Important Contest Instructions

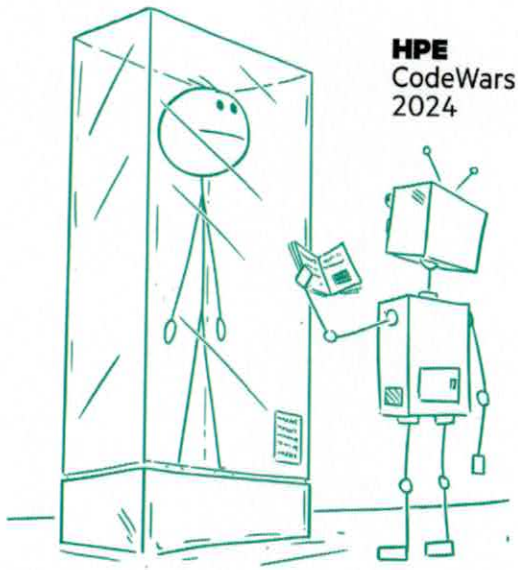
Good Luck!

Please read the following instructions carefully. They contain important information on how to write your programs, how to run them and how to prepare them for submission.

Language Versions

The judges will be using the following versions for each of our supported languages.

Language	Version
Java	21.0.2
Python	3.12
C	13.2
C++	13.2



Java

Your program source file name must be named **probXX.java** and your class name must be **probXX**, where **prob** is all lower case and **XX** corresponds to the two digit problem number.

- Example class name: `public class prob01`
- Example source file name: `prob01.java`

Java programs can rely on the standard Java library.

Python

We recommend naming your program **probXX.py** (where **XX** corresponds to the two digit problem number).

Python programs can rely on the standard python library.

C / C++

We recommend naming your program source file **probXX.c** or **probXX.cpp** (where **XX** corresponds to the two digit problem number).

C or C++ programs must be submitted as source files and can rely on glibc/libm and the standard headers.

> Download our "C/C++ Developers Guide" from https://hpecodewars.org/downloads

Note

If there is a discrepancy between a data set in the problem statement and the data set file (for either input or output), please consider the data set file to be correct! We will announce any errata as needed.

Program Input

Most programs will require input. You have two options, File Input or Keyboard Input. Some problems may require Directory Input.

File Input

Your program may read input from a file named **input.txt** for all problems. The file will be in the same directory as your program. Use `"input.txt"` as the filename and **do not prefix it with a path**.

> Download our "Guide to data file I/O" from https://hpecodewars.org/downloads

Keyboard Input

Your program may read input from standard in (the keyboard). Do **not** include any prompts in your output. There are two options to provide input to your program via standard in (the keyboard).

- (1) The preferred option is to redirect the contents of a file to standard in of your program.

```
java prob01 < input.txt
python prob01.py < input.txt
prob01.exe < input.txt
```

- (2) Otherwise you may type everything manually, or copy/paste from the data set files.

> Tip: Type `Ctrl-Z <return>` to signal the end of keyboard input.

Directory Input

For some problems, you will need to read from a directory (folder) named `"files"`. The directory will be in the same directory as your program. The student dataset includes a new directory called `"files"`. **MOVE IT** to the same directory as your program.

Use `"files"` as the its name and do not prefix it with a path. The judges will also place the `"files"` directory in their testing directory.

> Download our "Guide to Directory I/O" from https://hpecodewars.org/downloads

Program output

Your program must send the output to the screen (standard out, the default for any print statement). Do **not** include any prompts for input in the output! The only output for your program should be the expected output for the given problem.

Testing your program

Test your program like a judge will with the Python Check Problem script (`checkProb.py`) in the student ZIP file. The script will automatically run your program with each data set for that problem. See `README.txt` in the student ZIP file for more details.

> We strongly encourage you to test your program with all data sets in the `student_datasets` directory of the student ZIP file!

Online Code Submission

Programs will be submitted through the contest site using a text box (with a large character limit, don't worry!) instead of a file upload. You will need to copy and paste the program from your local source code file into the text box. Once you select the problem number and language from the respective drop-downs, you can submit the program for judging.

problem
00

Are You Prepared?

1
points

Lady Jessica has an important message for her son.
Make sure it is delivered.

Input

There is no data file to read in for this (**and only this**) problem.

Output

Print this ominous quote from Lady Jessica, exactly as shown:

```
You are not prepared for what is to come.
```

Discussion

Luckily, it's for her son, and not for you. Of course *you* are prepared. Right?

problem 01

Kung Fu Wisdom

1 points

Po has a new movie. Honor his friends by giving them wisdom as they arrive at the premiere!

Input

You will receive one name on a single line. It will not contain spaces or punctuation (even if the proper spelling of the name would include them).

```
JackBlack
```

Output

Print to the screen the following sentence. **The only part of the sentence which should change is the name!**

```
Your real strength comes from being the best JackBlack you can be!
```

Discussion

Be sure to run your solution against **all** of the student data sets. They have DIFFERENT names.

Additional Examples

Input	Output
AngelinaJolie	Your real strength comes from being the best AngelinaJolie you can be!
JackieChan	Your real strength comes from being the best JackieChan you can be!
SethRogen	Your real strength comes from being the best SethRogen you can be!

problem 02

Can You Solve This Equation?

2 points

Only 1 in 1,000,000 people can solve this simple equation

8 ÷ 2 x (2 + 2) = ?

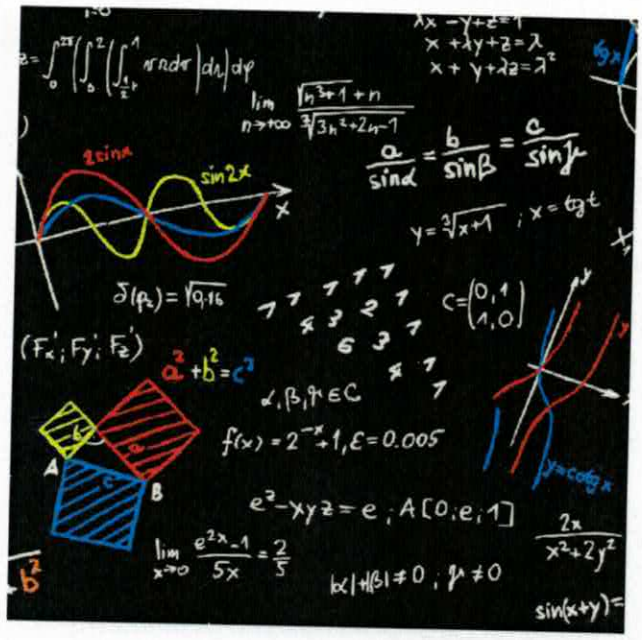
Well, maybe a few more can...

Math notation *should* be unambiguous, but sadly many students struggle with knowing how to perform calculations in the correct order. The reasons are varied, but this is a case where some computer programming may help.

If we want a computer to calculate a mathematical expression like the one above, we have to explicitly represent all of the operations with symbols on our keyboards. The result looks like this:

8 / 2 * (2 + 2)

Your programming language will calculate its answer by doing the math in parentheses first. Then it will do multiplication and division from left to right, to arrive at a final answer of 16.



Write a program to solve the mathematical expression, substituting input values for *x* and *y* in the equation.

```
x / y * (y + y)
```

Input

Input is a pair of integers, x and y on one line, separated by a space.

```
6 3
```

Output

The program must print the value of *x / y * (y + y)*.

```
12
```

Discussion

A little algebra could simplify your program. No decimal points will be needed. Check **all** of the student data sets.

Additional Examples

Input 1	Output 1	Input 2	Output 2
8 2	16	1234 5678	2468

problem 03

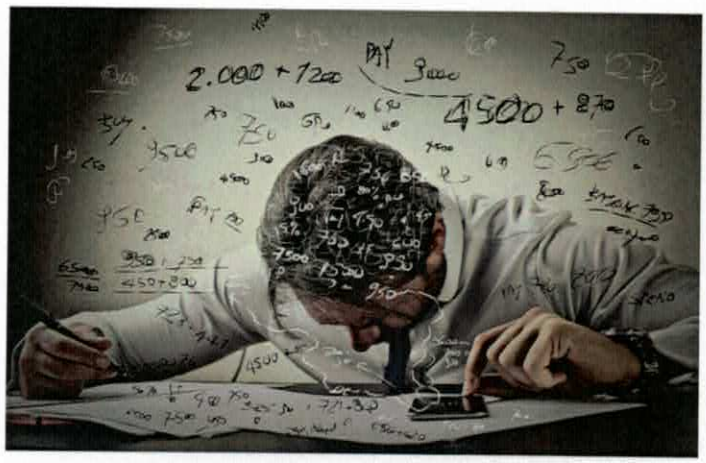
Is It Plural?

2 points

Good luck! You're the greatest!
- You, 1 minutes ago

When humans use your program, many of them will perceive the smallest of flaws as signs that your app is unreliable, or worse, uncool.

Don't kill your unicorn app with sloppy recognition of singular and plural.



Input

Each line of input is an integer from 0 to 99 inclusive. The input ends with a -1. Only the number 1 should be singular in your output sentence. Zero should also be pluralized. You may encounter numbers more than once in the same dataset (they are not guaranteed to be unique).

```
6
1
4
2
-1
```

Output

For each integer, the program must print the phrase, "You, N minutes(s) ago." Except of course, you must replace N with the input number, and it must print the singular "minute" for the number one (1), and "minutes" for zero and values greater than one.

```
You, 6 minutes ago.
You, 1 minute ago.
You, 4 minutes ago.
You, 2 minutes ago.
```

Additional Examples

Input 1	Output 1
87 63 99 1 22 3 21 -1	You, 87 minutes ago. You, 63 minutes ago. You, 99 minutes ago. You, 1 minute ago. You, 22 minutes ago. You, 3 minutes ago. You, 21 minutes ago.

