

This definition is not perfect, because there is no substring with 'cl' in it. While there is a 'c' and an 'l' in the definition, they can't be used because one-character substrings aren't allowed in perfect definitions. Let's look at one last word with the same definition:

```
leackstack - a needle lost in a haystack
012345678901234567890123456789012345678901
000000000011111111112222222222333333333344
```

This definition is perfect, because 'leackstack' can be formed from the substrings at indices 06-07 (le), indices 24-26 (ack) and indices 22-26 (stack). Notice in this example, two of the substring overlap ('ack' and 'stack'), which is perfectly legal. Also, it should be clear that whitespace between words separates substrings and cannot be removed when checking for substrings.
Given a list of definitions, decide which ones are perfect and belong in the CodeWars Perfect Dictionary. Solutions should complete all test cases in less than 30 seconds.

Input

The input will consist of multiple lines; the first line indicates the number of definitions to follow. Subsequent lines each contain one definition, as described above (word being defined, whitespace, hyphen, whitespace, words separated by whitespace that make up the meaning, end-of-line).

```
2
hayneedle - a needle lost in a haystack
clockneedle - a needle lost in a haystack
```

Output

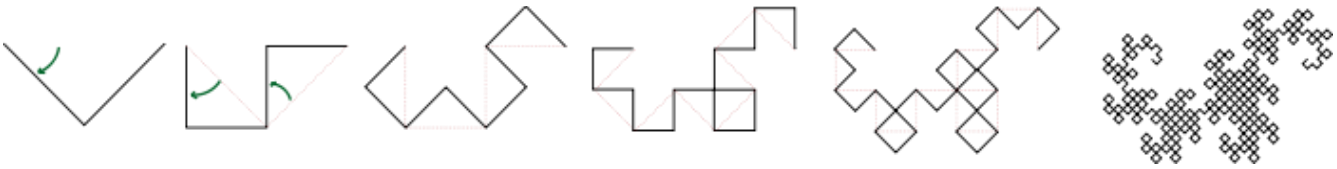
For each definition in the input, print 'Perfect' if the definition is perfect, as defined previously, or print 'Imperfect!', if the definition fails the test for perfection.

```
Perfect
Imperfect!
```

Problem 22 Dragon Fractals [15 points]

Summary

The Heighway Dragon is a fractal curve that is produced by iterative replacement of line segments. It begins with a line segment, a 90-degree left turn, and a second line segment. Then on each iteration, every line segment is replaced by two other line segments: the first at a 45-degree angle from the original segment, and the second at a 90 degree from the first. The pattern start with a right/left substitution. Each substitution afterward is reversed, e.g. right/left, left/right, right/left, left/right. Here are a few iterations of the dragon curve:



We'll print the line segments using character text as shown below. Each segment is three characters in length. The X-axis is stretched by a factor of two with blank spaces for visual effect.

```
                                                  #
                                                #   #
#       #         #   # # #             #       #       #
  #   #           #   #                 #           #
    #             # # #             #       #       #
                                        #   #   #   #
                                          #       #
ITERATION 0     ITERATION 1               ITERATION 2
```

Finally, we'll adopt a convention for numbering directions with the integers 1 through 8 like this:

```
    8
  7 | 1
   \|/
6----*----2
   /|\
  5 | 3
    4
```

This fractal generation technique is called a Lindenmayer system, or L-system, named after biologist Aristid Lindenmayer.

Input

The input consists of two integers on a single line. The first integer is the initial direction of the starting segment in iteration zero. The second integer tells how many iterations the program must produce, up to a maximum of five.

```
7 2
```

Output

The program must print the final iteration of the dragon curve indicated by the input values. With the input limited to five iterations, the output will easily fit within a standard width 80 column console. HINT: You have several sample input and output files -- be sure to test them all!

```
          #       #
        #   #   #   #
      #       #       #
        #           #
    #       #       #
      #   #
        #
```

Problem 23 Vertikleys [15 points]

Summary

Multiple keys must be inserted into a complex lock in order to open a door.

Input

The first line of input indicates the number of keys and the length of each key. There will be three to six keys and the length will be three to seven units long. The following lines of input consist of vertical key-like shapes enclosed by two docks. The keys are labeled with capital letters and the docks are labeled with hash tags. The labels will be aligned over the center bars as shown in the sample input. The keys and docks have greater-than and less-than symbols to indicate the teeth. A space with no tooth is called a cut.

```
3 4
 #    A    B    C    #
[|>  <|   <|    |>  <|]
[|   <|>   |>  <|   <|]
[|>   |>  <|    |    |]
[|   <|    |    |>  <|]
```

Output

The program must re-arrange the keys so that each side forms a complement of the adjacent key or dock. This means that each tooth must face a cut. So, a tooth should not be facing another tooth, nor should a cut be facing another cut. Keys may be rotated or inverted to make them fit. The docks are fixed, which means that they don't move.

```
 #    B    A    C    #
[|>   |   <|   <|   <|]
[|   <|   <|>   |   <|]
[|>   |>   |>   |>   |]
[|   <|   <|   <|   <|]
```

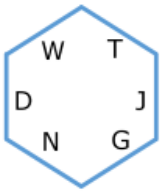
Problem 24 Hexagon Puzzle

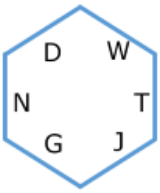
[19 points]

Summary

Every loves a good puzzle. Well, not everyone. But every CodeWars contestant loves a good puzzle! Am I right?

In this puzzle, there are seven hexagons. Each hexagon has letters at its six edges (sides). The goal is to arrange the seven hexagons with one in the center and the other six surrounding it. The letters on the edges of two adjacent hexagons must match. The hexagons may be rotated, but the order of the edge letters must be preserved. For example, the hexagon on the left can be rotated to a new orientation as on the right:





Hexagons cannot be flipped as a mirror image.

Input

The input consists of seven lines, each with six capital letters. Each line represents a single hexagon and each letter is an edge of the hexagon. The letters are listed in clockwise order around the hexagon.

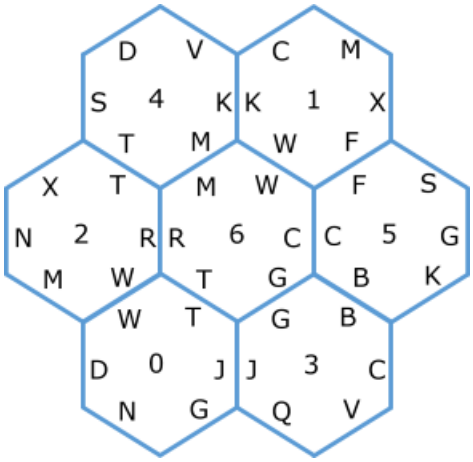
```
W T J G N D
X F W K C M
M N X T R W
B C V Q J G
M T S D V K
C F S G K B
G T R M W C
```

Output

The program must print a solution to the puzzle. There will actually be six possible orientations for the solution – the correct solution will be the one for which the center hexagon is oriented so that its lowest character, alphabetically, is on the right edge. The hexagon sequence must be printed left-to-right, top-to-bottom. The first hexagon in the input is the top left, and the last is the bottom right. After each index, the program must print the letter of the right edge. HINT: test your program with all the sample data.

```
4K 1X 2R 6C 5G 0J 3C
```

For your reference, here is a visualization of the solution for the sample input:



codewars 2017 years