Title: [intermediate] challenge #1
Text: create a program that will allow you to enter events organizable by hour. There must be menu options of some form, and you must be able to easily edit, add, and delete events without directly changing the source code.

(note that by menu i dont necessarily mean gui. as long as you can easily access the different options and receive prompts and instructions telling you how to use the program, it will probably be fine)

Title: [intermediate] challenge #2
Text: create a short text adventure that will call the user by their name. The text adventure should use standard text adventure commands ("l, n, s, e, i, etc.").

for extra credit, make sure the program doesn't fault, quit, glitch, fail, or loop no matter what is put in, even empty text or spaces. These will be tested rigorously!

For super extra credit, code it in C

Title: [2/11/2012] Challenge #3 [intermediate]
Text: Welcome to cipher day!

Create a program that can take a piece of text and encrypt it with an alphabetical substitution cipher. This can ignore white space, numbers, and symbols.

for extra credit, make it encrypt whitespace, numbers, and symbols!

for extra extra credit, decode someone elses cipher!

Title: [2/12/2012] Challenge #4 [intermediate]
Text: create a calculator program that will take an input, following normal calculator input (5*5+4) and give an answer (29). This calculator should use all four operators.

For extra credit, add other operators (6(4+3), 3 ** 3, etc.)

Tit**le: [2/13/2012] Challenge #5 [intermediate]**
Text: Your challenge today is to write a program that can find the amount of anagrams within a .txt file. For example, "snap" would be an anagram of "pans", and "skate" would be an anagram of "stake".

Title: [2/14/2012] Challenge #6 [intermediate]
Text: create a program that can remove all duplicate strings from a .txt. file. for example, "bdbdb" -> "bd"

we are really sorry about this :(.. I just woke up now and am looking at this disaster. We promise to give a bonus question soon ...

for those who still have time, here is the modified question:

remove duplicate substrings.

Ex: aaajtestBlaBlatestBlaBla ---> aaajtestBlaBla

another example:

aaatestBlaBlatestBlaBla
aaathisBlaBlathisBlaBla

```
aaathatBlaBlathatBlaBla
aaagoodBlaBlagoodBlaBla
aaagood1BlaBla123good1BlaBla123
```

output desired:

```
aaatestBlaBla
aaathisBlaBla
aaathatBlaBla
aaagoodBlaBla
aaagood1BlaBla123
```

I am really sorry for the vagueness. Hopefully will not be repeated again :(

Title: [2/15/2012] Challenge #7 [intermediate]
Text: Write a program that draws a recursive image.

For example, a [Sierpinski triangle](http://www.zeuscat.com/andrew/chaos/sierpinski.clear.gif), a [Barnsley fern](http://underdown.files.wordpress.com/2007/10/fern.png), or a [Mandelbrot set fractal](http://upload.wikimedia.org/wikipedia/commons/thumb/2/21/Mandel_zoom_00_mandelbrot_set.jpg/322px-Mandel_zoom_00_mandelbrot_set.jpg) would be good drawings.

Any recursive image will do, but try to make them look fun or interesting.

Bonus points for adding a color scheme!

Please post a link to a sample image produced by your program, and above all, be creative.

Title: [2/16/2012] Challenge #8 [intermediate]
Text: Write a program that will print the english name of a value. for example, "1211" would become "one-thousand, two hundred, eleven".

for extra credit, allow it to read the english value of a number and output the integer.

input: one-hundred, four
output: 104

Title: [2/17/2012] Challenge #9 [intermediate]
Text: Write a program that will take a string ("I LIEK CHOCOLATE MILK"), and allow the user to scan a text file for strings that match. after this, allow them to replaces all instances of the string with another ("I quite enjoy chocolate milk. hrmmm. yes.")

Title: [2/18/2012] Challenge #10 [intermediate]
Text: On December 7, 2011, Neil deGrasse Tyson tweeted:

Need a distraction today? Not only does 12+1=11+2, but the letters “twelve plus one” rearrange to give you “eleven plus two”

Your task is to write a program that finds equations similar to Tyson’s that form anagrams both in their symbols and in their letters.

Title: [2/19/2012] Challenge #11 [intermediate]
Text: An “upside up” number is a number that reads the same when it is rotated 180°. For instance, 689 and 1961 are upside up numbers.

Your task is to find the next upside up number greater than 1961, and to count the number of upside up numbers less than ten thousand.

edit: since there is a confusion about 2 and 5, please consider them as "upside up" numbers for this problem. If you have already done without it, its ok. Sorry for the late reply.

Title: [2/20/2012] Challenge #12 [intermediate]
Text: Create a program that will factor a number. for example:

12 = 2 * 2 * 3

14 = 7 * 2

20 = 2 * 2 * 5

Title: [2/21/2012] Challenge #13 [intermediate]
Text: Create a program that will take any string and write it out to a text file, reversed.

input: "hello!"

output: "!olleh"

Title: [2/23/2012] Challenge #14 [intermediate]
Text: Your task is to implement the [sieve of Sundaram](http://en.wikipedia.org/wiki/Sieve_of_Sundaram) and calculate the list of primes to 10000.

[this](http://plus.maths.org/content/sundarams-sieve) is also an interesting article about it.

Title: [2/24/2012] Challenge #15 [intermediate]
Text: A 30x30 grid of squares contains 900 fleas, initially one flea per square.
When a bell is rung, each flea jumps to an adjacent square at random (usually 4 possibilities, except for fleas on the edge of the grid or at the corners).

What is the expected number of unoccupied squares after 50 rings of the bell? Give your answer rounded to six decimal places.

Title: [2/27/2012] Challenge #16 [intermediate]
Text: [Craps is a gambling game](http://en.wikipedia.org/wiki/Craps)

Your task is to write a program that simulates a game of craps.

Use the program to calculate various features of the game:

for example, What is the most common roll or average rolls in a game or maximum roll? What is the average winning percentage? Or you can make your own questions.

Title: [3/4/2012] Challenge #17 [intermediate]
Text: build an application that will search a web page for instances of sentences in quotations: "like this".

Title: [3/5/2012] Challenge #18 [intermediate]
Text: Screen scraping involves interacting with the terminal display of a currently running program. There are commercial screen scraping applications available for mainframe programs that provide a web interface on top of a dumb terminal program.