Input 2	Output 2
1 9 7 7 1 9 8 0 1 9 8 3 -1	You, 1 minute ago. You, 9 minutes ago. You, 7 minutes ago. You, 7 minutes ago. You, 1 minute ago. You, 9 minutes ago. You, 8 minutes ago. You, 0 minutes ago. You, 1 minute ago. You, 9 minutes ago. You, 8 minutes ago. You, 3 minutes ago.

problem 04

Code-A-Sketch

2 points

El Barto is working on some epic Etch A Sketch art.

Only problem is, Homer knocked the thing to the ground, and all of Bart's hard work got messed up!

Help Bart, help you, have fun looking at sand art!

Input

You will receive exactly 2 lines of input. The first line will be a single character, that is the line break character. The second line will be Bart's art, as a single line (up to 10,000 unicode characters). We are showing it on several lines, because we don't have as much space on paper as we do in a data file.

```
~
''|\/\/\/\/
\/|''~''|''''''''''|''~''|''''''''''|''~''|''''''''''|''~''|''''__''__|''~''|'''/''\/''
\''~''|''(o''')o'')'~'/
c'''\__/'--.''~'\_''',''''''-'''~''|'''\_______)'~''|''''''_)''''''~''|''''''|''''''~'/
`-----'\''''''~/''''''''''\''''''
```

Output

Split up the line on the character given, and output Bart's original picture (without the split character in the output)

```
''|\/\/\/\/\/|''
''|''''''''''|''
''|''''''''''|''
''|''''''''''|''
''|''''__''__|''
''|'''/''\/''\''
''|''(o''')o'')'
'/c'''\__/'--.''
'\_''',''''''-'''
''|'''\_______)'
''|''''''_)''''''
''|''''''|''''''''
'/`-----'\''''''''
/''''''''''\''''''
```

Discussion

We do not have enough page space to print additional examples. Therefore you should run your solution against **all** of the student data sets to verify that it works!

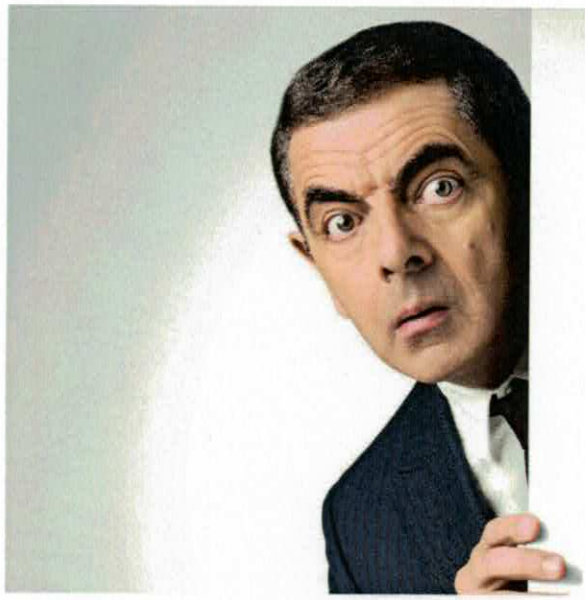
problem 05

Char Decoder

2 points

Humans have been trying to come up with sneaky ways to send messages to each other since we figured out how to paint on cave walls.

Computers haven't changed that. Help Johnny English crash the code.



Input

Johnny will be handed coded messages by field agents. The coded message will be on a single line of up to 10,000 characters in length. Note, due to the lack of horizontal space in these pages, the inputs are showing as wrapping to multiple lines. In your actual input it will all be on a single line.

```
73 116 39 115 32 97 110 32 117 110 109 105 116 105 103 97 116 101 100 32 100 105 115 97
115 116 101 114 44 32 69 110 103 108 105 115 104 33 10 73 32 99 111 117 108 100 110 39
116 32 97 103 114 101 101 32 109 111 114 101 44 32 115 105 114 46
```

Output

Each numeric character in the message will be separated by a space. Convert each numeric code into its unicode symbol equivalent (e.g. 65 == A). Messages can include letters, numbers, punctuation, spaces, and newline characters.

```
It's an unmitigated disaster, English!
I couldn't agree more, sir.
```

Discussion

Depending on your coding language of choice, you may have to pay special attention to the newline character (ASCII/Unicode 10). Make sure your solution passes **all** of the student datasets before submitting it!

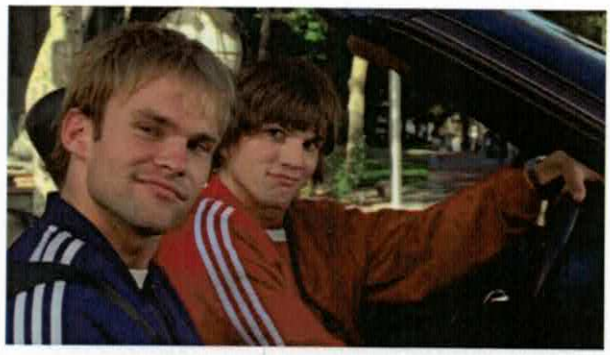
Additional Examples

Input 1	Output 1
73 110 116 111 32 116 104 101 32 103 97 114 98 97 103 101 32 99 104 117 116 101 44 32 102 108 121 98 111 121 46 10 78 111 32 114 101 119 97 114 100 32 105 115 32 119 111 114 116 104 32 116 104 105 115 46 10 45 45 72 97 110 32 83 111 108 111	Into the garbage chute, flyboy. No reward is worth this. --Han Solo

problem 06

The Beach That Makes You Old

3 points



So, there's this beach, and it makes you old.

And there's a group of high school students who want to visit the beach that makes you old. They want to rent a car to go on a trip, but most car rental places will not rent to you unless you are at least 25.

So, they want to go to the beach that makes you old.

The beach that makes you old doesn't make sense logically. Maybe it's the sand, or the rocks, or the coral. We don't know.

For every hour a person remains at the beach, their body ages one month. Write a program to calculate how long students should remain at the beach, so that they can rent a car.

Plot Twist: After the students return from the beach, their driver's licenses still show their birth dates. Checkmate, students.

Input

The first line is the number of students. Each following line gives a student's name and their age in years and months. All ages are less than 25 years.

```
5
Asok 16 3
Brenda 17 11
Cariana 15 9
Duc 17 0
Ezhno 15 6
```

Output

In the same order as the input, print each student's name, one space, and the number of hours they must spend at the beach to reach age 25.

```
Asok 105
Brenda 85
Cariana 111
Duc 96
Ezhno 114
```

Additional Examples

Input 1	Output 1	Input 2	Output 2
4 Gwen 16 1 Tom 17 2 Tony 18 0 Adrian 15 3	Gwen 107 Tom 94 Tony 84 Adrian 117	8 Carrie 17 9 Mark 16 11 Daisy 15 0 Adam 14 5 John 14 1 Oscar 15 3 Anthony 16 7 Naomi 17 2	Carrie 87 Mark 97 Daisy 120 Adam 127 John 131 Oscar 117 Anthony 101 Naomi 94

problem 07

Four of a Kind

3 points

Our brains are good at spotting repetition and similarities, such as the same cards in a row.

But sometimes it's harder to determine how many identical objects are in a line. Help your brain.

Write a program to alert you to all groups of 4 identical characters.

Input

The input is one line of letters (upper or lower case) and/or numbers (not more than 80 characters). The last character in the line is a period (the only character that isn't a letter or number.)

```
ABCCCCDEEEFZZZZZhellooooeFFFrepeatedddddGGGGGGeeeeWWWWzzaaaarrrrosssssOOOOO.
```

Output

Reading from left to right, find any sets of exactly 4 identical characters adjacent to each other. Print that character once. Do not print if there are more or less than 4 identical characters together. Continue to print characters on the same line until you reach the end of the input line. If there aren't any sets of 4 of a kind, print "No Four of a Kind"

```
CodeWars
```

Discussion

There are several more student datasets to test.

Additional Examples

Input	Output
12345123451234512345123451234512345123451234512345123451.	No Four of a Kind
3222111ae1111222233334444455555AAAAbbbXXxXLLLLLLLLzzzzZ.	123Az
111111111111HHHH22222222eeeell1133333333sllllooooo44444444s4444.	Hello4

problem 08

License Plate OCD

3 points

While sitting in traffic on your way to school, it suddenly strikes you that most of the license plates you see on the road list the letters and numbers on the plate in *whatever random order they feel like*. You find that wanton disregard for proper order *unacceptable*.

Help bring sanity back to the galaxy, at least in your mind, by re-ordering the characters on the license plates so that the characters read left to right in *ascending order!* (As is good and proper)

Input

You will receive 7 to 10 (inclusive) characters on a single line. The characters will be a combination of uppercase letters, and integers. It is possible you may receive a line with only letters or only numbers. It is not possible that you will receive a blank line.

```
G597LUC
```

Output

Arrange the letters and numbers, in place, in ascending order, reading left to right. DO NOT move any of the character positions to do this. Number positions need to stay number positions, and letter positions need to stay letter positions.

```
C579GLU
```

Discussion

Reminder: have you run your solution against **all** of the student data sets?

