Important Contest Instructions!!

Please read the following instructions carefully. They contain important information on how to run your programs and how to prepare them for submission. If you have any questions regarding these instructions, please ask it in our Discord server (your teacher has the link).

NEW for 2020! Input file name change

IMPORTANT: Starting this year (2020) we are changing the name of the input file. All problems will use the same filename **input.txt**. This file contains the dataset that your solution should be using. If you are reading from a file you must ensure that your solution is reading from **input.txt**. If you are reading from the keyboard (standard in) then you do not need to make any changes.

Program Input

Most programs will require input. You have two options:

1 Read from file

Your program may read the input from a file. The input data will be in a local directory in the file **input.txt**.

2 Read from keyboard input (standard in)

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 **input.txt** to 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 **input.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 < input.txt
%> java -jar js.jar prob01.js < input.txt
%> python prob01.py3 < input.txt
%> prob01.exe < input.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). Please remember to carefully review your file names.

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 (including leading zero, e.g. prob03.java). For Python, use the extension that matches the Python version you are using. Please submit only the source (.java, .js, .py2, .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 (including leading zero, e.g. prob03.exe). All dependencies must be compiled into your final executable.

Watson, I Presume?

problem 00

1 points

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.

On this day in 1876, the United States Patent Office granted a patent for the telephone to Alexander Graham Bell. In tribute, all you need to do is print to the screen the first words ever transmitted by telephone: "Mr. Watson, come here!"

Output

```
Mr. Watson, come here!
```

What's In a Name?

problem 01

1 points

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

Input will consist of a single line with a single word that is your new friend's first name.

```
Harvey
```

Output

Welcome your new friend to CodeWars in the message format shown below, including the name you received as input.

```
Welcome to CodeWars, Harvey!
```

Mirrored Number

problem 02

2 points

You are working on special effects for a movie, and the director wants to see a scene where numbers on a screen are reflected in a mirror. Your task is to program a function that will "mirror" a number.

Given an integer, such as 1234, reverse the number to 4321 so it is the mirror reflection of itself. Note: The maximum length of an integer you would encounter is 30.

Input

Example 1

```
7946
```

Example 2

```
998877665544332211
```

Output

Example Output 1

```
6497
```

Example Output 2

```
112233445566778899
```

Soda Tables

problem 03

2 points

Taylor has X cans of regular soda and Y cans of diet soda. He wants to create some identical refreshment tables for a party at his school. He also doesn't want to have any sodas left over. Write a program to find the greatest number of refreshment tables that Taylor can stock.

You will receive 2 lines of input, the first will be "X" and the second will be "Y".

Input

```
6
15
```

Output

```
3
```

Discussion

The goal is to set out the largest number of tables with identical sets of refreshments, using all of the available cans of soda. To do that, look for the largest number which divides evenly into both X and Y. In this case, X=6 can be divided by 1, 2 and 3 evenly. And Y=15 can be divided by 1, 3, and 5 evenly. So 3 is the largest divisor they both share.

Book 'em, Danno

problem 04

2 points

Troy and Gabriel are shelving books at a public library. Troy shelves X books at a time, whereas Gabriel shelves Y at a time. If they end up shelving the same number of books, write a program to find the smallest number of books each could have shelved.

You will receive 2 lines of input, the first will be Troy's books shelved, the second will be Gabriel's.

Input

```
10
6
```

Output

```
30
```

Discussion

Because you are told that the 2 people shelve the same number of books, and that they finish at the same time, you need to look for lowest multiple of the 2 numbers. E.G. People shelving 5 books at a time, and 6 books at a time will intersect and have shelved the same number of books at books: 30, 60, 90, etc. The lowest number they could both shelve at the same time is 30.

Hint this is not simply a multiplication problem! Run the multiples of the numbers to find the least common one.

Hordak's Boss

problem 05

2 points

Julia is working on an encryption algorithm which uses the factors of prime numbers for the basis of its one-way hashing. The first step she needs to solve for her algorithm is determining if her random number generator (RNG) is sending prime numbers or not. So she needs to code a prime number tester.

Read in a single integer from the "RNG", then test the number to determine if it is Prime or not. A Prime number is only evenly divisible by the number 1, and itself.

Print # is PRIME if the number being tested is Prime. Else, print # is NOT Prime Where # is the number being tested.

Input

```
6
```

Output

```
6 is NOT Prime
```

'Martian' Numerals

problem 06

4 points

How many X's did a Roman Centurion make a day in cold hard Lira? About a C's worth! Turns out, Martians gave Rome the idea for their number system. Use the conversion charts below to help translate some Martian numbers!

Note, that unlike the Roman Numerals, Martian Numerals reuse symbols to mean different values. B can either mean '1' or '100' depending on where it appears in the number sequence.

B's and W's and Z's, oh my!

Arabic	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**20**	**30**	**40**	**50**	**60**	**70**
Martian	B	BB	BBB	BW	W	WB	WBB	WBBB	BK	Z	ZZ	ZZZ	ZP	P	PZ	PZZ

Arabic	**80**	**90**	**100**	**200**	**300**	**400**	**500**	**600**	**700**	**800**	**900**	**1000**
Martian	PZZZ	ZB	B	BB	BBB	BG	G	GB	GBB	GBBB	BR	R

Input

You will receive a list of numbers in a data file, one number per line, up to 5 lines at a time (with a minimum of 1 line). No number will exceed 1000, or be less than 1.

```
26
67
186
408
500
```

Output

You will need to convert the numbers from Arabic (1,2,3...10...500...1000) to Martian (B,BB,BBB...Z...G...R) numerals.

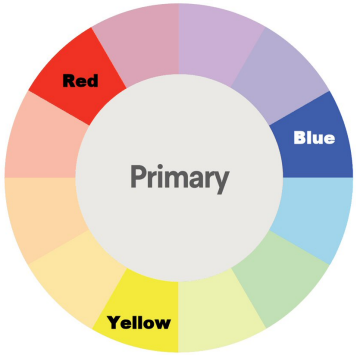
```
ZZWB
PZWBB
BPZZZWB
BGWBBB
G
```

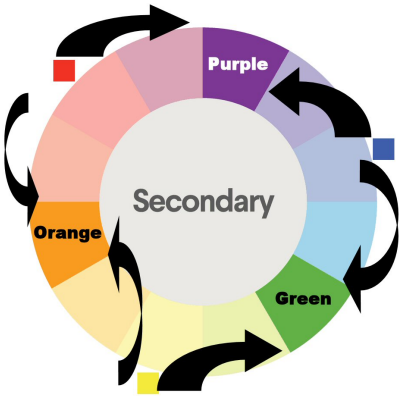
Color Mixer

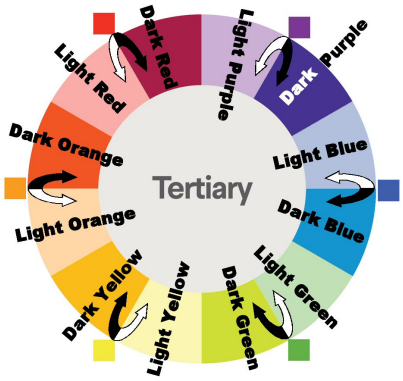
problem 07

4 points

ROYGBIV isn't just an acronym, it's a way of life for your paint company. The owner is considering modernizing her paint mixing equipment with a computerized model. She's hired you to code the prototype. Your simple program will need to correctly output the right color based on the blends she's given you.