Write a program that will read the following from a text file to simulate the dumb terminal program. Each line represents a prompt to the user (always ends with a colon). Input constraints may be available for prompts. If they are, then they will always be surrounded in parentheses. The accepted input values will always be separated by a comma, and the value that is actually counted will be surrounded in square brackets.

Once you've parsed the text file, convert the data into an HTML form output file. If the prompt did not have any input constraints, then the input type is just a text. If the prompt contained input constraints and there are less than 5 options, then the input type are radio buttons. If there are 5 or more possible input values, then the input type is a dropdown.

Example:

Input File

```
Name:

Gender ([M]ale, [F]emale):

Position ([C]ashier, [D]eli Clerk, [M]anager, [P]roduce Clerk, [S]tock Person):
```

Output File (HTML)

```
<html>
<body>
<form>

Name:
<input type="text" name="name"/>
<br/>

Gender:
<input type="radio" name="gender" value="m"/> Male
<input type="radio" name="gender" value="f"/> Female
<br/>

Position:
<select name="position">
<option value="c">Cashier</option>
<option value="d">Deli Clerk</option>
<option value="m">Manager</option>
<option value="p">Produce Clerk</option>
<option value="s">Stock Person</option>
</select>
<br/>
<input type="submit" value="Submit"/>
</form>
</body>

</html>
```

Title: [3/7/2012] Challenge #19 [intermediate]
Text: Challenge #19 will use [The Adventures of Sherlock Holmes](http://www.gutenberg.org/cache/epub/1661/pg1661.txt) from [Project Gutenberg](http://www.gutenberg.org).

The Adventures of Sherlock Holmes is composed of 12 stories. Write a program that counts the number of words in each story. Then, print out the story titles ordered by its word count in descending order followed by how many words each story contains. Exclude the Project Gutenberg header and footer, book title, story titles, and chapters.

Title: [3/8/2012] Challenge #20 [intermediate]
Text: create a program that will take user input and tell them their age in months, days, hours, and minutes

sample output:

how old are you? 18

months : 216, days : 6480, hours : 155520, and minutes : 388800

Title: [3/9/2012] Challenge #21 [intermediate]
Text: This idea is to implement the [haar wavelet](http://en.wikipedia.org/wiki/Haar_wavelet) transform on an array of length 2^n . This algorithm is a critical algorithm for many image compression and image processing applications, and it is relatively simple both recursively and iteratively.

The solution should take in as input an array of floating-point values, and return an array of floating point values that in some sense implements the haar wavelet transform.

Title: [3/10/2012] Challenge #22 [intermediate]
Text: Orginal post by somebody from reddit.com/cpp

I always found that games make for fun learning projects.

One of my favorite projects for beginners is to make a 2D (top down) maze style game using text to represent the map and the player. Print out a 20x20 (or whatever size floats your boat) grid of characters where '#' characters represent impassible walls, and spaces or '.' characters represent empty passable spaces. Draw the player in place of one of the passable characters as a + sign, or some other distinct character. Every time the player presses an arrow key, check and see if the adjacent grid square is a passable square. If so, update the player's position and reprint the 20x20 map with the + sign in the new position.

What makes this project great is you can get more and more advanced with it as you progress:

*Come up with an initial victory condition. Maybe if the player moves from a starting square to a ending square on the map, he wins

*Maybe you can add some monsters (^ characters or something) which will kill the player if they land on the same square. Every time you reprint the map update the monster positions so that they try to move into the player's square.

*If you want to get more advanced, build in a loop that will keep printing the map and let the monsters update position even if the player hasn't pressed a key

*Maybe you want to add more levels so that when the player reaches the end point on the map, a new map loads

*Maybe you want to make the map bigger than 20x20 characters, but you only want to display 20x20 characters of the map at a time and simply change what portion of the larger map is printed as the player moves around

*Maybe you want the player to be able to shoot some kind of projectile (maybe a '@' character) at the monsters

for a bonus use file IO learning by creating the maps in text files and then reading them in when loading too.

Title: [3/13/2012] Challenge #23 [intermediate]
Text: At McDonalds' Restaurants, the Chicken McNugget meals are available in sizes of 6 McNuggets, 9 McNuggets, or 20 McNuggets. A number is a McNugget number if it can be the sum of the number of McNuggets purchased in an order (before eating any of them). Henri Picciotto [devised](http://en.wikipedia.org/wiki/Coin_problem#McNugget_numbers) the math of McNugget numbers in the 1980s while dining with his son at McDonald's, working the problem out on a napkin.

Your task is to determine all numbers that are not McNugget numbers.

Title: [3/14/2012] Challenge #24 [intermediate]
Text: Happy (Be-Lated) Pi Day! To celebrate, write a program that calculates a list of rational approximations of Pi. Output should look like:

3/1, 22/7, 333/106, 355/113, 52163/16604, 103993/33102, ...

Title: [3/15/2012] Challenge #25 [intermediate]
Text: Write a program to do the following:

input: a base ten (non-fractional) number at the command line

output: the binary representation of that number.

Title: [3/16/2012] Challenge #26 [intermediate]
Text: An employer wants you to store some information about his employees in an easy to read list form. He wants you to input the names, ages, and the pay of his employees. Unfortunately, he is terrible with computers, and he'll call you in every time he wants to change something he's going to call you in, unless you add an easy way to edit the information.

use this list for testing:

New Years Baby, 1, $12.00 per hour

satan, 666, $66.66 per hour

Harry potter, 18, $15.00 per hour

Tarzan the wild man, 30, $12.00 dollars per hour

Title: [3/17/2012] Challenge #27 [intermediate]
Text: Happy St. Patrick's Day! Write a program that accepts a year as input and outputs what day St. Patrick's Day falls on.

Bonus: Print out the number of times St. Patrick's Day falls on a Saturday for this century.

Sample Run:

Enter Year: 2012

St. Patrick's Day is on a Saturday.

Enter Year: 2010

St. Patrick's Day is on a Wednesday.

Title: [3/20/2012] Challenge #28 [intermediate]
Text: A [tetrahedral number](http://en.wikipedia.org/wiki/Tetrahedral_number) is is a figurate number that represents a pyramid with a triangular base and three sides.