Additional Examples

Input	Output
STR8177WRS	RRS1778STW
Z543CBA	A345BCZ
Z7H1J9K	H1J7K9Z
895272	225789
JYABNDUB	ABBDJNUY

problem 09

Radio Issues

3 points

Dinky Do and his all kazoo crew are having radio issues.

All of the words in their broadcasts are hitting the airwaves elongated.

Help the audio crew clean up the transmission before people start calling the station!

Input

You will receive several lines of text, up to 125 characters per line. The text can include: letters, numbers, spaces, and punctuation. Input ends when the word STOP is encountered on its own line by itself.

```
WWee'rree hheerree ttoo kkiicckk ooffff yyoouurr eevveenniinngg hhoouurr wwiitthh
ssoommee ccoommeeddyy ffrroomm
JJaacckk BBeennnnyy, ssoommee mmuussiicc ffrroomm tthhee AAnnddeerrssoonn
SSiisstteerrss, aanndd tthhee nneewwss
ffrroomm yyoouurr ffaavvoorriittee aanndd mmiinnee -- EEddwwaarrdd RR. MMuurrrrooww aatt
7.
STOP
```

Output

Help the audio techs clean up the elongation of the signal by removing the duplications of letters from the signal before it hits the transmission tower. Ignore numbers, white space, and punctuation.

```
We're here to kick off your evening hour with some comedy from
Jack Benny, some music from the Anderson Sisters, and the news
from your favorite and mine -- Edward R. Murrow at 7.
```

Discussion

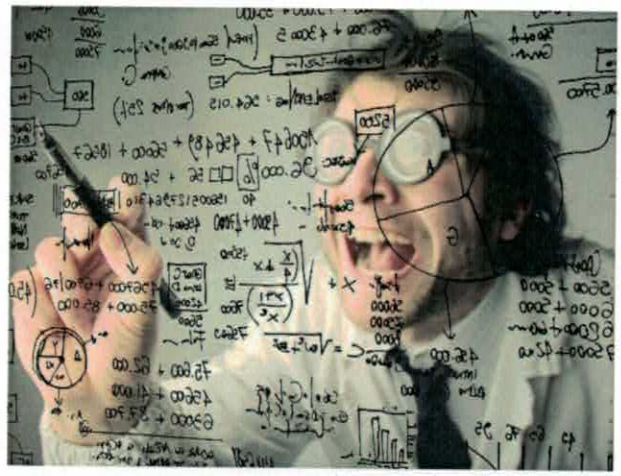
We do not have enough page space to print additional examples. Therefore you should run your solution against **all** of the student data sets to verify that it works!

problem 10

Sum Checker

3 points

As the TA for your math teacher, your job is to double check her work. She will hand you papers she has graded. Your need to then check her math by reading in one line of input at a time, verifying the math then printing out a results sheet for her.



Input

You will receive 3-10 addition equations, each on one line. Each line has an integer, a '+', another integer, an '=', and a third integer. The second integer is always positive. The first and/or third integer MAY be negative. There are no spaces. END on a line by itself signals the end of input.

```
77+23=100
1+1=11
1000000000+1=2000000000
-17+17=0
END
```

Output

Check the math of each equation. If the provided sum is correct, write the word CORRECT and move on to the next input. If it is wrong, write the correct version of the equation.

```
CORRECT
WRONG: 1+1=2
WRONG: 1000000000+1=1000000001
CORRECT
```

Discussion

Reminder: have you run your solution against **all** of the student data sets?

Additional Examples

Input 1	Output 1	Input 2	Output 2
8+7=19 88+1=881 1+1=2 2+2=22 5+0=5 END	WRONG: 8+7=15 WRONG: 88+1=89 CORRECT WRONG: 2+2=4 CORRECT	-101+105=6 -500+1001=52 99+1=101 99+2=101 77+77=7777 1001+1001=2002 52+8=16 -16+8=-16 -20+21=1 END	WRONG: -101+105=4 WRONG: -500+1001=501 WRONG: 99+1=100 CORRECT WRONG: 77+77=154 CORRECT WRONG: 52+8=60 WRONG: -16+8=-8 CORRECT

problem 11

Milliseconds

4 points

Time is an abstract concept. Until it isn't.

> And then one day you find
> Ten years have got behind you
> No one told you when to run
> You missed the starting gun

Tracking time on computer systems can be a big pain, because human attention covers large time scales, and yet computers can also measure time on very tiny scales. Most CPUs don't have native integers large enough to cover the entire scale, so we fix that problem by defining multiple scales.

Input

Input is a series of timestamps, where each stamp is a pair of unsigned integers. The first is the 64 bit number of seconds since Jan 1, 1970 and the second is the 32 bit number of nanoseconds since the start of the left time stamp. The timestamps are sorted in forward-time order. The input ends with a pair of zeros.

```
1362469998 703434300
1362470822 919311900
1362470973 534495500
0 0
```

Output

For each timestamp after the first, the program should print the number of full milliseconds between that timestamp and the one before it. Truncate any fraction of milliseconds.

```
824215
150615
```

Discussion

There are 1,000 milliseconds in a second. There are 1,000,000 nanoseconds in a millisecond.

For the first time comparison above:

```
1362470822 - 1362469998 = 824 s = 824000 ms. 919311900 - 703434300 = 215877600 ns =
215.8776 ms.
Add ms, and report only FULL milliseconds (truncate any fractions): 824215
```

However, if the number of nanoseconds is smaller on the later time in a comparison:

```
534495500 is less than 919311900.  Reduce 1362470973 by 1 to 1362470972 seconds and add
10^9 nanoseconds:
1362470972 - 1362470822 = 150 s = 150000 ms.  1534495500 - 919311900 = 615183600 ns =
615.1836 ms.
Add ms, and report only FULL milliseconds (truncate any fractions): 150615
```

problem 12

The Lost Socks

4 points

Have you ever been frustrated because the washing machine seems to have lost another sock? It has happened to all of us and yet no one knows where all the lost socks end up. Hmm... *The Lost Socks* would make for a great band name.

You will be given a list of socks that are ready to be folded and put back into the dresser. You need to identify all incomplete pairs of socks. An incomplete pair means there is only one sock of that type in the list. If there are no incomplete pairs, then you can say "No lost socks".

Socks are made out of the following materials: Cotton, Wool, Polyester, Nylon. They have three different sizes: No-show, Crew, and Knee-high. Lastly, the socks have various colors: Cream, Navy, Maroon, Gold, Grey, and Teal.

The list of socks will not contain multiple pairs of the same sock.

Input

Each line of the input will contain one sock described by its: material, size, and color. The last line of input will be "END END END".

```
Cotton No-show Maroon
Wool Knee-high Teal
Cotton No-show Maroon
Wool Crew Teal
Wool Knee-high Teal
END END END
```

Output

For each sock which does not have a matching pair, print the sock's material, size, and color in the same order as the input. If all socks have a matching pair, then print "No lost socks".

```
Wool Crew Teal
```

Additional Examples

Input 1	Output 1	Input 2	Output 2
Wool Knee-high Cream Cotton No-show Gold Cotton No-show Gold Wool Knee-high Cream END END END	No lost socks	Nylon Crew Cream Cotton Knee-high Navy Nylon Crew Teal Cotton Knee-high Maroon END END END	Nylon Crew Cream Cotton Knee-high Navy Nylon Crew Teal Cotton Knee-high Maroon

problem 13

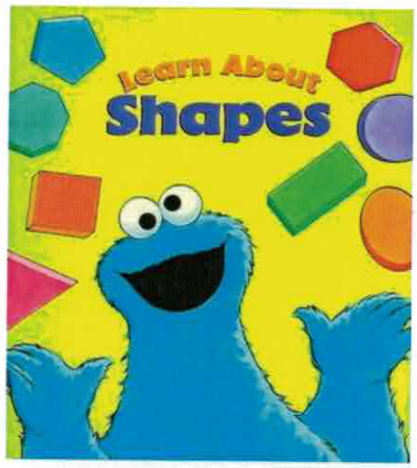
Complete The Shape

4 points

Your storybook editor asked you to mock up some shape sketches for your soon to be published children's book.

As your book is a preschool shapes book, all you need to do is make sure the rectangles section is complete, since it hasn't been reviewed yet.

Fix any incomplete rectangles.



Input

The input is a (possibly broken) rectangle of '#' symbols, each dimension at least 3 and at most 10.

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

Output

Complete the given rectangle. If the rectangle is already complete, instead write "Nothing to do"

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

Discussion

All illustrations in your book will have at least two complete Length and Width line segments from which you can determine the required lengths of the other sides.

Reminder: have you run your solution against **all** of the student data sets?

Additional Examples

Input 1	Output 1	Input 2	Output 2	Input 3	Output 3
### # # # # # # # #	### # # # # # # ###	##### # # # # #####	Nothing to do	#### # # # #	#### # # # # # # ####

problem 14

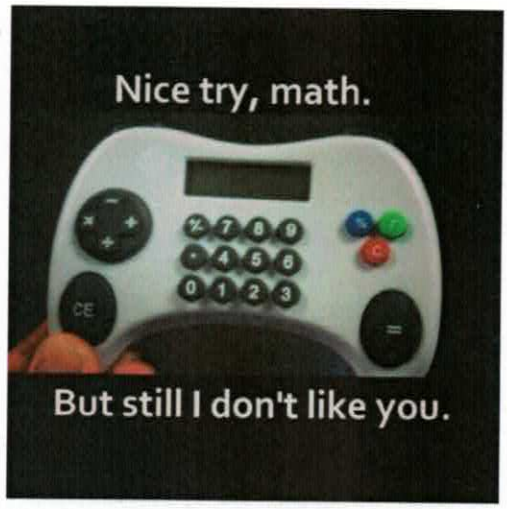
String Adder

4 points

Bad data has infiltrated your perfect data model.