Input

You will receive one to five lines of color combinations consisting of primary colors and secondary colors as well as black and white to make "dark" and "light" colors. The full science of colorization and pigments will be implemented next, if your prototype is successful.

```
RED YELLOW
BLUE WHITE
YELLOW YELLOW
ORANGE BLACK
```

Output

Your program should output the correct color depending on what two colors were "mixed" on the line. Primary colors should mix together to create secondary colors. Anything mixed with "WHITE" or "BLACK" should be output as either "LIGHT X" or "DARK X" where X is the color "WHITE" or "BLACK" were mixed with. Anything mixed with itself won't change colors. You are guaranteed not to receive incompatible colors, or colors not listed in the color wheels shown above (aside from "WHITE" and "BLACK").

```
ORANGE
LIGHT BLUE
YELLOW
DARK ORANGE
```

Text Wrap

problem 08

4 points

You are tasked with writing a program that takes a text document as input, and reformats the document so the lines are of reasonable length. Specifically, you will find good places to insert line breaks (`\n` character) to make the optimal line lengths.

For example, given the input:

```
You are tasked with writing a program that takes a text document as input, and reformats the
document so the lines are of reasonable length.
```

Your program will output:

```
You are tasked with writing a program that takes a text document as input, and
reformats the document so the lines are of reasonable length.
```

Follow these simple rules:

- A line must not be longer than 80 characters before a line break occurs
- A line break (`\n`) may only be inserted at a an existing word break, as denoted by a space character (). In other words, do not add a line break in the middle of a word!
- The program should maximize possible line length while following these rules
- Preserve existing line breaks
- Remove surrounding whitespace on each line of output (ig. a line should end with `brown cow\n` not `brown cow \n`, and should begin `hi there` not `hi there`)

Input

```
The scenery of Walden is on a humble scale, and, though very beautiful, does not approach to
grandeur, nor can it much concern one who has not long frequented it or lived by its shore; yet
this pond is so remarkable for its depth and purity as to merit a particular description.
```

Output

```
The scenery of Walden is on a humble scale, and, though very beautiful, does not
approach to grandeur, nor can it much concern one who has not long frequented it
or lived by its shore; yet this pond is so remarkable for its depth and purity
as to merit a particular description.
```

Vinyl Records

problem 09

5 points

With vinyl records making a comeback, industry executives need to hire a programmer to tell their factory machines when to flip the record, as all of their original equipment was destroyed during the rise of the CD. Help your company turn the tables on the industry!

Input

The factory machine will receive a single data line composed of two integers. The first integer is the total number of minutes for the recording. The second integer is the total number of seconds. You are guaranteed to only get one line of data per file, and that line is guaranteed to always have two, and only two, integers on it. When you reach the line with double zeros on it, stop reading more inputs.

Examples

```
17 5
26 33
51 1
0 0
```

Output

The vinyl records your company presses can hold 25 minutes of audio on one side. Output to the data operator's screen the remaining time on the record as follows:

```
Time remaining 32 minutes and 55 seconds
Time remaining 23 minutes and 27 seconds (we'll need both sides)
Time remaining -1 minutes and -1 seconds (we're gonna need a bigger record)
```

If the recording will exceed the amount of time one side can contain, add " (we'll need both sides)" to the output for the operator to see. However, if the recording will exceed the total amount of time an entire record can contain, add "(we're gonna need a bigger record)" to the output instead.

Mad Libs

problem 10

5 points

Your first job out of college is for a children's book publisher. They are currently publishing a line of "mad libs" books, and they need you to check their print-copy. Write a program that will take a sentence with placeholders in it, and substitute in for those placeholders words from the lists of example words the publisher has provided.

Input

Line one will always be the line with the story. The order of the Nouns, Adverbs, Verbs and Adjectives will always be the same. Input stops when it reaches the word END alone on a line. You are guaranteed to always have enough words to substitute into the sentence TWICE (not repeating any words). The number of words given under each section will be random. The data file you receive will always be in the format of:

```
Story sentence
NOUNS
noun
ADVERBS
adverb
VERBS
verb
ADJECTIVES
adjective
END
```

Your goal will be to substitute the appropriate words into the sentence from the list of provided words TWICE (keeping track of the words you have used) where:

```
[N] = nouns
[AV] = adverbs
[V] = verbs
[AJ] = adjectives
```

Example

```
There once was a [AJ] [N] from [N] who [AV] [V] all day.
NOUNS
squirrel
London
tree
lettuce
ADVERBS
silently
quickly
happily
VERBS
skipped
ran
climbs
ADJECTIVES
delightful
homely
quaint
END
```

Output

Always take the TOP-most UNUSED word from a group when substituting. E.G. from the example Nouns list, use: "squirrel," then "London," then "tree," etc. Output the final sentences when all substitutions are done.

```
There once was a delightful squirrel from London who silently skipped all day.
There once was a homely tree from lettuce who quickly ran all day.
```

Is it a Cyclops number? "Nobody" knows!

problem 11

5 points

Given an integer input, figure out whether or not it is a Cyclops Number.

What is a Cyclops number, you may ask? Well, it's a number whose binary representation only has one 0 in the center!

Examples:

Input	Cyclops?	Binary	Explanation
0	yes	0	only one zero at "center"
1	no	1	contains no zeroes
5	yes	101	only one zero at center
9	no	1001	contains two zeroes (even though both are at the center)
10	no	1010	contains two zeroes
27	yes	11011	only one zero at center
85	no	1010101	contains three zeroes
101	no	1100101	contains three zeroes
111	no	1101111	only one zero, not at center
119	yes	1110111	only one zero at center

Input

The input consists of up to seven lines, each line having an integer in the range zero to one billion, inclusive.

```
304
90211
27
730193
130815
119
987654321
```

Output

For each input line, the program should print the integer and a 'yes' or 'no' to indicate if the integer is a cyclops number.

```
304 no
90211 no
27 yes
730193 no
130815 yes
119 yes
987654321 no
```

Metric Meltdown

problem 12

5 points

You're building a ramp to practice your mad skating skills, but your local lumber yard only takes measurements in metric units! All of the measurements you took were in US-Imperial units. Quick! You need to convert those measurements before the store closes!

Input

You will receive up to 3 numbers, separated by spaces. The left-most number is yards. A second number on the line will be feet. A third number on the same line will be inches.

```
1 1 1
5
1 2
0 7
0 0 9
```

Output

Convert the yards (+ feet + inches) to centimeters. Round answers to 2 decimal places. One foot (ft) is equal to 0.3048 meters (m)
One inch (in) is equal to 2.54 centimeters (cm)

```
124.46
457.20
152.40
213.36
22.86
```

Do I Have Enough?

problem 13

5 points

Great news! You get to go to Japan on a class trip! Bad news, you don't know how to use the Yen which is the name of the Japanese cash system. Japan uses coins for cash a lot more than the United States does. Yen comes in coins for values of: 1, 2, 10, 50, 100, & 500 To practice your Yen skills, you have selected random items from Amazon.co.jp and put them into a list along with their prices in Yen. You now want to create a program to check your Yen math.

Your goal is to maximize your buying power to buy AS MANY items as you can with your available Yen.