Write a program to find the base of the tetrahedron that contains an input number of balls.

example: 169179692512835000 balls

Title: [3/22/2012] Challenge #29 [intermediate]
Text: This is a simple web frontend and web server (CGI, PHP, servlet, server component, etc.) exercise. Write a web frontend that contains a text area and button. Then write a program that accepts the contents of the text area and writes them out to a file. When the user clicks the button, the submission of the content can be either form-based, AJAX-based, or even websockets-based.

You can complete this project in several ways. You can write up the HTML yourself and submit the form to a program written in C/C++, Perl, Python, PHP, etc. You can do all the work in Javascript and hit a server using Node.js. You can also show off how easy it is to do this project using a Java/Python/Ruby/etc. web framework.

Title: [3/26/2012] Challenge #30 [intermediate]
Text: Most credit card numbers, and many other identification numbers including the Canadian Social Insurance Number, can be validated by an algorithm developed by Hans Peter Luhn of IBM, described in U. S. Patent 2950048 in 1954 (software patents are nothing new!), and now in the public domain. The Luhn algorithm will detect almost any single-digit error, almost all transpositions of adjacent digits except 09 and 90, and many other errors.

The Luhn algorithm works from right-to-left, with the right-most digit being the check digit. Alternate digits, starting with the first digit to left of the check digit, are doubled. Then the digit-sums of all the numbers, both undoubled and doubled, are added. The number is valid if the sum is divisible by ten.

For example, the number 49927398716 is valid according to the Luhn algorithm. Starting from the right, the sum is 6 + (2) + 7 + (1 + 6) + 9 + (6) + 7 + (4) + 9 + (1 + 8) + 4 = 70, which is divisible by 10; the digit-sums of the doubled digits have been shown in parentheses.

Your task is to write two functions, one that adds a check digit to a identifying number and one that tests if an identifying number is valid.

Title: [3/27/2012] Challenge #31 [intermediate]
Text: The complexity c(n) of a whole number is the smallest possible sum of a set of whole numbers that can be combined to make n using only addition, multiplication, and parentheses. Some examples:

* c(9) = 6, because 9 = 3x3 and 3+3 = 6
* c(11) = 8, because 11 = 2x5+1 and 2+5+1 = 8
* c(154) = 16, because 154 = 2x(2x3+1)x(3x3+2) and 2+2+3+1+3+3+2 = 16
* c(446) = 19, because 446 = 2x(2x3x(2x2x3x3+1)+1) and 2+2+3+2+2+3+3+1+1 = 19

In each case, only the formula with the smallest sum matters. For instance, 11 = 3+2x4, but since 3+2+4 = 9, and there's a formula with a sum of 8, this one doesn't matter. Since 5 is the highest number for which c(n) = n, 5 is the highest number that will appear in any formula.
Write a program that calculates the complexity of every whole number and finds at least one formula whose sum is that complexity [Check complexities for numbers through 10,000 here.](http://oeis.org/A005245/b005245.txt)

Post formulas for 956 (complexity 22) and 6458 (complexity 29).

Title: [3/28/2012] Challenge #32 [intermediate]
Text: [Tower of Hanoi](http://en.wikipedia.org/wiki/Tower_of_Hanoi) is a famous problem.

the challenge today is a very famous one where you are to write a function to calculate the total number of moves to solve the tower in fastest way possible

Title: [3/30/2012] Challenge #33 [intermediate]
Text: Write a program that will help you play poker by telling you what kind of hand you have.

input

The first line of input contains the number of test cases (no more than 20). Each test case consists of one line - five space separated cards. Each card is represented by a two-letter (or digit) word. The first character is the rank (A,K,Q,J,T,9,8,7,6,5,4,3 or 2), the second character is the suit (S,H,D,C standing for spades, hearts, diamonds and clubs). The cards can be in any order (but they will not repeat).

Output

For each test case output one line describing the type of a hand, exactly like in the list above.
Title: [3/31/2012] Challenge #34 [intermediate]
Text: Your task today is show the implementation of two sorting algorithms [Stooge sort](http://en.wikipedia.org/wiki/Stooge_sort) and [Bogosort](http://en.wikipedia.org/wiki/Bogosort) in anyway you like!

Title: [4/3/2012] Challenge #35 [intermediate]
Text: Imagine you are given a function flip() that returns a random bit (0 or 1 with equal probability). Write a program that uses flip to generate a random number in the range 0...N-1 such that each of the N numbers is generated with equal probability. For instance, if your program is run with N = 6, then it will generate the number 0, 1, 2, 3, 4, or 5 with equal probability.

N can be any integer >= 2.

Pseudocode is okay.

You're not allowed to use rand or anything that maintains state other than flip.

Title: [4/5/2012] Challenge #36 [intermediate]
Text: Because I want to watch the world burn, write a program that accepts a sentence as input and outputs the sentence in leetspeak. [Here](http://en.wikipedia.org/wiki/Leet) is a link for a leetspeak translation table. Since leetspeak has multiple character selections per letter, randomly pick the character selection. The challenging part will be to be a resourceful developer and write a utility or use an existing application to save the table into a format that you will load in your program to do the translation. Oh yeah, if the input sentence contains one !, for the love of God translate that into !!!!11!!!1! ;)

Title: [4/8/2012] Challenge #37 [intermediate]
Text: Enter an integer for the number of iterations, and create a program that prints out a sierpinski triangle.

[First 4 iterations as an example](http://i.imgur.com/fjlTg.png)

Title: [4/10/2012] Challenge #38 [intermediate]
Text: [Reverse Polish Notation](http://en.wikipedia.org/wiki/Reverse_Polish_notation)(RPN) is a mathematical notation where every operator follows all of its operands. For instance, to add three and four, one would write "3 4 +" rather than "3 + 4". If there are multiple operations, the operator is given immediately after its second operand; so the expression written "3 − 4 + 5" would be written "3 4 − 5 +" first subtract 4 from 3, then add 5 to that.

Transform the algebraic expression with brackets into RPN form.