Each character in your data should be a number (either integer or floating point), but someone wasn't paying attention in the data entry team, and allowed bad user input to corrupt the data model.

Quick, we must clean it out!



Input

Ask the database for one string at a time, and then examine the characters in the string given to you. There will be two lines: The input line, and a checksum line. The characters on the first line will be separated by spaces. You may encounter a hyphen (dash), but there **are no negative numbers** in the datasets (because the database won't allow it).

```
1 C 2 3 4 6.1 5 6 9.2222222 W 7 8 9 10 @ 99 100 # | 654 = 712 4.44
CheckSum=1639.7622222
```

Output

Clean up the data so that it follows the model explained above, and then double check that the sum of the remaining characters matches the given checksum. If it does, write CHECKED. If it does not, write BADCHECK:YourCalculatedCheckSum (with no spaces).

```
1 2 3 4 6.1 5 6 9.2222222 7 8 9 10 99 100 654 712 4.44
CHECKED
```

Discussion

The number of decimal points you need to output needs to match the longest number of decimal points found in your input. For example, given the input above, there are several decimal numbers: 6.1, 9.2222222, and 4.44. Because 9.2222222 has the largest number of decimal points, that number determines the number of decimal points to return for your checksum verification. Reminder: have you run your solution against **all** of the student data sets?

Additional Examples

Input	Output
9.9 A 0 2 $ 3.11 Y 6 * 9 2.0 _ Q - 0 2.8 8 @ 7 2.36796 ! CheckSum=53.92	9.9 0 2 3.11 6 9 2.0 0 2.8 8 7 2.36796 BADCHECK:52.17796
7.0 RR 2.99 3 t 6 7.1 _ 2 5.5555 % 3 2.548] 56 9999.99 @ CheckSum=10095.1835	7.0 2.99 3 6 7.1 2 5.5555 3 2.548 56 9999.99 CHECKED

problem 15

Whimsy's Formulas

4 points

Professor Whimsy needs your help in printing the formulas for various chemical compounds to the screen.

He has a number of chemical compound formulas written in in his notes followed by some whitespace (which could be one or more spaces or tabs) then a brief explanation of the compound.

He needs those written as just the formulas, with proper subscripts, without the explanations. He needs to emphasize to his students that when chemical formulas are hand-written, they should be written with the subscripts on the line below the rest of the formula.

Input

For example, the file for water reads:

```
H2O     (water; some say it is the universal solvent)
```

Output

The data from the professor's inputs should be modified to remove the explanation then pull all of the numbers down to the next line, leaving the alphabetic characters in place where they are. The numbers should also retain their position relative to where they were in the input.

```
H O
 2
```

Discussion

Insert a space in the original line wherever a number was pulled down to the next line. On the next line insert spaces when there are alphabetic characters above the position.

Reminder: have you run your solution against **all** of the student data sets?

Additional Examples

```
Input:  CH3CH2OH (ethyl alcohol aka ethanol aka something you shouldn't touch)
Output: CH CH OH
          3  2

Input:  CaCO3 (calcium carbonate, one of several active ingredients used in antacids)
Output: CaCO
           3
```

problem 16

Trend Line

4 points

As a Junior Trader at Day Traders R Us (yes, the "R" is spelled backwards) you are tasked with watching the stocks and keeping track of which ones are going up (everyone gets really excited when the line goes up).

To keep stock picks secret from Junior Traders who might use their insider knowledge to do a little trading on the side, the Senior Traders have replaced the stock ticker symbols with numbers.

Every trade made on that stock (making the line go up) gets logged to your systems. Your job is to read in all of those stock ticker numbers and find the trends.

Input

You will receive positive integer inputs (less than 10,000), one at a time on its own line (up to 100 lines). 0 on a line by itself signals the end of input.

```
1
2
2
3
21
5
7
101
882
21
23
7
54
4
2
3
77
1
7
2
8
8
99
100
0
```

Output

Keep track of how many times you have seen each non-zero number. Whichever number has been seen the most often will be your company's Trending Stock of the day. Second Place will be your company's Buy Now pick of the week. Ignore everything else. There will not be any ties for Trending or Second Place. (Some suspect the CEO is playing favorites, but that isn't for us lowly Junior Traders to decide).

```
Trending: 2 [4 count]
Second Place: 7 [3 count]
```

Discussion

We do not have enough page space to print additional examples. Therefore you should run your solution against **all** of the student data sets to verify that it works!

problem 17

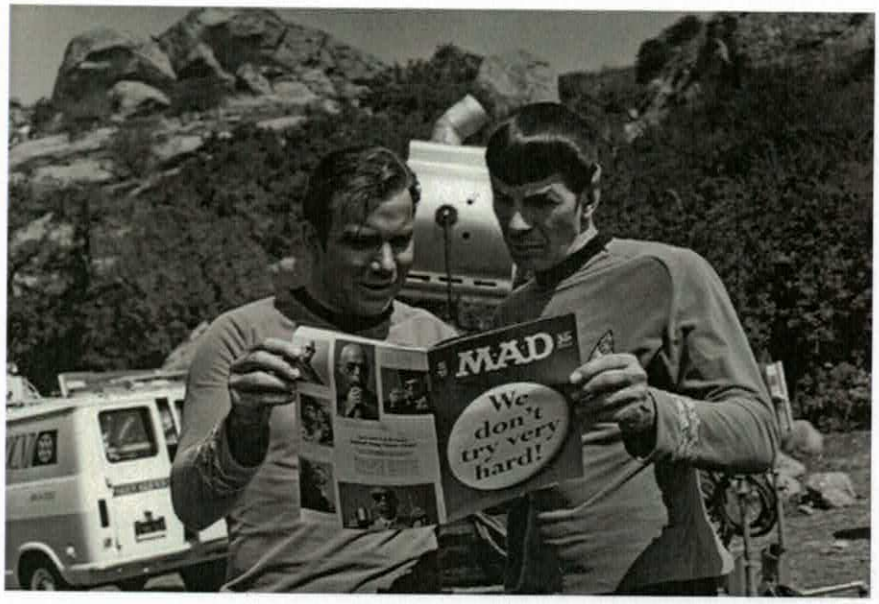
Mad Libs

5 points

Mad Libs are the funniest. Right? OK, well, either you are a mad-libber or you aren't. Some humans enjoy them.

"Mad Libs" is a party game where players are asked to provide words based on categories like *noun*, *adjective*, *food*, *color*, etc. The words are fit into blanks in a paragraph, in a way where the sentences are grammatically correct, but the meaning of the sentences may be completely nonsensical.

The point is to have fun! But some humans just aren't mad-libbers. It's OK. We still like you.



Input

The first line of input tells you the number of categories. Each category has its own line with at least one word(s) in the category, each word will be separated by a comma. The mad lib follows after the categories. The input ends with the line "THE END."

```
6
noun=whisper,car,laundry
place=island,upstairs,racetrack
adjective=understated,fast,sublime
color=mauve,blue,maroon
food=pizza,stir-fry,cake
ing-verb=singing,yodeling,dreaming
Yesterday, my friends and I went to the <PLACE>, and there
was a <COLOR> <NOUN> that we all admired. But then, some
guy started <ING-VERB> in a very <ADJECTIVE> way. So my
friends and I left and we ate <ADJECTIVE> <FOOD> at a
restaurant near the <PLACE>.
THE END.
```

Output

The program should print the mad lib paragraph line-for-line with the input, but with the category words filled in. The program must fill in the category words in the order given in the input file.

```
Yesterday, my friends and I went to the island, and there
was a mauve whisper that we all admired. But then, some
guy started singing in a very understated way. So my
friends and I left and we ate fast pizza at a
restaurant near the upstairs.
```

Discussion

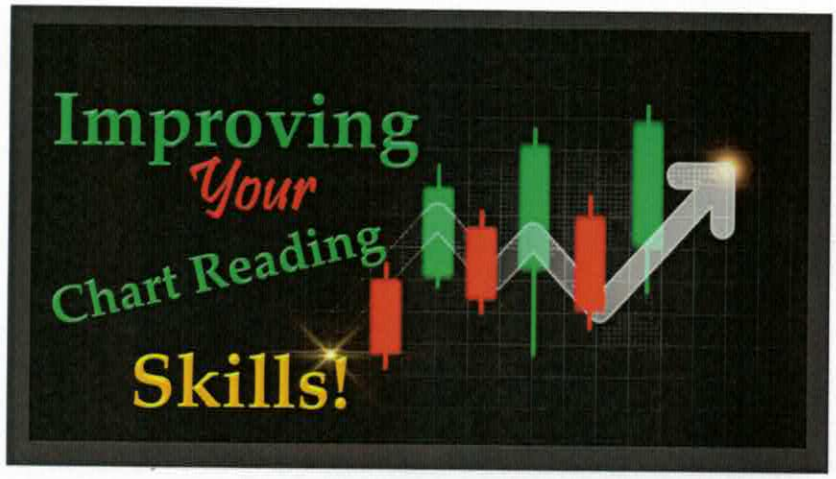
You will only encounter one of a category of words to replace on the same line. E.G. there will not be 2 verbs to replace on one line. Line lengths can be up to 250 characters long, and as short as 5 characters. For a given category not all words are guaranteed to be used in the mad-lib.

problem 18

Read The Charts

5 points

You've come across a set of printed charts and need to write a basic parser to convert the charts into a table of data. There will up to 26 series on the X axis, labeled alphabetically from A up to Z. The maximum number for the Y axis is 20.