Input

You will receive a datset file listing 2 to 6 items in the format of:

```
ITEM YYYYY
```

Where:

```
ITEM = the name of the item you want to buy
YYYYY = the price of the item (in Yen)
```

The very first line of the dataset file will be the amount of Yen you have in your "pocket," and a space followed by the number of items in the list. Full example dataset file:

Example: Affordable

```
6000 3
Phone-case 1486
Candybar 863
Sunglasses 5529
```

Example: Unaffordable

```
2100 3
Camera 69555
TV 76439
iPhone 90000
```

Output

List the items you can afford to buy. Each item on its own line. Your goal is to buy as many items as possible. If you can only afford the one expensive item, or 2 less expensive items on a list, but not all three, then list the less expensive items as affordable. If you cannot afford anything in the list, output "I need more Yen!" after the items. The final line you output should be the remaining Yen you will have left over after you make your purchases.

Output: Affordable

```
I can afford Phone-case
I can afford Candybar
I can't afford Sunglasses
3651
```

Output: Unaffordable

```
I can't afford Camera
I can't afford TV
I can't afford iPhone
I need more Yen!
2100
```

Equal Sum Numbers

problem 14

5 points

Your friend is a math geek. He likes to play with numbers and has given you a puzzle to solve. You are required to find all numbers within a given range which have equal sum of digits between the even and odd indexes of the number.

Write a program to solve the puzzle.

Discussion

To determine if a number's digits are equal between the even and odd indexes, start counting from the right of the number at the One's column as index zero, and work left. Count index zero as even. So, if examining 101, look at it like this:

Number	**1**	**0**	**1**
Index	2	1	0

In this case the sum of the even indexes would be 1 (index 0) + 1 (index 2) = 2 And the sum of the odd indexes would be 0 (index 1) + nothing = 0 So, 101 would NOT have equal sums.

If examining 6721, look at it like this:

Number	**6**	**7**	**2**	**1**
Index	3	2	1	0

In this case the sum of the even indexes would be 1 (index 0) + 7 (index 2) = 8 And the sum of the odd indexes would be 2 (index 1) + 6 (index 3) = 8 So, 6829 WOULD have equal sums.

Input

Your program's input is:

```
N1
N2
```

Note: N1 and N2 may be single digit numbers but you must be aware that the problem only focuses on numbers with digits >=2.

Example 1

```
100
150
```

Example 2

```
365
374
```

Example 3

```
800
1500
```

Output

The output should check for N1 but not N2 for the equal sum requirement.

Example Output 1

```
110 121 132 143
```

Example Output 2

```
No Numbers found with Equal Sum in the given range!!
```

Example Output 3

```
880 891 990 1001 1012 1023 1034 1045 1056 1067 1078 1089 1100 1111 1122 1133 1144 1155 1166 1177 1188 1199 1210
1221 1232 1243 1254 1265 1276 1287 1298 1320 1331 1342 1353 1364 1375 1386 1397 1430 1441 1452 1463 1474 1485
1496
```

Counting Duplicates

problem 15

6 points

Counting duplicate items in two lists applies to many practical uses, such as identifying common factors in medical patients or measuring similarity of two documents (or programs, or DNA strands). A word is a duplicate if it appears in both lists. Write a program to count the number of duplicate words in two lists of words. The program must ignore letter case when comparing words.

Input

The first line of input contains two integers, which are the numbers of words listed on the two following lines. All punctuation has been removed (commas, periods, apostrophes, etc.) for ease of comparison. The next line after the word lists is another pair of integers. If non-zero, the integers signify another pair of word lists. The input ends if both integers are zero.

```
4 5
the frog must win
the fly and the frog
14 16
Two roads diverged in a wood and I took the one less traveled by
I carved two rods of yellow wood and hid the one less pleasing to my eye
17 15
The free bird leaps on the back of the wind and floats downstream till the current ends
The caged bird sings with fearful trill of the things unknown but longed for still
16 16
Success is counted sweetest By those who never succeed To comprehend a nectar Requires sorest
need
Success is not final failure is not fatal it is the courage to continue that counts
13 15
My love is like a red red rose thats newly sprung in June
At dawn my love awoke and searching found a scarlet rose wet with morning dew
0 0
```

Output

For each list pair, the program should print three lines of output. First it should print the two lists on two separate lines. On the third line it should print the number of duplicates and the list of duplicates. The CaSe and order of the duplicates is not important.

```
the frog must win
the fly and the frog
2 frog the
Two roads diverged in a wood and I took the one less traveled by
I carved two rods of yellow wood and hid the one less pleasing to my eye
7 I Two and less one the wood
The free bird leaps on the back of the wind and floats downstream till the current ends
The caged bird sings with fearful trill of the things unknown but longed for still
3 The bird of
Success is counted sweetest By those who never succeed To comprehend a nectar Requires sorest
need
Success is not final failure is not fatal it is the courage to continue that counts
3 Success To is
My love is like a red red rose thats newly sprung in June
At dawn my love awoke and searching found a scarlet rose wet with morning dew
4 My a love rose
```

Math Checker

problem 16

6 points

Your old sixth grade teacher needs help grading math homework! He's made a few mistakes grading papers lately, and wants to make sure his math is correct. Write a simple program that will take an equation setup, and an answer and determine if the answer is correct or not.

Input

Your program needs to handle positive and negative numbers, and care should be taken to translate integer math to decimal output. Your program needs to be able to read in up to five lines of data in the format of:

```
A# B# operation answer#

Examples:

100 7 DIVIDE 5.0
7 7 MULTIPLY 49.0
-6 3 POWER 216.0
3 7 SUBTRACT 6.0
3 7 ADD 12.0
```

Your program needs to be able to handle the following operations:

```
POWER
MULTIPLY
DIVIDE
ADD
SUBTRACT
```

Operations should be carried out left to right A# to B# as follows:

```
A# ^ B# (A raised to the power of B)
A# * B# (A multiplied by B)
A# / B# (A divided by B; note the teacher does not give divide by zero problems)
A# + B# (A added to B)
A# - B# (B subtracted from A)

Note: either the value for A or B can be negative, except for power (A^B) equations, where B
will always be non-negative.
```

Output

Output all numbers rounded to 1 decimal place. If the student's answer is correct, output:

```
answer# is correct for A# operation B#
```

If the student's answer is incorrect, output:

```
A# operation B# = correct answer, not answer#

Examples:

100.0 / 7.0 = 14.3, not 5.0
49.0 is correct for 7.0 * 7.0
-6.0 ^ 3.0 = -216.0, not 216.0
3.0 - 7.0 = -4.0, not 6.0
3.0 + 7.0 = 10.0, not 12.0
```

Who Is My Neighbour?

problem 17

6 points

Your geography teacher has given you homework to find the neighbours of a city. The teacher explained that a neighbour to a city can be directly or indirectly connected to the city via road or train but not by air. The teacher is quite helpful and has provided her students with the road/train connectivity. Write a simple program to output all the neighbours (direct or indirect) of a given city using this information.

Note: If any two cities are connected by air as well as road or train then the cities are considered to be neighbours. Cities are considered to not be neighbours only in the case where they are solely connected by air.