You can assume that for the test cases below only single letters will be used, brackets [ ] will not be used and each expression has only one RPN form (no expressions like a*b*c)

Test Input:

(a+(b*c))

((a+b)*(z+x))

((a+t)*((b+(a+c)) ^ (c+d)))

Test Output:

abc*+

ab+zx+*

at+bac++cd+ ^ *

Title: [4/12/2012] Challenge #39 [intermediate]
Text: Today's challenge is to determine if a number is a [Kaprekar Number](http://mathworld.wolfram.com/KaprekarNumber.html)

Title: [4/16/2012] Challenge #40 [intermediate]
Text: Write a program that computes the [Kaprekar chain](http://mathworld.wolfram.com/KaprekarRoutine.html) for a given starting number, and compute the longest possible [Kaprekar chain](http://mathworld.wolfram.com/KaprekarRoutine.html)

Title: [4/19/2012] Challenge #41 [intermediate]
Text: Write a program that will use the [FOIL method](http://www.algebrahelp.com/lessons/simplifying/foilmethod/pg2.htm) to solve multiplying binomials. Your program will accept a binomial algebraic formula as input and you will parse the data, use the FOIL method to reduce the formula, and print out the solution. You decide how you want to represent exponents (could use caret, or just write out the value after the variable).

Sample Run:

```
  Enter Binomials:  (2x + 6)(7x + 3)
  Result:  14x^2 + 48x + 18
  Enter Binomials: (2x2 + 3x)(5x2 + 9x)
  Result:  10x4 + 33x3 + 27x2
```

Bonus: Support trinomials

Title: [4/23/2012] Challenge #42 [intermediate]
Text: We have it easy nowadays when it comes to numbers. It's so simple to add them and subtract them that we don't even consider that it wasn't always so. Even multiplication is not that tough! And you can represent very large numbers very easily, if you want to write down a million, you just write "1000000". That's only seven digits!

It wasn't so easy for the Romans. They had to make due with their clumsy Roman numerals. Think how hard it is to add Roman numerals together. If you wanted to add XXVIII with LXXXII, you would have to smush them together to form LXXXXXVIIIII, then you would have to realize that XXXXX is equal to L and IIIII is equal to V, and turn it to LLVV, and then you'd have to shorten that to CX. Look how much work to just add 28 and 82! And just imagine trying to write a million with Roman numerals: you'd literally have to write down one thousand copies of M!

But Roman numerals aren't without advantages: they at least look very pretty. I think we can all agree that Rocky V looks way cooler than Rocky 5.

So in this challenge, we honor the romans. Your task is to write a function that can add together two Roman numerals supplied as strings. Example: roman_addition("XXVIII", "LXXXII") returns "CX".

The easiest way to do this would obviously be to just to convert the roman numerals to integers and then convert the sum of those integers back to a Roman numeral. But since the Romans couldn't do that, you can't either! Write the function so that it performs the task similarly to how it might have been done in ancient Rome, either with the "smushing together" method I described above, or another method of your choosing (don't worry about efficiency). But at no point shall you convert the numerals to decimal or binary integers! Imagine if you lived in ancient Rome and decimal numbers hadn't been invented yet, how would you do it?

The output of this function should as "minimal" as possible. I.e. if the answer is XVII, it should output XVII, not XIIIIIII or VVVII.

Real Roman numerals sometimes uses a trick to make the numbers shorter: you put a smaller numeral in front of a larger numeral to represent the difference between the two. So for instance, 4 would be written as "IV", 9 as "IX" or 1949 as "MCMIL". For the purposes of this problem, lets ignore that. 4 is "IIII", 9 is "VIIII" and 1949 is "MDCCCCXXXXVIIII". The numbers become longer this way, but it makes much more sense if all the numerals are "in order". Also, the exact rules for how this trick worked was never rigorous, and changed over time.

For reference, here's the different single numerals and their value:

```
    I = 1
    V = 5    = IIIII
    X = 10   = VV
    L = 50   = XXXXX
    C = 100  = LL
    D = 500  = CCCCC
    M = 1000 = DD
```

Bonus 1: Write a function that does subtraction. Example: roman_subtraction("CX", "LXXXII") returns "XXVIII"

Bonus 2: Write a function that performs multiplication with Roman numerals, again without ever converting them to regular integers. Example: roman_multiplication("XVI", "VII") returns "CXII".

Title: [4/24/2012] Challenge #43 [intermediate]
Text: Inspired by [this Wikipedia article](http://en.wikipedia.org/wiki/Doomsday_rule), which was found in the comments in [this video](http://www.reddit.com/r/videos/comments/rrx9a/15_year_old_plays_dubstep_song_on_the_piano_after/c486ph7), I thought it would be cool to come up with a programme which uses the Doomsday algorithm to determine which day of the week a certain date falls on.

Bonus: Now, learn to perform the algorithm mentally, and create a tester programme! Write another programme which takes parameters (e.g. the desired range of years), give a random date (e.g. if I input that I want a date this year it may output (5/6/2012); if I say I want a date between 1900 and 2100 it may output (17/3/2046)) and then ask for you to enter the day of the week and have the programme tell you if you are correct.

Title: [4/25/2012] Challenge #44 [intermediate]
Text: *The only winning move is not to play* - Joshua

Let's take our minds off this Global Thermonuclear War business, and lets play som Tic-Tac-Toe! Make a Tic-Tac-Toe playing AI and have it either play games against you or against itself.

If it plays against you, it should always win or draw, regardless of who goes first. If it plays against itself, it should always draw.

Here's how the beginning of a game might look in a terminal window (this is just a suggestion, yours can look however you want it to look, you could even make a GUI if you really wanted to):

```
    Lets play tic-tac-toe!

    Do you wish to play first? Yes

       |   |
    ---+---+---
       |   |
    ---+---+---
       |   |

    What is your move? 2 2

       |   |
    ---+---+---
       | x |
    ---+---+---
       |   |

    My move is 1 1

     o |   |
    ---+---+---
       | x |
    ---+---+---
       |   |

    What is your move? 1 3

     o |   |
    ---+---+---
       | x |
    ---+---+---
     x |   |
```

And so on. Everything after the question marks are inputs from the user.

Title: [4/27/2012] Challenge #45 [intermediate]
Text: When linguists study ancient and long dead languages, they sometimes come upon a situation where a certain word only appears once in all of the collected texts of that language. Words like that are obviously very bothersome, since they are exceedingly hard to translate (there's almost no context to what the word might mean).