Input

The first line of input contains two numbers N and M, where N is the number of series on the X axis, and M is the maximum Y axis value for this chart. The next N lines contain the data. This is followed by the bottom of the graph. The last line of input contains the X axis labels. The Y axis labels and each series is 3 columns wide, with the series label and data points in the third column. All data points are represented by an X.

```
5 5
 5 |
 4 |       X
 3 |     X X X
 2 |     X X X
 1 |   X X X X X
---+-------------
   |   A B C D E
```

Output

Read the chart and determine the value for each series on the X axis. Then print each X axis label followed by a colon, and the value on separate lines.

```
A: 1
B: 3
C: 4
D: 3
E: 1
```

Additional Examples

Input 1	Output 1	Input 2	Output 2

`1 1`
` 1 |`
`---+---`
`  | A`
	`A:0`	`3 11`
		`11 |`
		`10 |`
		` 9 |  X`
		` 8 |  X`
		` 7 |  X`
		` 6 |  X`
		` 5 |  X`
		` 4 |  X X`
		` 3 |  X X`
		` 2 | X X X`
		` 1 | X X X`
		`---+---------`
		`  | A B C`
			`A: 2`
			`B: 9`
			`C: 4`

problem 19

Mars Minutes

5 points

Tiny Spock has taken Sheldon to Mars to learn a lesson, and stranded him there. Help Sheldon figure out the changes to the time, since Mars has a slightly different *day* than Earth.

Input

You will receive one floating point number on a single line by itself representing a count of minutes on Mars.

```
555.0
```

Output

Help Sheldon calculate the number of days, hours, minutes, and seconds (to the nearest 10th of a second, truncating beyond that) represented by the Mars minutes.

```
0 days 9 hours 0 minutes 22.5 seconds
```

Discussion

Sheldon has observed that a day on Mars is 24 hours and 39 minutes long (exactly). In order to keep the math simple, Sheldon split up those extra 39 minutes into each hour, making an hour on Mars 61.625 minutes long. There are still 60 seconds in a minute though, and the hours still divide into 24 hour periods making up a day. You will need to do the rest of the calculations to determine how many days, hours, minutes and seconds are represented with each input of minutes.

Reminder: Check *all* of the student data files for examples of various scenarios.

Additional Examples

Input	Output
3600.0	2 days 10 hours 25 minutes 45.0 seconds
1541.675	1 days 1 hours 1 minutes 3.0 seconds

problem 20

Raffle Man

6 points

Mr. Raffleman needs some help delivering the raffle prizes for the CodeWars contest this year.

He's got his hands full with helping develop problems for the contest and doesn't have time to program the address label maker for shipping out the raffle prizes.

The names and addresses are already contained in a comma delimited file, but he needs you to pretty print them so they fit nicely on the mailing labels he purchased from the local office supply store.

Input

You will receive between 2 and 100 lines of input. Input ends when END is encountered on a line by itself. Each line is of the form: "Name", "Street Address", "City", "State", "PostalCode" (with exactly 4 commas and one space after each comma.)

```
Mike Wazowski, #193 Scream St. Apt. 5, Monstropolis, MU, 96334-2234
Strider, 0 Weathertop, Tolkien's Book, Middle Earth, 54321
S-Man, 1234 A Red and Blue Street, Metropolis, New Krypton, 101010
END
```

Output

Read in each line and separate the fields to their proper locations. "Name" on the first line. "Street Address" on the second. "City", "State" "Postal Code" on the third. A comma and space must separate the city and state, but only one space separates the state and postal code. No trailing spaces are allowed. Print a blank line between addresses.

```
Mike Wazowski
#193 Scream St. Apt. 5
Monstropolis, MU 96334-2234

Strider
0 Weathertop
Tolkien's Book, Middle Earth 54321

S-Man
1234 A Red and Blue Street
Metropolis, New Krypton 101010
```

Discussion

Note 1: address format is not necessarily strictly US-format (e.g. postal codes may be longer than 5 digits.) Do not enforce rules on lengths or other formatting rules which have not been specified in this problem's input/output directions.

Note 2: there will be no commas in the actual address data itself, you can safely split on commas.

Reminder: have you run your solution against **all** of the student data sets?

problem 21

Sort Checker

7 points

Some English words, like access, below, ghost, and knotty, are spelled with letters in alphabetic order. Other words may have the vowels in alphabetic order but not the consonants, or vice-versa.

Write a program to determine the sort grade of English words. The grades and their meanings in priority order are:

- SORTED: all the letters appear in sorted order (just because a word is CON and VOW sorted **does not** mean that a word is SORTED!
  ALL letters in the word *must* appear in alphabetical order!)
- CON-SORTED: all the consonants appear in sorted order
- VOW-SORTED: all the vowels appear in sorted order
- UNSORTED: not sorted

For this problem, the list of vowel letters is a, e, i, o and u.

Input

Each line of input contains a single english word with no spaces or punctuation. The last line of the file contains a single letter z. The shortest word you will have to check is guaranteed to be at least 2 characters in length.

```
ghost
sorted
berry
apple
zebra
smackers
merry
fix
hippo
z
```

Output

The program must print each input word, followed by the word's *highest* priority sort grade. The priorities are listed above.

```
ghost SORTED
sorted UNSORTED
berry SORTED
apple VOW-SORTED
zebra UNSORTED
smackers VOW-SORTED
merry CON-SORTED
fix SORTED
hippo CON-SORTED
```

Additional Examples

Input 1	Output 1	Input 2	Output 2
hills	hills SORTED	who	who VOW-SORTED
green	green VOW-SORTED	lives	lives UNSORTED
jar	jar CON-SORTED	in	in SORTED
keyboard	keyboard UNSORTED	apineapple	apineapple UNSORTED
mouse	mouse CON-SORTED	under	under UNSORTED
knife	knife UNSORTED	the	the VOW-SORTED
chair	chair CON-SORTED	sea	sea CON-SORTED
board	board UNSORTED	z	
z			

problem 22

y=mx+b

7 points

Monty Burns is fed up with his army of sycophants who tell him whatever it is they think he wants to hear. He just paid for an independent audit of his finances, and things aren't looking good. You seem like you know your way around the ol' teletype-o-graph, and Monty wants you to help!

Help Monty chart his finances so he can figure out if he can afford that second gold-plated back scratcher or not.

Input

Given integers m, b, and the size of your square output grid all separated by spaces on a single input line, graph the output. Note, m can be negative! Plan accordingly.

```
1 0 10
```

Output

Write the equation for the line in slope-intercept (y=mx+b) format, then graph it (only in quadrant 1, and only for the square size listed in your input). Your graph should show the intersection with the y-axis by having the second number on the y-axis replaced with a point (#). Other than (0,0), do not show intersections with the X-axis (stop before intersecting the X-axis). Spacing for your X and Y axis labels should always be 2 characters wide, left padded with zeros if needed. All points on the graph should be on the same horizontal line as their Y coordinate. For their X coordinate, place the point (#) above the second number in the x-axis labels (e.g. above the "9" in "09", above the "3" in "13", etc.). Do not extend the graph line past the boundary of the graph size, even if one of the coordinates would have been within size -- both must be within size to show the point.

```
y = 1x+0
10                  #
09                #
08              #
07            #
06          #
05        #
04      #
03    #
02  #
01 #
0#0102030405060708091O
```

Special Case:

In the case where slope is zero, that is the equation for a horizontal line. In the interest of simplicity, you can simply write out the equation as you always have substituting zero for "m", however, your line must be a horizontal line on that Y-intercept line -- following the rest of the rules as outlined above.

Discussion

There is an example of a negative slope in your datasets.

We do not have enough page space to print additional examples. Therefore you should run your solution against **all** of the student data sets to verify that it works!

problem 23

Shape Finder

8 points

Frank Lloyd Wright had *way* too much espresso, and his designs are *out of* ***control***.

Help his assistant try to catalog the ... *interesting* shapes Mr. Wright is using to compose his latest building designs so we can try to get out in front of this pending disaster before construction starts!

Input

You will receive up to 15 lines of input less than 40 characters wide, most of which will be spaces. Points for the shapes will be drawn with the # symbol. Input ends when you encounter the word END on a line by itself.

```
  #     #



  #     #
END
```

Output

Determine if the points from the input are a: triangle, square, rectangle, parallelogram, hexagon, pentagon, or unknown shape. Rules for shape definitions are listed in the discussion section. Once you have determined the shape, state it, along with the width and height. The origin is at the lower left, above the "E" in "END". The above points are: (2,5); (7,5); (2,0); (7,0);

```
Square with width 5 height 5
```

Discussion

- **Triangles** have exactly 3 points and not all in the same vertical or horizontal line.
- **Squares**, **Rectangles** and **Parallelograms** have 4 points (2 points each on 2 lines.) The horizontal distance is the same on both lines.
  - For **Squares** the lower points align vertically, and the vertical distance is the same as horizontal.
  - For **Rectangles** the lower points align vertically, but the vertical distance is different than horizontal.
  - For **Parallelograms** the lower points do not align vertically with the top.
- **Pentagons** have exactly 5 points. They MUST fit this description:
  - A single point on its own line; 2 points on a lower line; 2 more points on a lower line.
  - The lowest pair of points must both be between the first pair (horizontally), and the single (top) point must be between the lower pair of points.
- **Hexagons** have exactly 6 points. They MUST fit this description:
  - A single point on its own line; 2 points on a lower line;
  - 2 more points on a lower line vertically aligned with the first pair of points;
  - and 1 point on a lower line, vertically aligned with the first point.
  - The first and last points must be horizontally between the points on the 2-point lines.
- **For all shapes**, width is the maximum horizontal distance between any 2 points. Height is the maximum vertical distance between any 2 points.
- **Unknown shapes** do not fit any definitions defined above. Print on one line: "Unknown with points: (X,Y); (X,Y); ...", substituting all of the points (X,Y) found for the shape, as they were encountered in the dataset: left to right (X), top to bottom (Y), in that order.
  - For example: "Unknown with points: (1,0); (5,0); (4,2); (4,4)"

Reminder: The student data sets show various shapes, including shapes which ALMOST work, but don't due to a rule violation. Your solution needs to correctly solve **all** of the student datasets.

problem 24

Music Management

9 points

Your best friend has asked you to help them organize their jumbled mess of top-40 music collection.