Input

Your program needs to read in data in the following format:

```
Number of cities
X
Y
Z
...
City X connected to City Y by [train/road/air]
....
```

Note: Your program should be able to handle duplicate records. For example:

```
City X connected to City Y by [train/road/air]
City Y connected to City X by [train/road/air]
```

Example 1

```
3
X
Y
Z
City X is connected to City Y by train
City Y is connected to City Z by air
City Z is connected to City X by road
```

Example 2

```
7
X
Y
Z
A
B
C
D
City X is connected to City Z by train
City X is connected to City Y by air
City A is connected to City C by air
City Y is connected to City Z by road
City Z is connected to City B by train
City B is connected to City Y by road
City C is connected to City D by road
City D is connected to City C by train
```

Output

The order in which neighbours are displayed should be in the order the cities were described. This means: City Z is neighbour to Cities X,Y is accepted but not City Z is neighbour to Cities Y,X

Example 1 Output

```
City X is neighbour to Cities Y,Z
City Y is neighbour to Cities X,Z
City Z is neighbour to Cities X,Y
```

Example 2 Output

```
City X is neighbour to Cities Y,Z,B
City Y is neighbour to Cities X,Z,B
City Z is neighbour to Cities X,Y,B
City A is remote and has no neighbours!
City B is neighbour to Cities X,Y,Z
City C is neighbour to Cities D
City D is neighbour to Cities C
```

Clock Hands' Difference

problem 18

6 points

Would it be handy to keep track of "how wide is the angle between the minute hand and hour hand" to describe time? Let's try to write a program that finds the shorter angle between the hour hand and minute hand in an analog clock and see for ourselves.

Calculating degrees

Given H hours and M minutes time:

- Degrees travelled by hour hand would be: `H * (360/12) + M * (360/(12 * 60))` i.e. `(H * 30 + M * 0.5)`
- Degrees travelled by minute hand would be: `M * (360/60)` i.e. `M * 6`

Input

The input is of the form:

- HH:MM if H >= 10,
- H:MM if H < 10

Time is specified in 24 hour format, that is:

- 0 <= H <= 23
- 0 <= M <= 59

Example

```
5:30
12:00
2:20
9:05
15:45
```

Output

Note: The angle should be displayed with a precision of 2 decimal places.

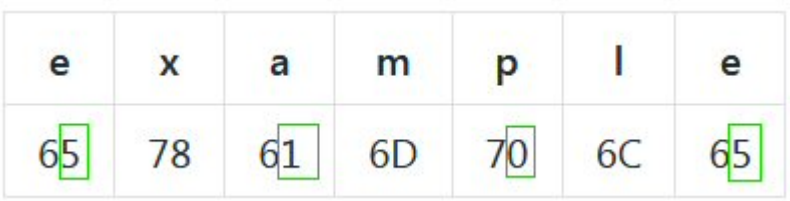
Example Output

```
The angle between the Hour hand and Minute hand is 15.00 degrees.
The angle between the Hour hand and Minute hand is 0.00 degrees.
The angle between the Hour hand and Minute hand is 50.00 degrees.
The angle between the Hour hand and Minute hand is 117.50 degrees.
The angle between the Hour hand and Minute hand is 157.50 degrees.
```

Hidden Hex Message

problem 19

7 points



Steganography is the science of hiding messages in pictures or text. You have been hired to write a program which will decode words hidden in a popular daily word game, to help a freedom fighter smuggle information out of their oppressive homeland. The steganographic code you have agreed on is that every 4th character of the hex code of the letters in the puzzle will be part of the secret message.

Letter Hex Codes:

ASCII	**a**	**b**	**c**	**...**	**i**	**j**	**k**	**...**	**o**	**p**	**q**	**...**	**w**	**x**	**y**	**z**
Hex	61	62	63	...	69	6A	6B	...	6F	70	71	...	77	78	79	7A
ASCII	**A**	**B**	**C**	**...**	**I**	**J**	**K**	**...**	**O**	**P**	**Q**	**...**	**W**	**X**	**Y**	**Z**
Hex	41	42	43	...	49	4A	4B	...	4F	50	51	...	57	58	59	5A

Input

You will receive two lines in your data file. The first will be the puzzle sentence. The second will be the question or prompt which the secret message will answer.

```
bump ow fun waps wept guy
What fixes everything?
```

Decoding:

e	**x**	**a**	**m**	**p**	**l**	**e**
65	78	61	6D	70	6C	65

Every 4th character starting from the right: 5105 (You can also approach the problem from the left, but if you do that, you need to take into account odd vs. even in your algorithm)

Output

Convert the puzzle sentence into hex code for each letter (leave the spaces as spaces). Then output the rightmost character from the hex code, then output every 4th character after that (ignoring spaces), working left. Then combine those characters in groups of 2, and output the corresponding ASCII letter to see the secret message.

```
62 75 6d 70 6f 77 66 75 6e 77 61 70 73 77 65 70 74 67 75 79
5075707079
Puppy
```

Pizza Toppings

problem 20

7 points

You've been hired by a local pizza place to help them save money on toppings, and to also make their assembly line more effecient. You need to write a computer program which can input an order of pizzas, and output the total number of toppings which will be needed. The output from your program will be fed into an automatic toppings dispensor which the employees will then put onto the pizzas in the order.

Input

You will receive a list of pizzas in an order. Each order will have at least one pizza on the order, and as many as 50 for very large orders for parties. Each pizza will display in the format of:

```
# TYPE-OR-TOPPING
```

Or

```
# 1/D TYPE-OR-TOPPING & 1/D TYPE-OR-TOPPING
```

Where # is the number of pizzas to make of that type/topping,
and D is the optional divisor (your restaurant only accepts orders for whole pizzas, 1/2s or 1/4s (which must add up to a whole); if not given then D by default is 1-whole) if the pizza will be split into sections of different types/toppings. The available toppings are:

```
Pepperoni:   32 pieces for a whole pizza
Red Peppers: 16 pieces for a whole pizza
Pineapple:   84 pieces for a whole pizza
Olives:      20 pieces for a whole pizza
Sardines:    12 pieces for a whole pizza
Onion:       28 pieces for a whole pizza
Sausage:     40 pieces for a whole pizza
Ham:         36 pieces for a whole pizza
```

Additionally, several named "types" of pizzas can also be ordered. Any toppings listed for named "type" of pizza will use the full count as listed above:

```
Hawaiian:    pineapple & ham
Combo:       red peppers & olives & onion & sausage
Fishaster:   sardines & onion
Meat-Lovers: pepperoni & sausage & ham
Cheese:      no toppings
```

Example order

```
2 Pepperoni
1 1/4 Pepperoni & 1/4 Sausage & 1/4 Olive & 1/4 Ham
5 1/2 Hawaiian & 1/4 Pepperoni & 1/4 Sardines
3 Combo
1 Cheese
```

Output

Output a simple count of each topping that is needed, in total, to make all of the pizzas on the order.

```
Pepperoni: 112
Red Peppers: 48
Pineapple: 210
Olives: 60
Sardines: 15
Onion: 84
Sausage: 130
Ham: 99
```

Time Clock

problem 21

8 points

Math can be tricky. You have been hired to fix the timeclock calculations for a company whose prior developer failed to account for the differences in units when calcuating the amount an employee earns for a day.