Such a word is refered to as a *[hapax legomenon](http://en.wikipedia.org/wiki/Hapax_legomenon)* (which is Greek for roughly "word once said"). The proper plural is *hapax legomena*, but they are frequently refered to as just "hapaxes".

However, a hapax legomenon doesn't just need to be a word that appears only once in an entire language; they can also be words that appears only once in a single work, or the body of work of an author. Lets take Shakespeare as an example. In all the works of Shakespeare, the word "correspondance" occurs only in one place, the beginning of Sonnet 148:

>*O me! what eyes hath love put in my head,*
>*Which have no correspondence with true sight,*
>*Or if they have, where is my judgment fled,*
>*That censures falsely what they see aright?*

Now, "correspondance" is 14 letters long, which is a pretty long word. It is, however, not the longest hapax legomenon in Shakespeare. The longest by far is *[honorificabilitudinitatibus](http://en.wikipedia.org/wiki/Honorificabilitudinitatibus)* from *Love's Labour's Lost* (drink a couple of shots of whiskey and try and pronounce that word, I dare you!)

[Here is a link](http://www.gutenberg.org/cache/epub/100/pg100.txt) to a text-file containing the Complete Works of William Shakespeare (it's 5.4 mb big), provided by the good people of Project Gutenberg. Write a program that analyses that file and finds all words that

1. Only occur once in the entire text
2. Are longer than "correspondance", i.e. words that are 15 letters long or longer.

***
Bonus: If you do the first part of this problem, you will likely come up with a list of words that cannot be said to be "true" hapax legomenon. For instance, you might have found the word "distemperatures" which appears only once in *The Comedy of Errors*. But that is simply the plural of *distemperature*, and *distemperature* appears in *A Midsummer's Night Dream*, so "distemperatures" cannot be said to be a "true" hapax. Same thing with the word *superstitiously*: it also occurs only once but *superstitious* occurs many times. Even the example I used above can be said to not be a true hapax, because while *correspondance* only appears once, variations of *correspond* appears a number of times.

Modify your program to identify and make it detect when a word appears twice or more in a simple variation, like a plural or adverbial form (hint: words ending in "s", "ly", "ing" and "ment"), so that it can sort it out. Then, find the *true* number of hapax legomena in Shakespeare that are longer than 14 characters. By my count (which may very well be wrong), there are less than 20 of them.

Title: [4/30/2012] Challenge #46 [intermediate]
Text: Consider this game: Write 8 blanks on a sheet of paper. Randomly pick a digit 0-9. After seeing the digit, choose one of the 8 blanks to place that digit in. Randomly choose another digit (with replacement) and then choose one of the 7 remaining blanks to place it in. Repeat until you've filled all 8 blanks. You win if the 8 digits written down are in order from smallest to largest.

Write a program that plays this game by itself and determines whether it won or not. Run it 1 million times and post your probability of winning.

Assigning digits to blanks randomly lets you win about 0.02% of the time. Here's a python script that wins about 10.3% of the time. Can you do better?

```
    import random
    def trial():
        indices = range(8)  # remaining unassigned indices
        s = [None] * 8      # the digits in their assigned places
        while indices:
             d = random.randint(0,9)    # choose a random digit
             index = indices[int(d*len(indices)/10)]  # assign it an index
             s[index] = str(d)
             indices.remove(index)
        return s == sorted(s)
    print sum(trial() for _ in range(1000000))
```

Title: [5/2/2012] Challenge #47 [intermediate]
Text: Given a string containing the English word for one of the single-digit numbers, return the number *without using any of the words in your code*. Examples:

```
eng_to_dec('zero') # => 0
eng_to_dec('four') # => 4
```

Note: there is no right or wrong way to complete this challenge. Be creative with your solutions!

Title: [5/4/2012] Challenge #48 [intermediate]
Text: Your task is to write a program that implements the [Trabb Pardo Knuth](http://en.wikipedia.org/wiki/Trabb_Pardo%E2%80%93Knuth_algorithm) algorithm.

Title: [5/7/2012] Challenge #49 [intermediate]
Text: Your task today is to create a program that graphically plots some equation y = f(x), in some specified range of values for x.

The output can be whatever you want: if you want to output it as an image, that's fine, but if you don't want to deal with graphical libraries, you can just as well just output the plot as text, either to the terminal or to a text-file. You do not need to include axes in your plot.

For instance, if you wished to plot a simple sine wave (i.e. y = sin(x)) for x values from -2*pi to 2*pi, you could either output an image ([like this](http://i.imgur.com/NcEwT.png)), or print out something like this:

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

Note that the point of this challenge is just to plot the functions, not necessarily to write a program that can parse a mathematical equation. It's perfectly acceptable to "hard-code" the function you want to plot into the code. Also, I've used a sine wave as an example, but you can use whatever equation you want.

***

Bonus: If you *do* choose to output the plot as an image, make the plot into an animated gif by introducing a variable that changes frame by frame. For instance, [here is an animated gif](http://i.imgur.com/06BmQ.gif) of y = k*sin(x) as k varies from 1 to -1 and then back again (i.e. for each frame, k takes a different value, starting at 1, going to -1 and then back to 1 again), and [here is an animated gif](http://i.imgur.com/ZXjqQ.gif) of y = sin(k*x) as k varies from 1 to 10 and then back again.

Again, I used a sine wave as an example, but you may plot whatever equation you want.

Title: [5/9/2012] Challenge #50 [intermediate]
Text: Given an absolute path, write a program that outputs an ASCII tree of that directory.

Example output here: [HERE](http://www.acooke.org/cute/UnixComman0.html)

Note: 'tree' utility is not allowed.

Extra credit: Limit the depth of the tree by variable n.

Title: [5/11/2012] Challenge #51 [intermediate]
Text: Brainfuck is an extremely minimalistic programming language. The memory consists of a large array of bytes, the "tape", which is manipulated by moving around a single tape pointer. The 8 commands are:

Character	Action
&lt;	move the pointer to the left
&gt;	move the pointer to the right
+	increment the byte the pointer is pointing at (mod 256)
-	decrement the byte the pointer is pointing at (mod 256)
[	if the data which the tape pointer is pointing at is 0, jump *forward* to the command after the matching square bracket. Otherwise, just continue to the next command
]	if the data which the tape pointer is pointing at is *not* 0, jump *backwards* to the command after the matching square bracket. Otherwise, just continue to the next command
,	read a character from the input and store it into the current pointer byte
.	output the current pointer byte as an ascii character

Any other character is ignored and treated as a comment

`[ ... ]` thus make a kind of while loop, equivalent to something like "while(data[pointer] != 0) { ... }".
The brackets match like parentheses usually do, each starting one has a matching ending one. These loops can be nested inside other loops.