They want you to help them collect the songs into groupings by genre by suggesting ways you could rename the files to organize them, so they can visualize them.

Help bring order to the madness!

Input

You will receive inputs of music genres, each on their own line. END on a line by itself ends input.

```
Rock
Synth Pop
END
```

Output

Search your **files directory** (note, other files besides songs may be in your files directory!) for songs in your (legally purchased!) music library which match the genres given, and supply a list of those songs in the format of: GENREALLCAPSNOSPACES_TitleNoSpaces_Artist Name Unchanged.mp3

- Only output the Genre matched to the song, not all of the genres associated with the song (some songs may match multiple genres).
- The song title should have all spaces removed in your output.
- The artist name should not be altered at all except to remove any leading or trailing whitespace from it.
- Order your list, grouped by genre, in Ascending ASCII/Unicode order (**case matters!**). The files list should have read in from the OS in sorted order, but if your OS doesn't do that, you will need to sort before outputting.
- All music files in your file directory will be named following the pattern of: Song Title - Artist (s, comma delimited if more than one) - Genres (comma delimited)
- Note that many genres list "Pop" as part of the genre name (e.g. Country Pop, Pop Rap, Synth Pop, etc.).
- For example, if you are given a specific Pop genre to find, like Pop Rap, then only return songs in your list which have that specific genre.
- If you are given a generic genre, like Rock, then any genre which has Rock as part of the genre name (e.g. Country Rock, Alternative Rock, etc.) should be returned.

Your output should look like this:

```
ROCK_CruelSummer_TAYLOR SWIFT.mp3
ROCK_IRememberEverythingfeat.KM_ZACH BRYAN.mp3
ROCK_IsItOverNow(Taylor'sVersion)_TAYLOR SWIFT.mp3
ROCK_JChrist_LIL NAS X.mp3
ROCK_MyLoveMineAllMine_MITSKI.mp3
ROCK_NotThe1975_KNOX.mp3
ROCK_SaveMe(withLaineyWilson)_JELLY ROLL.mp3
ROCK_Seasons_THIRTY SECONDS TO MARS.mp3
ROCK_StrongEnoughfeat.B.Zimmerman_JONAS BROTHERS.mp3
ROCK_XXL_LANY.mp3
ROCK_gethimback!_OLIVIA RODRIGO.mp3
SYNTHPOP_Fix This feat. Bryce Vine_OLIVIA LUNNY.mp3
SYNTHPOP_XXL_LANY.mp3
```

Discussion

Reminder: have you run your solution against **all** of the student data sets?

Additional Examples

Input	Output
Trap Alternative Rock Electropop END	TRAP_IKNOW_TRAVIS SCOTT.mp3 TRAP_JChrist_LIL NAS X.mp3 TRAP_RichBabyDaddyfeat.SexyyRed_DRAKE.mp3 ALTERNATIVEROCK_Seasons_THIRTY SECONDS TO MARS.mp3 ALTERNATIVEROCK_XXL_LANY.mp3 ELECTROPOP_Feather_SABRINA CARPENTER.mp3 ELECTROPOP_FixThisfeat.BryceVine_OLIVIA LUNNY.mp3 ELECTROPOP_Strangers_KENYA GRACE.mp3 ELECTROPOP_Yes,I'mAMess_AJR.mp3

problem 25

HPEmari Damacy

10 points

The King has once again tasked the Prince with gathering up enough material to make more stars.

The King has spoken, get rollin'!

Input

You will receive exactly 2 input lines in your data file.

The first line will show the maximum top width of the rolled up shape (the largest width you will ever receive is 28).

The second line will contain all of the characters to roll up into your shape (note, we are showing them below on several lines, because we don't have as much space on paper as we do in a data file).

```
20
AAAAAAAABBCCCCCCCCCDDDDEEEEEEEEEEEEFFFFFFFGGGGGGGGGGGGGGHHHHHHHHIIIIIIIIIIIIIIIIJJJJ
JJJJJJKKKKKKKKKKKKKKKKKLLLLLLLLLLLLLMMMMMMMMMMMMMMMMMMMMNNNNNNNNNNNNNNNNOOOOOOOOOOOO
OOOOOOOPPPPQQQQQRRRRSSSS
```

Output

Starting with the leftmost character from the second line of your input, start "rolling" the characters into a ball-shape, assuming the characters are being rolled by a tiny little alien with magical powers working left to right on the characters.

By the time the ball shape has achieved the specified top-width length, the innermost characters should be the ones which were originally the leftmost characters on the line.

The innermost fold must always have a length of 8. The dataset has been crafted to show you that, with the A-characters starting on the left of the line, and ending up "rolled" up into the shape in the very center.

Any remaining characters from the line should be trailing off on the "ground" to the right, waiting for the Prince to roll them up into the ball.

```
MMMMMMMMMMMMMMMMMMML
N                   L
N IIIIIIIIIIIIIIH L
N J               H L
N J EEEEEEEEEEED H L
N J F           D H L
N J F AAAAAAAA D H L
N J F B         D H L
N J F BCCCCCCCCC H L
N J F             H L
N J FGGGGGGGGGGGGG L
N J                 L
N JKKKKKKKKKKKKKKKKK
N
NOOOOOOOOOOOOOOOOOOPPPPQQQQQRRRRSSSS
```

Discussion

The innermost fold must always have a length of 8, and the spiral must always progress from the innermost fold's left edge down, and then to the right, then up, then back to the left (etc.) in a spiral pattern.

There is a definitive algorithm that will help you programmatically create the rolled-up ball shape out of the line of characters. There are set relationships between the lengths of the lines, the heights of the lines, and how far out the "ball" grows with each revolution. Part of your challenge will be to figure that out and include it in your algorithm.

Reminder: have you run your solution against **all** of the student data sets?

problem 26

5 Nights At HPE

12 points

Our intrepid hero has spent four nights guarding some of HPE's buildings, and on this 5th night, he has discovered the last building was originally an old Freddy's ... and you know what ... he's noping out.

Help our hero plan his exit by reviewing potential escape routes and the location of the dangerous animatronics stalking the halls outside the control room.

Input

You will read comma delimited input on a single line. Our hero starts in the control room at position 0.0.

The plan includes:

- our hero's current running speed (in meters/second (m/s)),
- the chase speed of the nearest animatronic (in m/s),
- the current position of the nearest animatronic (in negative meters),
- the escape route's hallway length (in meters),
- locations of any tools: flashlights & security doors (not every escape plan will include every possible tool),
- and the exit's distance from the hero (a little bit further than the length of the hallway, due to poor construction).

The listed elements will always be preceded by their label (Run, Chase, Start, Length, Flashlight, Door, Exit), and will always appear in that order (except for Flashlight and Door, which may position swap).

Flashlight and Door are variable - both, or one, or neither may appear in an escape plan. There will never be more than one door or flashlight in an escape plan (if present at all). Positions and speeds will always be given with 1 decimal place.

```
Run:1.7m/s, Chase:1.8m/s, Start:-2.1m, Length:10.0m, Flashlight:5.1m, Door:8.2m, Exit:
10.1m
```

Output

Determine if the given plan will allow our hero to make it to the exit before they are caught. Following the rules listed in the discussion section, output two sets of data.

First show our hero's position at and after each clock tick of one second. Always include one decimal place. The time index starting the movement lines should always be 2 digits (left pad with a zero if needed).

Tool actions should be on their own line *after* a movement is shown.

Next, after the hero's time-lapsed plan, output a blank line.

Finally output the animatronic's time-lapsed plan. The time ticks on both plans end when our hero escapes or is caught.

The action notes for the hero when a tool is used should be:

- [flashlight used at x.x] or [door shut at x.x] (where x.x is the position of the tool).

The action notes for the animatronic when a tool is used should be:

- [blinded, stopped] for the flashlight, and [door, stopped] for a door.

If the animatronic is being affected by the flashlight, instead of moving for a time tick, print "blinded". If the animatronic is blocked by a door for a time tick, print "blocked".