Input

You will receive the data from the timeclock application in the format of:

```
EMPLOYEE NAME (string)
RATE ##.## (double)
IN #### (int)
OUT #### (int)
IN #### (int)
OUT #### (int)
```

The number of employees in the data file can number as few as one, and up to ten in a single data file. Each employee's data for the day will always contain 6 lines in the standard format (even if they only work a half-day and leave after lunch, their in-2 & out-2 times will show times of '0000')

Example

```
EMPLOYEE JANE
RATE 12.00
IN 0800
OUT 1200
IN 1230
OUT 1630
EMPLOYEE MIKE
RATE 09.99
IN 0917
OUT 1200
IN 1303
OUT 1700
EMPLOYEE MICHELLE
RATE 15.00
IN 0730
OUT 1200
IN 0000
OUT 0000
```

Output

Run the calculations for each employee's day and output the amount they earned by multiplying the time they worked by their hourly rate. Use the employee's name, and format their day's earnings into the format of $#.00 (rounded to 2 decimal places)

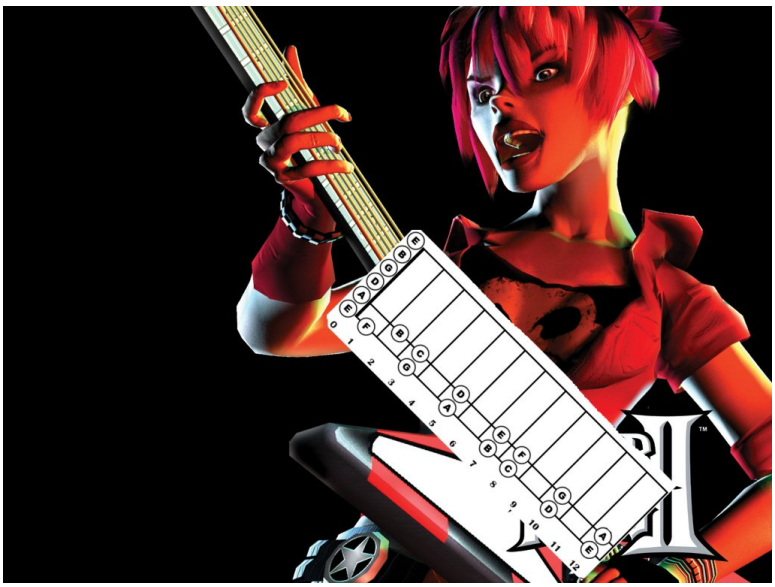
```
JANE earned $96.00
MIKE earned $66.60
MICHELLE earned $67.50
```

Guitar Scale Hero

problem 22

8 points

Quick, your lead guitarist for your band has forgotten her scales. Write a program that will either output the NEXT note given a fret and a string, or (if given a Note name) will output all fret+string combination that will produce that note. Luckily for you, your punk band's setlist for your next gig only uses the E and A strings.



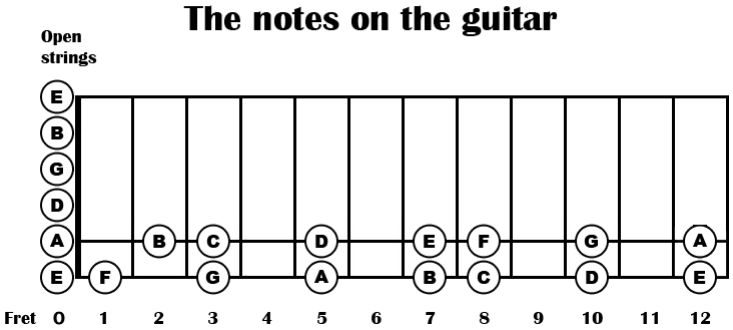
Input

You will receive either a fret number and a string name (separated by a space) or you will receive a Note name. If you receive a fret and a string, output the NEXT note in the scale (working from fret zero (open) towards fret 12; if the next fret would go past 12, return to the "open string" /zero fret).

If you receive a Note name, output the fret and string (on both strings, list the E string first) the note can be played.

Notes:

- 0 is used to represent the "open string", e.g. the top of the fret board.
- For those of you with musical backgrounds, we're ignoring the sharps and flats, skip over them like they aren't there.
- If the "next" note called for is past fret 12, you will need to "return" to the zero/"Open" fret, but the note will stay the same. E.G. A on fret twelve will also be A on fret zero.



Example

```
0 E
2 A
8 E
12 A
D
C
```

Output

```
F
C
D
A
10 E 5 A
8 E 3 A
```

Down, Oh You Mop!

problem 23

8 points

This problem is a Palindrome with a twist -- or rather, a flip! You probably already know that a palindrome is a phrase or sentence in which the order of the letters is the same forward and backward (ignoring spaces and punctuation), as in "Madam, I'm Adam." Palindrome problems are standard fodder for programming competitions, but we're going to turn that frown upside down! Specifically, we're going to rotate our sentences 180 degrees to see if they read the same in both orientations.

To match, the letters of the original sentence must be the rotational complements of the letters in the rotated sentence. All letters should be converted to lower-case before comparing. There are two classes of rotational complements:

- letters that resemble themselves after rotation: o, s, x, and z
- letters that resemble other letters after rotation: a/e, b/q, d/p, h/y, m/w, and n/u

For example, consider the phrase, "Down, oh you mop!". If you convert the letters to lower-case and remove the punctuation, the remaining string is "down oh you mop". After being rotated 180 degrees, and depending on the font used, it looks like this: "dow noh yo umop". You may need to turn this paper upside down to "see" the rotation effect.

Input

The first line of input is an integer (up to 20) indicating the number of lines that follow. Each of the following lines is to be tested for its down-oh-you-mop-isity. The maximum line length is 80 characters.

```
8
Down, oh you mop!
a snow mouse
Peppem hue was a pun, and so many wedded
sundown puns
Queen B
Swamp ahoy, Edwems.
Max weeps 'em
spaam weeds
```

Output

For each line of input, the program should print the input line (CaSe and punctuation optional) followed by a classification in parentheses, either the word "is" or the word "not", according to whether the input line is a down-oh-you-mop. For debugging, it may be useful to print the rotated line after the classification -- but this is not required.

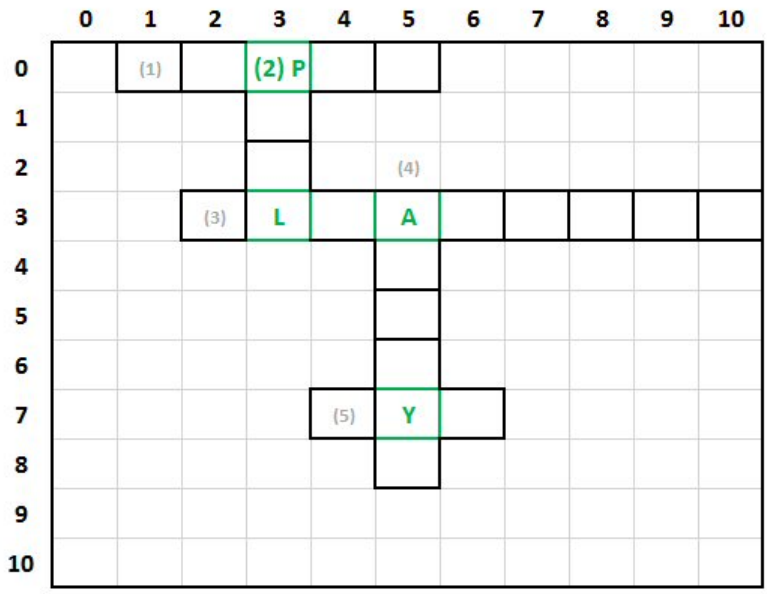
```
down oh you mop (is) dow noh yo umop
a snow mouse (is) asnow mous e
peppem hue was a pun and so many wedded (not) pappam huew os pue und e sem any waddad
sundown puns (not) sund umopuns
queen b (not) q uaanb
swamp ahoy edwems (is) swampa hoye dwems
max weeps em (not) wa sdaam xew
spaam weeds (is) spaam weeds
```

Crossword

problem 24

10 points

The regional crossword puzzler competition is coming up, and your team is in training. In order to practice for the competition, your team is making their own puzzles. The puzzle grid is always a 11x11 grid, with (0,0) in the upper left corner. As the only programmer on your team, it falls to you to sanity check the puzzles to make sure everything fits.