Write a program that reads a brainfuck program and its input, interprets the code, and returns the output.

More information, including a "Hello World" program, can be found [on wikipedia](http://en.wikipedia.org/wiki/Brainfuck).

If you've written your program successfully, try running this and see what pops out:

```
++++++++++[>>++++++>+++++++++++>++++++++++>+++++++++>+++>+++++>++++>++++++++>+[<
]<-]>>+++++++.>+.-.>+++.<++++.>>+++++++.<<++.+.>+++++.>.<<-.>---.<-----.-.+++++.
>>>+++.-.<<-.<+..----.>>>>++++++++.>+++++++..<<<<+.>>>>-.<<<<.++++.------.<+++++
.---.>>>>>.<<<++.<<---.>+++++++.>>>>+.<<<-.--------.<<+.>>>>>>+++.---.<-.<<<<---.
<.>---.>>>>>>.
```

Title: [5/14/2012] Challenge #52 [intermediate]
Text: After years of study, scientists have discovered an alien language transmitted from a faraway planet. The alien language is very unique in that every word consists of exactly L lowercase letters. Also, there are exactly D words in this language.

Once the dictionary of all the words in the alien language was built, the next breakthrough was to discover that the aliens have been transmitting messages to Earth for the past decade. Unfortunately, these signals are weakened due to the distance between our two planets and some of the words may be misinterpreted. In order to help them decipher these messages, the scientists have asked you to devise an algorithm that will determine the number of possible interpretations for a given pattern.

A pattern consists of exactly L tokens. Each token is either a single lowercase letter (the scientists are very sure that this is the letter) or a group of unique lowercase letters surrounded by parenthesis (and). For example: (ab)d(dc) means the first letter is either a or b, the second letter is definitely d and the last letter is either d or c. Therefore, the pattern (ab)d(dc) can stand for either one of these 4 possibilities: add, adc, bdd, bdc.

Please note that sample i/p and o/p is given in the link below

[Link](http://code.google.com/codejam/contest/90101/dashboard#s=p0)

Title: [5/16/2012] Challenge #53 [intermediate]
Text: A simple pseudo-random number generator looks like this:

```
    s(0) = 123456789
    s(n) = (22695477 * s(n-1) + 12345) mod 1073741824
```

So each number is generated from the previous one.

Using this generator, generate 10 million numbers (i.e. s(0) through s(9,999,999)) and find the 1000 largest numbers in that list. What is the sum of those numbers?

Try to make your solution as efficient as possible.

Title: [5/19/2012] Challenge #54 [intermediate]
Text: For this challenge, create the worlds simplest IM client. It should work like this: if Alice on computer A wants to talk to Bob on computer B, she should start the IM program as a server listening to some port. Bob should then start the program on his end, punch in computer A's IP address with the right port. The two computers should now be connected to each other and Alice and Bob should be able to communicate by sending short strings to each other. Example conversation seen on Alice's computer:

```
You: "Hey Bob!"
Bob: "Hey Alice!"
Bob: "I can't believe I successfully connected!"
You: "Isn't it cool?"
Bob: "It really is!"
```

Same conversation seen on Bob's computer:

```
Alice: "Hey Bob!"
You: "Hey Alice!"
You: "I can't believe I successfully connected!"
Alice: "Isn't it cool?"
You: "It really is!"
```

If you don't have to computers lying around, or just want to make it easier for yourself, it is perfectly allowed to run both programs on the same computer and connect to "localhost".

If you want to, you can design a very simple GUI for this, but that is not necessary. If you can finagle this to work in a terminal, that is perfectly fine.

Title: [5/21/2012] Challenge #55 [intermediate]
Text: Write a program that will allow the user to enter two characters. The program will validate the characters to make sure they are in the range '0' to '9'. The program will display their sum. The output should look like this.

```
INPUT .... OUTPUT

3 6 ........ 3 + 6 = 9
4 9 ........     4 + 9 = 13
0 9 ........ 0 + 9 = 9
g 6 ........ Invalid
7 h ........ Invalid
```

Title: [5/23/2012] Challenge #56 [intermediate]
Text:
At some point or another, most programmers will encounter problems in text processing that has a very elegant solution using [regular expressions](http://en.wikipedia.org/wiki/Regex). But regular expressions can also be over-relied on and abused, and make code unreadable. There is a lot of truth in the old saying "Some people, when confronted with a problem, think 'I know, I'll use regular expressions.' Now they have two problems." So today, lets embrace those two problems and pound some regular expressions into submission!

Your task is to write a regular expression that will match a string if and only if the number of vowels (both upper and lower case) in that string is exactly divisible by 3. For instance, the regular expression will not match the string "Hello!", because there are only two vowels there, "e" and "o" (and 2 is not divisible by 3), but it will match "Daily programmer" because there are six vowels, "D**ai**l**y** pr**o**gr**a**mm**e**r" (and 6 is divisible by 3).

For the purposes of this problem, the vowels of the English language are "A", "E", "I", "O", "U" and "Y" (in both upper and lower cases, obviously).

Here are a few strings you can test your regular expressions on:

1. "Friends, Romans, countrymen, lend me your ears!"
2. "Double, double, toil and trouble; Fire burn and cauldron bubble."
3. "Alas, poor Yorick! I knew him, Horatio. A fellow of infinite jest, of most excellent fancy."
4. "To be, or not to be- that is the question: Whether 'tis nobler in the mind to suffer The slings and arrows of outrageous fortune Or to take arms against a sea of troubles, And by opposing end them."
5. "Everybody stand back! I know regular expressions."