If our hero escapes, add "MADE IT!" for the hero and "MISSED!" for the animatronic. If our hero is caught, add "OH NO!" for our hero and "CAUGHT!" for the animatronic.

```
HERO:
Time-01: 0.0->1.7
Time-02: 1.7->3.4
Time-03: 3.4->5.1
[flashlight used at 5.1]
Time-04: 5.1->6.8
Time-05: 6.8->8.5
[door shut at 8.2]
Time-06: 8.5->10.2
MADE IT!

ANIMATRONIC:
Time-01: -2.1->-0.3
Time-02: -0.3->1.5
Time-03: 1.5->3.3
[blinded, stopped]
Time-04: blinded
Time-05: blinded
Time-06: 3.3->5.1
MISSED!
```

Discussion

ESCAPE PLAN SIMULATION RULES

- Time ticks by in one second intervals. "Time-01" (from 0 to 1 second) is the first interval
- Our hero starts at position 0.0. The animatronic chasing them starts at least 1 meter behind the hero (negative)
- Run and chase speeds are constant
- Update positions of the runner and chaser for each interval, *then* determine consequences
- A flashlight to the face will not stop an animatronic during a time tick, but will blind (and stop) it for the NEXT 2 ticks
- A shut security door will *not* affect the animatronic *until they reach it*, which is where their position will end for that tick. (They may also reach the door exactly at the end of a tick.) They will be blocked for the NEXT 3 ticks while they break the door down.
- Items are used instantly as soon as they are reached (or passed during a movement)
- However, if a shut door is blocking the path between our hero and the chasing animatronic, a flashlight cannot be used, and no note of that action should appear in the plan
- In the case of a position tie, the win goes to the chasing animatronic. (Our hero doesn't escape if the animatronic reaches the Exit at the same time.)
- If the chasing animatronic will reach our hero during or at the end of a time tick, do not adjust their final positions to show the collision. Instead, show the ending position of our hero on that time index (less than or equal to the ending position of the animatronic), followed by OH NO!

Reminder: have you run your solution against **all** of the student data sets?

Additional Examples

The inputs are shown as broken up across lines, but that is simply because of space issues on the page. Your dataset inputs will always be on a single line. The outputs for each time tick must also be on a single line.

Input 1	Output 1	Input 2	Output 2

`Run:1.2m/s, Chase:1.9m/s, Start:-1.5m, Length:9.1m, Flashlight:5.1m, Door: 9.0m, Exit:9.2m`	`HERO:`
	`Time-01: 0.0->1.2`
	`Time-02: 1.2->2.4`
	`Time-03: 2.4->3.6`
	`OH NO!`
	
	`ANIMATRONIC:`
	`Time-01: -1.5->0.4`
	`Time-02: 0.4->2.3`
	`Time-03: 2.3->4.2`
	`CAUGHT!`
		`Run:2.9m/s, Chase:3.1m/s, Start:-6.3m, Length:9.9m, Door:9.1m, Flashlight: 3.2m, Exit:10.0m`	`HERO:`
			`Time-01: 0.0->2.9`
			`Time-02: 2.9->5.8`
			`[flashlight used at 3.2]`
			`Time-03: 5.8->8.7`
			`Time-04: 8.7->11.6`
			`[door shut at 9.1]`
			`MADE IT!`
			
			`ANIMATRONIC:`
			`Time-01: -6.3->-3.2`
			`Time-02: -3.2->-0.1`
			`[blinded, stopped]`
			`Time-03: blinded`
			`Time-04: blinded`
			`MISSED!`

problem 27

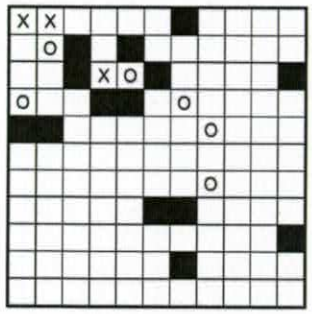
Mega Tic-Tac-Toe Ties

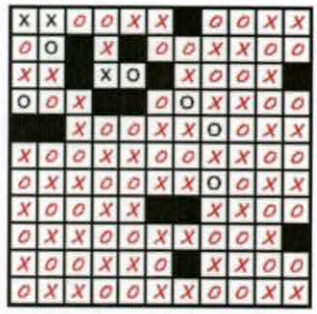
13 points

In this version of Mega Tic-Tac-Toe, two players work together to achieve a final arrangement where neither of them place more than two of their marks in a row on various square grids.

The first diagram shows hints to a completed board. The second diagram shows the final arrangement, where there are no 3-in-a-row of either X or O.

Your task is to determine X or O for every square in the grid, where there is no 3-in-a-row of either X or O.





Input

The first line is an integer N from 3 to 11, the size of the square grid. The next N lines contain N characters of the grid. 'X' and 'O' are known entries. '#' marks a wall (not used in the grid). A period '.' is an unknown square.

```
4
X...
XX..
....
O.#.
```

Output

Print the completed grid, replacing all periods with either X or O.

```
XOXO
XXOX
OXOX
OO#O
```

Discussion

All input grids are configured such that guesses are not needed for solving. That is, there will always be at least one 2-in-a-row to block. This puzzle was inspired by the "Puzzle Page" app.

Additional Examples

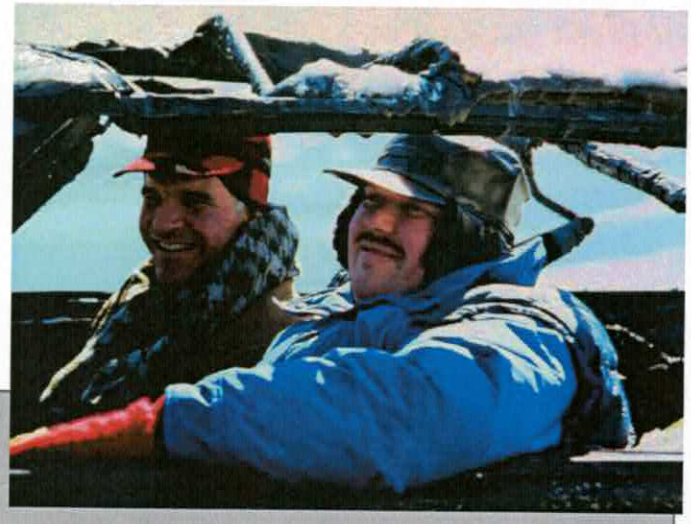
Input 1	Output 1	Input 2	Output 2
11		6	
XX....#....	XXOOXX#OOXX	O....#	OXOXO#
.O#.#.....	OO#X#OOXXOO	.....#	XOOXO#
..#XO#....#	XX#XO#XOOX#	#..#X.	#XX#XO
O..##.O....	OOX##OOXXOO	...XX.	XOOXXO
##.....O...	##XOOXXOOXX	.O....	XOOXOX
....	XOOXXOOXXOO	#.##XO	#X##XO
.......O...	OXXOOXXOOXX		
.....##....	XOOXX##XXOO		
.........#	OXXOOXXOOX#		
......#....	XOOXXO#XXOO		
....	OXXOOXXOOXX		

problem 28

Traveling Salesman

15 points

It's 2024 and it's all about sustainability.



You're given the task of finding the shortest route for your sales team to hit a number of cities.

You've got a couple of resources.

The first is a list of 100 cities in CSV (Comma Separated Values) format in the file **files/cities.csv**.

Here's an excerpt from the file. The format is City, State, Latitude, Longitude:

```
Albuquerque,New Mexico,35.12,-106.62
Anaheim,California,33.84,-117.87
Anchorage,Alaska,61.2176,-149.8997
Arlington,Texas,32.69,-97.13
Arlington,Virginia,38.88,-77.1
Atlanta,Georgia,33.76,-84.42
Aurora,Colorado,39.71,-104.73
Austin,Texas,30.31,-97.75
Bakersfield,California,35.36,-119
Baltimore,Maryland,39.3,-76.61
```

Note that the coordinates are given in degrees, since that's what we use on maps and globes.

The next resource you have is the following trig formula which gives the distance on a sphere between two (lat,lon) pairs:

```
d = ARCCOS((SIN((Lat1)) * SIN((Lat2))) + (COS((Lat1)) * COS((Lat2))) * (COS((Lon2) -
(Lon1)))) * Radius
```

For Earth, use the radius 6371km.

Your math library uses radians and not degrees, so you'll need to convert degrees to radians. Use pi = 3.1415926.

```
rad = deg * pi / 180.
```

Input

First read in the cities from the **files/cities.csv** file.

In your input data set, the first line is the number of cities, and the next lines are the names of the cities. This is also the starting route order.

```
6
Philadelphia
Jacksonville
Washington DC
Las Vegas
Sacramento
Colorado Springs
```

Traveling Salesman

continued

Output

First, calculate the distance in km of the route in the input order.

Print out the list of cities either in the initial order, or reversed, so that the first city printed is west of the last city printed. Then add ' = distance in km'. Truncate the final distance to an integer (do not truncate any values until printing this final total.)

Then compute the optimal routing between all the cities in the input list. That means checking each and every possible routing - a brute force solution.

- Here's a hint, if at any time you see the distance you've accumulated is greater than the best route you've found so far, stop, and try the next order.

If the initial route is the optimal route then print the word OPTIMUM, otherwise print RE-ROUTE followed by the optimal order, again with the first city west of the last city.

```
ROUTE: Colorado Springs:Sacramento:Las Vegas:Washington DC:Jacksonville:Philadelphia =
7687
RE-ROUTE: Sacramento:Las Vegas:Colorado Springs:Jacksonville:Washington DC:Philadelphia
= 5134
```

Discussion

Traveling Salesman problems are a class of problem when the amount of work grows exponentially with the number of data points.

All of the problems have been selected so that if you're careful when you're trying out possible routes, you can rule many out very quickly.

Your code should find the optimal routing in 30 seconds or less.

Reminder: Closely inspect **all** of the student data sets. Later data sets require routing 15 cities.