Input

You will receive a 3-part dataset separated by 6 hyphens on a line alone "------". The first section is the list of spaces for the puzzle in the format of: N V #1 #2 #3 where:

```
N = the number of the word (from the
puzzle)
V = orientation (H: horizontal, V:
vertical)
#1 = Length of the space
#2 = X coordinate of start of space
#3 = Y coordinate of start of space
```

The second section lists the coordinates of given letters in the puzzle in the format of: #1 #2 C where:

```
#1 = X coordinate of the letter
#2 = Y coordinate of the letter
C = the letter
```

The third section lists all possible words for the puzzle. Here is an example of a complete dataset:

Example

```
1 H 5 1 0
2 V 4 3 0
3 H 9 2 3
4 V 6 5 3
5 H 3 4 7
-------
3 0 P
3 3 L
5 3 A
5 7 Y
-------
PICKLE
APPLE
WALRUS
PEEL
ALMAMATER
COLLEGE
ALWAYS
AYE
```

Output

Your program should output the number of the puzzle space, and the word that fits in the space both by length and which matches the given known letters from section 2 of the dataset. List the output in ascending (smallest to largest from the top) order. List all space numbers with a leading zero if they are not 2 digits long

```
01 is APPLE
02 is PEEL
03 is ALMAMATER
04 is ALWAYS
05 is AYE
```

Jump Space Map

problem 25

14 points

A classic sci-fi pen & paper RPG uses hexagon maps for star systems across the galaxy. For simplicity -- it is a game, after all -- the star systems are mapped in two-dimensional space with each hexagon being one parsec across. The hexagons are numbered along the X and Y axis. The X and Y coordinates are printed as a four-digit code: the first two digits are X and the second two are Y.

The game has thousands of star systems spanning more than a hundred parsecs of space. The game allows starships to travel faster than light using a sci-fi technology called a "jump drive". Basic Jump drives can travel only one parsec. Advanced (and more expensive) drives can travel up to six parsecs. So players need to know if another star system is within range of their jump drive from their current location.

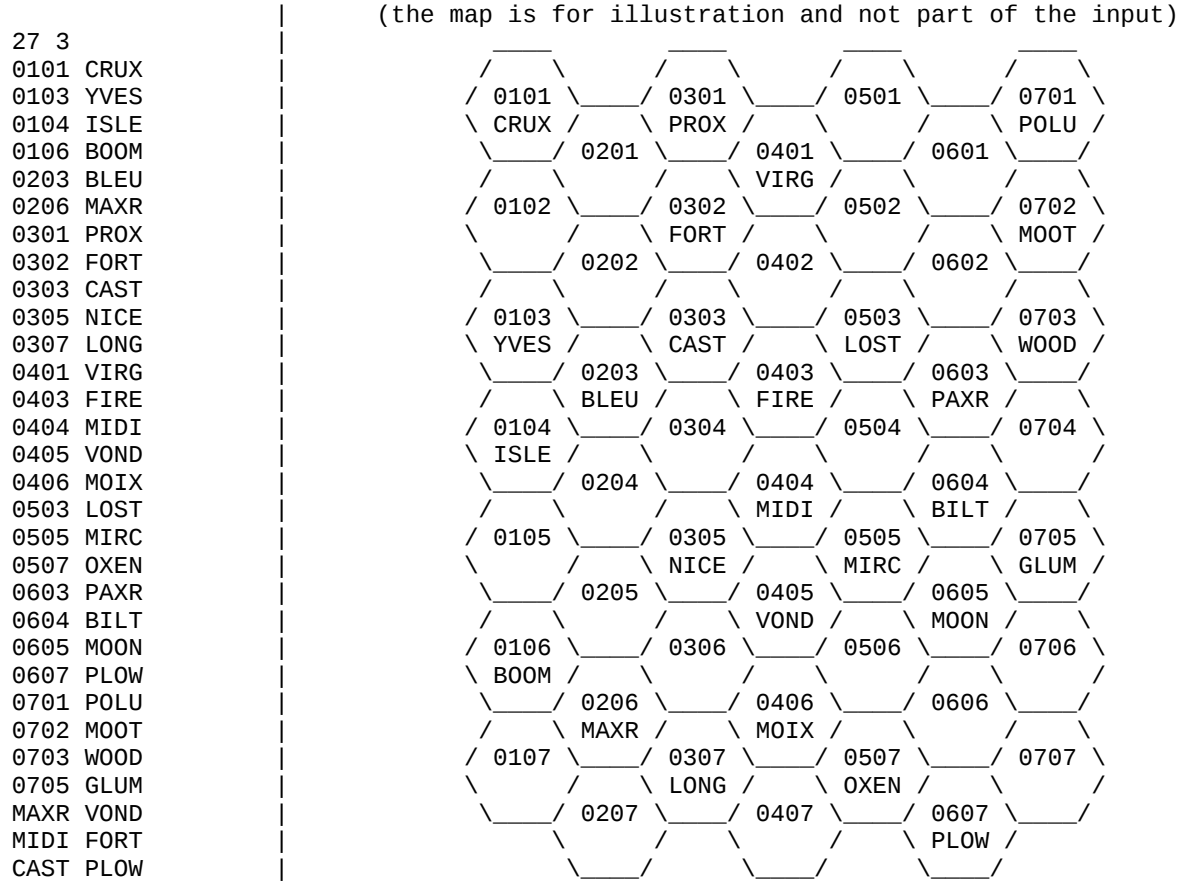
Write a program to read a list of star systems and then generate distances for a set of queries.

Input

The first line of input gives the number of planetary systems and the number of queries. Each star system is represented on a single line with a four-digit code for the X and Y coordinates of the hexagon and a four-letter code name for the system. All coordinates are positive integers up to 99. Each query is represented on a single line with the code names of the departure and destination star systems.

```
27 3
0101 CRUX
0103 YVES
0104 ISLE
0106 BOOM
0203 BLEU
0206 MAXR
0301 PROX
0302 FORT
0303 CAST
0305 NICE
0307 LONG
0401 VIRG
0403 FIRE
0404 MIDI
0405 VOND
0406 MOIX
0503 LOST
0505 MIRC
0507 OXEN
0603 PAXR
0604 BILT
0605 MOON
0607 PLOW
0701 POLU
0702 MOOT
0703 WOOD
0705 GLUM
MAXR VOND
MIDI FORT
CAST PLOW
```

Map Based on Sample Input Data
(the map is for illustration and not part of the input)



Output

For each query line, the program should print the two star system code names and the integer distance between them, in parsecs.

```
MAXR VOND 2
MIDI FORT 3
CAST PLOW 6
```

Countdown Clock

problem 26

15 points

You're going skiing in Nagano! You created a countdown clock so you can see just how long you have to wait until your trip. Now your friends have seen your coundown clock, and they want their own! You decide to generalize your countdown clock code to pull the start and end date from a configuration file, so anyone can use it and put in their own custom dates and times.