Title: [5/25/2012] Challenge #57 [intermediate]
Text: Given a 3x3 table where 1 represents on and 0 represents off:

 ABC
A010
B111
C011

Where "inverted match" is defined as a case where the values at the coordinates in the format of (X, Y) and (Y, X) are the same, the inverted matches are as follows:

[[(A, B), (B, A)], [(A, C), (C, A)], [(B, C), (C, B)]]

Of these, the matches that have a value of 1 are:

[[(A, B), (B, A)], [(B, C), (C, B)]]

Therefore, there are 2 sets of inverted matches that have a value of 1.

Find the amount of inverted matches in the table in table(below) with a value of 1.

Table:

` ABCDEFGHIJKLMNOPQRST`
`A11110101111011100010`
`B10010010000010001100`
`C01101110010001000000`
`D10110011001011101100`
`E10100100011110110100`
`F01111011000111010010`
`G00011110001011001110`

`H01111000010001001000`
`I01101110010110010011`
`J00101000100010011110`
`K10101001100001100000`
`L01011010011101100110`
`M10110110010101000100`
`N10001111101111110010`
`O11011010010111100110`
`P01000110111101101000`
`Q10011001100010100000`
`R11101011100110110110`
`S00001100000110010101`
`T01000110011100101011`

Title: [5/28/2012] Challenge #58 [intermediate]
Text: For the easy part of today's challenge, we considered numbers that are palindromes in different bases. For this problem, lets only concern ourselves with numbers that are palindromes in base 10.

Define a function P(N) that takes as input a number N, and returns the smallest base 10 palindrome larger than N (i.e. it returns the "next" palindrome after N). So, for instance:

P(808) = 818
P(999) = 1001
P(2133) = 2222

What is P(3^39)?

***
BONUS: What is P(7^100)

Title: [6/2/2012] Challenge #59 [intermediate]
Text: Given a binary matrix like this:

```
    0 1 1 1 1 0
    1 0 0 1 1 1
    1 0 1 1 1 1
    1 1 1 1 1 1
    0 1 1 1 1 0
```

Output the clues for a [nonogram puzzle](http://en.wikipedia.org/wiki/Nonogram) in the format of "top clues, empty line, bottom clues", with clues separated by spaces:

```
    3
    1 2
    1 3
    5
    5
    3

    4
    1 3
    1 4
    6
    4
```

That is, count the contiguous groups of "1" bits and their sizes, first in columns, then in rows.

Title: [6/4/2012] Challenge #60 [intermediate]
Text: Write a program or a function that can print out arbitrarily sized smiley face in ascii art. The smiley face can be however you want, but the eyes can't be single points (that is, they have to be circles at large size). Your program should be able to take in an integer between 16 and 1000 that represents the dimensions to render the face.

[Here](http://www.reddit.com/r/dailyprogrammer_ideas/comments/sq93c/dynamic_ascii_smiley_face_intermediatehard/c4g485v) is a sample output.

Title: [6/6/2012] Challenge #61 [intermediate]
Text: Today you should implement a function that all of us programmers depend heavily on, but most never give a second thought as to how it's actually coded: the square root function.

Write a function that given a number as input, will return the square root of that number, in floating point.

Obviously, you are not allowed to use the library version of sqrt() in the implementation of your function. Also, stay away from log() and exp(). In fact, don't use any mathematical functions that aren't the basic arithmetic ones (addition, subtraction, multiplication, division) or bit operations (though you can of course still use operators that compares numbers with each other, like "less than", "equal to", "more than", etc.)

Title: [6/8/2012] Challenge #62 [intermediate]
Text: Find all the subsets of a set of non-negative integers where the largest number is the sum of the remaining numbers, and return a count of the number of them. For instance, for the set { 1, 2, 3, 4, 6 } the subsets are 1+2=3, 1+3=4, 2+4=6, and 1+2+3=6, so the result is 4 subsets. Apply your program to the set { 3, 4, 9, 14, 15, 19, 28, 37, 47, 50, 54, 56, 59, 61, 70, 73, 78, 81, 92, 95, 97, 99 }.

Your task is to write a program to solve the challenge.

Bonus: you might like to apply your solution to the set of prime numbers less than 2^10

Title: [6/11/2012] Challenge #63 [intermediate]
Text: You can use the reverse(N, A) procedure defined in [today's easy problem](http://www.reddit.com/r/dailyprogrammer/comments/uw14f/6112012_challenge_63_easy/) to completely sort a list. For instance, if we wanted to sort the list [2,5,4,3,1], you could execute the following series of reversals:

```
A = [2, 5, 4, 3, 1]

reverse(2, A)      (A = [5, 2, 4, 3, 1])
reverse(5, A)      (A = [1, 3, 4, 2, 5])
reverse(3, A)      (A = [4, 3, 1, 2, 5])
reverse(4, A)      (A = [2, 1, 3, 4, 5])
reverse(2, A)      (A = [1, 2, 3, 4, 5])
```

And the list becomes completely sorted, with five calls to reverse(). You may notice that in this example, the list is being built "from the back", i.e. first 5 is put in the correct place, then 4, then 3 and finally 2 and 1.

Let s(N) be a random number generator defined as follows:

```
s(0) = 123456789
s(N) = (22695477 * s(N-1) + 12345) mod 1073741824
```

Let A be the array of the first 10,000 values of this random number generator. The first three values of A are then 123456789, 752880530 and 826085747, and the last three values are 65237510, 921739127 and 926774748

Completely sort A using only the reverse(N, A) function.

Title: [6/13/2012] Challenge #64 [intermediate]
Text: Find the longest palindrome in the following 1169-character string:

`Fourscoreandsevenyearsagoourfaathersbroughtforthonthisconta`
`inentanewnationconceivedinzLibertyanddedicatedtotheproposit`
`ionthatallmenarecreatedequalNowweareengagedinagreahtcivilwa`
`rtestingwhetherthatnaptionoranynartionsoconceivedandsodedic`
`atedcanlongendureWeareqmetonagreatbattlefiemldoftzhatwarWeh`
`avecometodedicpateaportionofthatfieldasafinalrestingplacefo`
`rthosewhoheregavetheirlivesthatthatnationmightliveItisaltog`
`etherfangandproperthatweshoulddothisButinalargersensewecann`
`otdedicatewecannotconsecratewecannothallowthisgroundThebrav`
`elmenlivinganddeadwhostruggledherehaveconsecrateditfarabove`
`ourpoorponwertoaddordetractTgheworldadswfilllittlenotlenorl`
`ongrememberwhatwesayherebutitcanneverforgetwhattheydidhereI`
`tisforusthelivingrathertobededicatedheretotheuInfinishedwor`
`kwhichtheywhofoughtherehavethusfarsonoblyadvancedItisrather`
`forustobeherededicatedtothegreattdafskremainingbeforeusthat`
`fromthesehonoreddeadwetakeincreaseddevotiontothatcauseforwh`
`ichtheygavethelastpfullmeasureofdevotionthatweherehighlyres`
`olvethatthesedeadshallnothavediedinvainthatthisnationunsder`
`Godshallhaveanewbirthoffreedomandthatgovernmentofthepeopleb`
`ythepeopleforthepeopleshallnotperishfromtheearth`

Your task is to write a function that finds the longest palindrome in a string and apply it to the string given above.