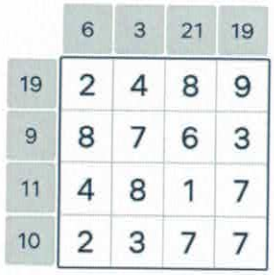
problem 29

Pineapple Packing

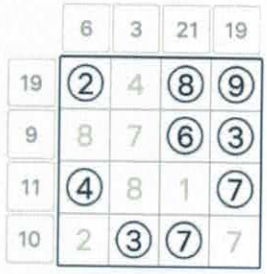
16 points

Penelope Peters is a professional pineapple packer. Each evening, she picks a pallet and evaluates its properties to properly populate it with packages of pineapples.

Unfortunately, each morning, she arrives to find the pallet has been fully populated with pineapple packages, but she knows if the pallet were lifted, the precarious placement would puncture the pallet and all the pineapples would perish.



	6	3	21	19
19	2	4	8	9
9	8	7	6	3
11	4	8	1	7
10	2	3	7	7



	6	3	21	19
19	②	4	⑧	⑨
9	8	7	⑥	③
11	④	8	1	⑦
10	2	③	⑦	7

Luckily, Penelope's plan of the perfect number of pineapples in each row and column is known . Given the fully populated pallet of pineapple packages (first diagram), remove packages until each row and column holds the proper pineapple placement (second diagram).

Input

The input is a map of the pallet. The first line is an integer N from 4 to 8, the size of the square grid. The next line begins with 3 spaces, followed by N 2-digit sums of the the pineapples in each column, separated by 1 space. On the last N lines, the first number is the 2-digit sum of the pineapples in the row; the next N 1-digit numbers are the current packages of pineapples on the pallet, separated by 2 spaces. (Packages hold from 1 to 9 pineapples.)

```
4
   06 03 21 19
19  2  4  8  9
09  8  7  6  3
11  4  8  1  7
10  2  3  7  7
```

Output

Reprint the map with the properly packed pallet, replacing all pulled packages with periods.

```
   06 03 21 19
19  2  .  8  9
09  .  .  6  3
11  4  .  .  7
10  .  3  7  .
```

Discussion

In the 4x4 above, on the third row, we must remove the 8, or we can't reach a sum of 11. Similarly, on the fourth row, we must keep the 3, or we can't reach a sum of 10. This puzzle was inspired by the "Number Sums" app.

Additional Example

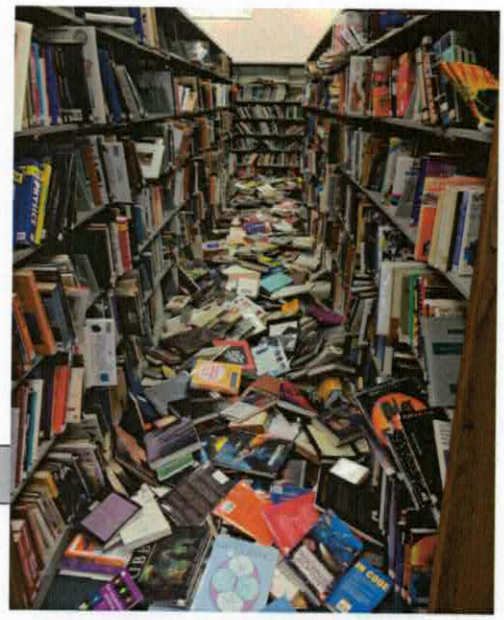
```
Input                          Output

6
   02 28 20 11 04 06              02 28 20 11 04 06
14  6  1  9  5  6  4        14     .  1  9  .  .  4
18  9  8  6  4  3  6        18     .  8  6  4  .  .
04  1  3  2  9  4  4        04     1  3  .  .  .  .
14  1  9  1  4  8  9        14     1  9  .  4  .  .
12  5  7  3  3  1  2        12     .  7  .  3  .  2
09  3  7  5  9  4  3        09     .  .  5  .  4  .
```

problem 30

Unstructured Data

20 points

Software applications collect lots and lots of data. Some programs collect lots of data generated by lots of other programs. Often times, the data from various sources has a variety of different structures. When data with various structures is collected together, it is called *Unstructured Data*.



It is possible to find really interesting and useful patterns in unstructured data. However, it is not always obvious how to process the data in a meaningful way. One simple way that humans examine unstructured data is by using queries. For example, we might query to find the age values of all entries in a data set with a certain zip code. The entries in the unstructured data could be web clicks, or traffic citations, or college applications, or all of the above mashed up together.

One simple way humans explore unstructured data is by using queries. A query could take the following form:

```
query key1, key2 for key2=value and/or key3=value
```

The *query* clause must have one or more key names separated by commas. The *for* clause must have one or more key=value specifications, separated by the conjunctions "and" or "or". The *for* clause cannot use both "and" and "or". Note that with the *or* conjunction, you could repeat a key, for example: age=28 or age=65.

Input

The first line of input is the name of the unstructured data file (see the Discussion section for details). This will be in the program's local directory, where it is running. The lines that follow are queries. The last line of input is the single word `exit`.

```
files/customer.csv
query name, age for zipcode="90210" and bald=No
query zipcode, age, employer for hair="blonde"
query name for hair="red" or beard=FALSE
exit
```

Output

The program should print each query, followed by the query results. The result is a comma-separated list of values for keys in the *query* clause, for the entries that match the *for* clause in the order that they occur in the unstructured data file. If an entry does not include a queried key name, the program should print `null`. If there are no matching entries, the program should print the phrase `no results`.

```
query name, age for zipcode="90210" and bald=No
"Tom", 34
"Li", null
query zipcode, age, employer for hair="blonde"
"82001", 17, null
query name for hair="red" or beard=FALSE
no results
```

Discussion

In the unstructured data file, we define an entry as a line of text in a file (100 lines/entries maximum). An entry line consists or one or more key/value pairs, separated by commas (maximum 100 keys per entry and 1000 characters per line). A key value may be either a quoted string (which may contain spaces), a number, or a boolean. A boolean value may be true, false, yes, no, or just the first letter of one of those words, upper or lower case. Here are some examples:

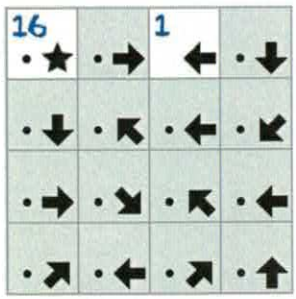
```
name="Tom",hair="black",age=34,zipcode="90210",bald=F,beard=T,citation="X3002FT839-31630722",employer="Acme"
url="http://tiny.url/not-a-real-url",age=17,occupation="hairdresser",hair="blonde",zipcode="82001"
zipcode="90210",vendor="Wally's
Widgetorium",amount=497.13,name="Li",product="WidgetWatch9000",bald=n
```

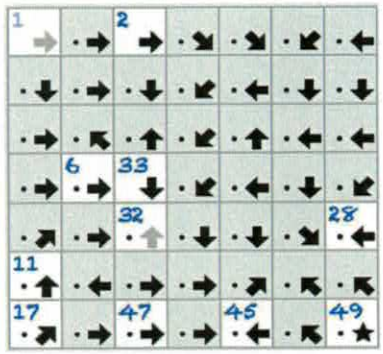
problem 31

Arrow Maze

25 points

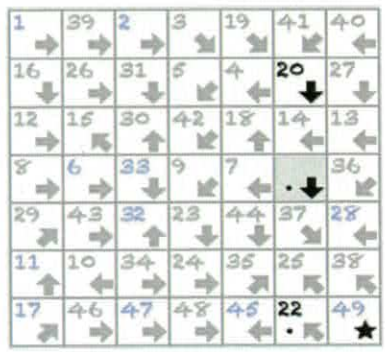
The first two diagrams show unfinished puzzles.





Some squares hold numbers. Most hold arrows. Each arrow points to the square that holds the next sequential number. The last square doesn't have an arrow.

The third diagram shows most of the solution to the second diagram.



Your task is to determine where all the numbers from 1 to N^2 go in the grid.

Each arrow points to the next square in the sequence, but that square could be anywhere along the path of the arrow vertically, horizontally, or diagonally.

Input

The first line of input is an integer N from 4 to 7, the size of the square grid. The next N lines contain a map of the grid. Each grid entry has 4 characters. The first 2 characters show the known number in that square, or 00 if the number is not known. The last 2 characters show the direction the arrow points, or XX for the final square without an arrow.

UU=Up, RR=Right, DD=Down, LL=Left. Diagonals are UL, UR, DR, DL.

Grid entries are separated by a space.

```
4
16XX 00RR 01LL 00DD
00DD 00UL 00LL 00DL
00RR 00DR 00UL 00LL
00UR 00LL 00UR 00UU
```

Output

Print the completed grid, as 2-digit numbers separated by spaces.

```
16 02 01 03
09 15 08 05
10 11 14 13
07 06 12 04
```

Additional Example

```
Input                                  Output
7
01RR 00RR 02RR 00DR 00DR 00DL 00LL     01 39 02 03 19 41 40
00DD 00RR 00DD 00DL 00LL 00DD 00DD     16 26 31 05 04 20 27
00RR 00UL 00UU 00DL 00UU 00LL 00LL     12 15 30 42 18 14 13
00RR 06RR 33DD 00DL 00LL 00DD 00DL     08 06 33 09 07 21 36
00UR 00RR 32UU 00DD 00DD 00DR 28LL     29 43 32 23 44 37 28
11UU 00LL 00RR 00RR 00UR 00UL 00UL     11 10 34 24 35 25 38
17UR 00RR 47RR 00RR 45LL 00UL 49XX     17 46 47 48 45 22 49
```

Inspiration for this problem comes from "Signpost" puzzles in the app "Genius Puzzles".