Formatting

Your program needs to handle a dataset of dates and times, with one beginning date/time and ending date/time per line. The lines will be in the format of:

```
yyyy-MM-dd HH:mm:ss yyyy-MM-dd HH:mm:ss DHMS
```

The leftmost date and time should be taken together to be the "start" date, and the rightmost date and time are the "end" date. The DHMS tell you how to output the remaining time. You are guaranteed to receive at least one "D" or "H" or "M" or "S" (possibly up to all 4 together). The D,H,M,S output format flags may be listed in any order.

```
D = output total remaining whole days (ignoring hours, minutes, seconds)
H = output total remaining hours (ignoring days, minutes, seconds)
M = output total remaining minutes (ignoring days, hours, seconds)
S = output total remaining seconds (ignoring days, hours, minutes)
```

The output formats can also be combined (in which case they need to subtract from the output any time already output as part of a larger "unit". E.G. if 49 hours are remaining, and the output asks for DH output:

```
2 days 1 hour, taking into account the 24 hours/day when outputting the hours
```

Other combinations:

```
DH = output total remaining days + total remaining hours (less the time output as days)
DHM = output total remaining days + total remaining hours (less the time output as days) + total remaining
minutes (less the time output as hours)
DHMS = output total remaining days hours minutes and seconds, each taking into account prior time already
accounted for

Any output not asked for being output as a whole number (E.G. if there are 61 seconds to output, but the format
only calls for minutes, leaving 1 second left over) discard the remainder of non whole-number time units
```

Input

Here are some sample input lines from a dataset (note, your program will only ever receive 5 lines at a time, except for the example file showing all 12 combinations of output types)

```
2019-01-01 00:00:00 2019-02-01 01:01:01 D
2019-01-01 00:00:00 2019-02-01 01:01:01 H
2019-01-01 00:00:00 2019-02-01 01:01:01 M
2019-01-01 00:00:00 2019-02-01 01:01:01 S
2019-01-01 00:00:00 2019-02-01 01:01:01 DHMS
2019-01-01 00:00:00 2019-02-01 01:01:01 HMS
2019-01-01 00:00:00 2019-02-01 01:01:01 MS
2019-01-01 00:00:00 2019-02-01 01:01:01 DH
2019-01-01 00:00:00 2019-02-01 01:01:01 DM
2019-01-01 00:00:00 2019-02-01 01:01:01 DS
2019-01-01 00:00:00 2019-02-01 01:01:01 HM
2019-01-01 00:00:00 2019-02-01 01:01:01 HS
```

Output

Output the remaining time until the countdown finishes, taking care to output the largest units first, and subtract those units of time from the remaining units output, in sequence.

```
there are 31 days remaining until 2019-02-01 01:01:01
there are 745 hours remaining until 2019-02-01 01:01:01
there are 44701 minutes remaining until 2019-02-01 01:01:01
there are 2682061 seconds remaining until 2019-02-01 01:01:01
there are 31 days 1 hours 1 minutes 1 seconds remaining until 2019-02-01 01:01:01
there are 745 hours 1 minutes 1 seconds remaining until 2019-02-01 01:01:01
there are 44701 minutes 1 seconds remaining until 2019-02-01 01:01:01
there are 31 days 1 hours remaining until 2019-02-01 01:01:01
there are 31 days 61 minutes remaining until 2019-02-01 01:01:01
there are 31 days 3661 seconds remaining until 2019-02-01 01:01:01
there are 745 hours 1 minutes remaining until 2019-02-01 01:01:01
there are 745 hours 61 seconds remaining until 2019-02-01 01:01:01
```

Rhymes with Ablaut

problem 27

15 points

This problem explores various types of reduplication in the English language. In linguistics, reduplication is the repetition of a word, either with an altered consonant or an altered vowel.

For example, have you ever noticed pairs of English words in which only an inner vowel sound changes, such as chit-chat, hip-hop, or ping-pong? The technical term for this pattern is Ablaut Reduplication. In addition to word pairs, English has examples of three or more words that follow the pattern of ablaut reduplication, such as sing, sang, song, and sung.

What could be more fun? Just this: when ablaut reduplication is used in a phrase, native English speakers usually prefer saying and hearing a specific short vowel-sound sequence. That's why we say splish-splash and tick-tock, but almost never splash-splish or tock-tick. When the vowels are reversed it just doesn't sound right. This is called progressive vowel retraction. The progressive order of the short vowels is I, A, E, O, U. When the inner vowel changes in a non-progressive order, we'll just call it ablaut reduplication.

The English language includes other types of reduplication. For example, in copy reduplication a word is wholly repeated, as in bye-bye and yada-yada-yada. And in rhyming reduplication, a change in the initial consonant sound produces words that rhyme, such as nitty-gritty and hocus-pocus. In some cases one of the initial consonants may be missing, as in itsy-bitsy. Finally, there is a "shm" reduplication, which is a specific type of rhyming reduplication in which there are exactly two words and the initial consonant sound of the second word is replaced by the letters s-h-m, as in fancy-shmancy, Joe Shmoe, and apple-shmapple.

All the above rules are simplifications of patterns that have many complications and variations. Since this is a CodeWars problem, we'll not delve too deeply into phonetics and exceptions. In particular, we'll pretend English words are always pronounced the way they are spelled. Words will be one or two syllables. Vowel substitutions only occur with single-letter vowel sounds. Write a program to determine the type of reduplication present in a sequence of words.

Input

Each line of input begins with an integer word count (two, three, or four) and that same number of words, all in upper case. (For parsing simplicity hyphenated and compound words have been presented as individual words.) The end of input is signalled with the number zero.

```
2 CHIT CHAT
2 APPLE SHMAPPLE
2 BOT BIT
4 LONG PONG TONG SONG
2 SHMICK SHMUCK
2 CHICK FLICK
4 SING SANG SONG SUNG
2 BUG BAG
2 BYE BYE
2 HOCUS POCUS
2 DIG DUG
3 YADA YADA YADA
2 JOE SHMOE
2 CULTURE VULTURE
2 FORGET FORGOT
0
```

Output

For each word sequence, the program must print the sequence and the correct type of reduplication, either COPY, RHYMING, SHM, ABLAUT, or PROGRESSIVE. Every line will fit a pattern. Keep in mind that shm is a specific subtype of rhyming reduplication and progressive is a specific subtype of ablaut reduplication.

```
CHIT CHAT  PROGRESSIVE
APPLE SHMAPPLE  SHM
BOT BIT  ABLAUT
LONG PONG TONG SONG  RHYMING
SHMICK SHMUCK  PROGRESSIVE
CHICK FLICK  RHYMING
SING SANG SONG SUNG  PROGRESSIVE
BUG BAG  ABLAUT
BYE BYE  COPY
HOCUS POCUS  RHYMING
DIG DUG  PROGRESSIVE
YADA YADA YADA  COPY
JOE SHMOE  SHM
CULTURE VULTURE  RHYMING
FORGET FORGOT  PROGRESSIVE
```

Series Identifier

problem 28

20 points

What type of series are you looking at? Geometric? Exponential? TV? Who knows? That's the question your program needs to solve.

Geometric Series

A series is geometric if the items in the series set are increasing (or decreasing) at the same rate. For instance, a geometric series which multiplies every term by 10 would be: 1 10 100 1000, etc. NOTE: you may encounter items from a geometric series which contains gaps! In those cases you will need to "fill in the gaps" to test if it is a valid geometric series or not. If you find a geometric series with gaps in it, add "(With Gaps)" to your output.

Exponential

A series is exponential if the next item in the series set is a result of raising the prior item to a set power. For instance, X^2 exponential series raises each term to the power of 2, example: 2 4 16 32, etc. (Note, your program will only need to identify X^2 and X^3 exponential series)

Fibonacci

A series is a Fibonacci series if the next term in the series set is a result of adding the prior two terms together (with term 0 not really factoring in except as a seed to start the series). For example: 1 1 2 3 5, etc.

Input

You will receive a dataset with 1-5 lines consisting of numbers in a series. Numbers can be negative, and they can either increase or decrease, depending on their series. You will not receive any fraction or decimal numbers, and you are guaranteed to receive at least one data line.

```
4 40 400 4000
8 64 4096 16777216
-3 -27 -19683 -7625597484987
5 8 13 21
300000 3000 300 3
8 3 82 17 9
```

Note, you may encounter items from a series which are not contiguous (as in the second Geometric series above in the sample input). In those cases you will need to "fill in the gaps" to determine what series you are dealing with

Output

For each line, simply name the series the line is using for its number progression. (For exponential, either output X^2 or X^3 depending on the exponent you have determined the series is using). If you CANNOT determine the series the line represents, output the line + "is an Unknown series"

```
Geometric
X^2
X^3
Fibonacci
Geometric (With Gaps)
8 3 82 17 9 is an Unknown series
```

Linter

problem 29

20 points

A "linter" is a programming tool generally used to help find issues with code that might otherwise go missed. For example, some code might compile without errors, but have subtle bugs that could be caught via static analysis (like linting).

You are tasked with writing a simple linter that checks a new programming language, BALONEYPLUSPLUS, for the following code issues:

Lint Error Code	Message	Explanation
10	Variable declared but not used	A variable declaration statement exists but the variable is not referenced elsewhere
20	Unexpected indentation	A style check, improves readability of code
30	Trailing whitespace	A style check, lines should end with a line break, not a space or tab character
40	Func declaration without documentation	Encourages the programmer to add an explanatory comment to their functions

See the next page for the complete BALONEYPLUSPLUS Language Specification

Here is a valid code example

```
VAR i
VAR j
FOR $i = 1 TO 5 DO:
    PRINT SUM( $i , $j )
NEXT

# IF statement with correct indentation
IF SUM( 1, 1 ) != 2 THEN:
    PRINT "Something is seriously wrong"
ENDIF

# SUM returns the arithimetic total  of its two parameters
FUNC SUM( a, b )
    RETURN $a + $b
ENDFUNC

# Note required comment above "SUM" function declaration
```

Tip

Keep in mind that comments are ignored by the compiler, so from a language standpoint a variable appearing in a comment is not actually an instance of that variable being used.

Input

The input file is a single text document, the source code of a valid BALONEYPLUSPLUS program.

```
VAR flag
VAR result

$result = 0

FOR $i = 0 TO 10 DO:
   $result = $result + $i
NEXT

PRINT $result   (there are trailing spaces here)
```

Output

Your linter will scan the program for the listed lint error checks, and report the following information:

- Line number where the lint error occurred
- The lint error code
- The lint Message (from the table above, must match exactly including case)

For example, this is expected output for the above input example:

```
1:10 Variable declared but not used
7:20 Unexpected indentation
10:30 Trailing whitespace
```

Report lint errors in sorted order. Sort by line number (ascending). If one line has multiple lint errors, sort by lint error code (ascending).

Linter (continued) - BALONEYPLUSPLUS Language Spec

General Guidelines

- The language is case sensitive
- Variables must be declared with `VAR foo`, and then used like `$foo`.
- Variables must be surrounded by whitespace, "MYFUNC($i)" is invalid but "MYFUNC($i)" is correct.
- Nested loops and conditionals are not allowed.

Conditional Statements

```
IF <condition> THEN:
    <statements>
    <...>
ENDIF
```

Note each statement in the inner block is indented 4 spaces.

Loops

Valid `FOR` loops:

```
FOR $z = 1 TO 10 DO:
    <statements>
NEXT

VAR num
$num = 10
FOR $z = 1 TO $num DO:
    <statements>
NEXT
```

Note the inner block is indented 4 spaces.

Whitespace Before Statement

In this language, all lines should begin with a non-whitespace character, unless indentation is required as already specified. For example, this is correct:

```
# some comment
VAR xyz
```

This is not correct:

```
# some comment
 VAR xyz
```

Empty lines (lines without any non-space characters) are ignored.

Comments

Comments begin with the symbol # and indicate the remainder of the line is a comment. Comments are discarded by the compiler before building the application.

Examples:

```
# this is a comment
VAR xyz # this is also a comment
```

Function Declarations

Syntax:

```
FUNC <name>( <arg1>, <arg2>, ... )
    <statements>
ENDFUNC
```

Example:

```
FUNC myfunc( a, foo )
    VAR b
    $b = $a + $foo
    RETURN $b
ENDFUNC

FUNC limitOne( j )
    RETURNIF $j == 0 : 0
    RETURNIF $j > 0 : 1
ENDFUNC
```

To encourage documentation, the language requires that function declarations are preceeded with an explanatory comment. The comment must begin with the function name and it must be written on the line immediately above the function declaration. Example:

```
# myfunc adds a and foo and returns the result
FUNC myfunc( a, foo )
...
```

The Unibaker

problem 30

29 points

The newest fad for party cakes is the cupcake-cake, where many cupcakes are placed side-by-side to make a single cake. The celebrated Unibaker has developed his own version. All of his cupcakes are either White or Chocolate cake.

- and they are square,
- and they are placed together to create a larger square cake,
- with specific requirements:

The White and Chocolate cupcakes each form a shape called a Nurikabe, which means:

- No 2x2 set of cupcakes can all be the same type
- All cupcakes of the same type must be connected by neighboring sides. This creates 2 fully-connected regions: one White, one Chocolate
- No loops are formed. So, at least one of each type is on the border of the full cake.

Note: The number of White and Chocolate cupcakes does not have to be equal.

Input

The first line of input is a single integer N (max of 11), the size of the square cake. The next N lines contain N characters, each reprenting:

- '.' - a period or White cupcakes
- '#' - a hash for Chocolate cupcakes
- '?' - a question mark for Unknown cupcakes

You must determine all the cupcakes and print the diagram for the full cake. There is only one solution.

Example 1

```
4
???#
?.#?
?.??
..?.
```

Example 2

```
7
??.????
????.??
?##?.??
..?#?#?
?.?####
???.???
?#???#?
```

Output

Output the fully defined grid showing the 2 Nurikabe regions for White (.) and Chocoloate (#) cupcakes.

Output 1

```
####
#.#.
#.#.
....
```

Output 2

```
.......
.#.#.#.
.###.#.
...#.#.
#.#####
#.....#
#######
```