Title: [6/15/2012] Challenge #65 [intermediate]

Text: Anyone who've tried to get through the *A Song of Ice and Fire* books written by George R.R. Martin (the basis for the HBO show Game of Thrones) knows that while the books are generally excellent, there are *a lot* of characters. A staggering number, in fact, and it can be very hard to remember who's who and who is related to who and who had an incestual relationship with what sister or brother.

So, today, we make that a little bit easier! What follows at the end here is a list of 50+ characters from the books and a list detailing how their related. Each character is given a two-letter code (for instance "AA" or "BQ") and a specification of their gender ("M" or "F"), and then what follows is a list detailing how they're related to the other characters.

To make things simple, there's only one "basic" relationship, which is "A is parent to B", written as "->". So, for instance, if Arya Stark has the code "AI" and Eddard Stark has the code "AB", then "AB->AI" means "Eddard Stark is parent to Arya Stark". Each person may have 0, 1 or 2 parents specified somewhere in the list, but no more.

(I should point out here that this family tree contains *no spoilers*. This is the family tree as it stands in the beginning of Book 1, though some of the characters you wont meet until much later. For those of you who've read the books or seen the show, please don't put any spoilers in the comments, even in hidden spoiler text.)

Write a program that parses this list, and can then answer questions about the relationships between people. Here are a list of functions you should write:

* ancestors(person) which gives the direct ancestors of that person (parents, grand-parents, great-grand-parents, etc.). For instance, ancestors("Asha Greyjoy") should return ["Balon Greyjoy", "Alannys Harlaw", "Quellon Greyjoy"]. What is the result to ancestors("Daenerys Targaryen")?

* descendants(person) which gives you the direct descendants of that person (children, grand-children, great-grand-children, etc.). What is the result of descendants("Jaehaerys Targaryen")?

* brothers(person) and sisters(person) which gives the brothers and sisters of the specified person (including half-brothers and half-sisters, though you could write special functions for full siblings and half siblings if you want).

* aunts(person) and uncles(person) which gives you the aunts and uncles of the specified person.

* cousins(person), which gives you the 1st cousins of the specified person.

* Bonus: As a bonus, write a function called relationship(person1, person2) which returns person1's relationshipt to person2 as a string (i.e. "Grandfather", "1st cousin", "Brother", "Great uncle", "Not related" etc.). As with all bonuses on /r/dailyprogrammer, this is entirely optional. EDIT: Since this chart gives no indication about who is married to whom, you can safely exclude all familial relationships that somehow involves marriage. That means that relationship("Eddard Stark", "Catelyn Tully") should return "Not related", and you can also skip all brother/sister/mother/father in-laws. Only relationships "by blood", so to speak.

***

And now, here is the family tree of some of the major characters in A Song of Ice and Fire:

```
AA = Rickard Stark (M)       AB = Eddard Stark (M)        AC = Catelyn Tully (F)
AD = Brandon Stark (M)       AE = Benjen Stark (M)        AF = Jon Snow (M)
AG = Robb Stark (M)          AH = Sansa Stark (F)         AI = Arya Stark (F)
AJ = Bran Stark (M)          AK = Rickon Stark (M)        AL = Hoster Tully (M)
AM = Minisa Whent (F)        AN = Edmure Tully (M)        AO = Lysa Tully (F)
AP = Jon Arryn (M)           AQ = Robert Arryn (M)        AR = Tytos Lannister (M)
AS = Tywin Lannister (M)     AT = Joanna Lannister (F)    AU = Kevan Lannister (M)
AV = Cersei Lannister (F)    AW = Jamie Lannister (M)     AX = Tyrion Lannister (M)
AY = Robert Baratheon (M)    AZ = Joffrey Baratheon (M)   BA = Myrcella Baratheon (F)
BB = Tommen Baratheon (M)    BC = Lancel Lannister (M)    BD = Steffon Baratheon (M)
BE = Stannis Baratheon (M)   BF = Selyse Florent (F)      BG = Shireen Baratheon (F)
BH = Renly Baratheon (M)     BI = Jaehaerys Targaryen (M) BJ = Aerys Targaryen (M)
BK = Rhaella Targaryen (F)   BL = Rhaegar Targaryen (M)   BM = Elia Martell (F)
BN = Rhaenys Targaryen (F)   BO = Aegon Targaryen (M)     BP = Viserys Targaryen (M)
BQ = Daenerys Targaryen (F)  BR = Quellon Greyjoy (M)     BS = Balon Greyjoy (M)
BT = Euron Greyjoy (M)       BU = Victarion Greyjoy (M)   BV = Urrigon Greyjoy (M)
BW = Aeron Greyjoy (M)       BX = Rodrik Greyjoy (M)      BY = Maron Greyjoy (M)
BZ = Asha Greyjoy (F)        CA = Theon Greyjoy (M)       CB = Alannys Harlaw (F)
---------------------------------------------------------------------------------------
AA->AB, AA->AD, AA->AE, AB->AF, AB->AG, AB->AH, AB->AI, AB->AJ, AB->AK, AC->AG,
AC->AH, AC->AI, AC->AJ, AC->AK, AL->AC, AL->AN, AL->AO, AM->AC, AM->AN, AM->AO,
AO->AQ, AP->AQ, AR->AS, AR->AU, AS->AV, AS->AW, AS->AX, AT->AV, AT->AW, AT->AX,
AU->BC, AV->AZ, AV->BA, AV->BB, AY->AZ, AY->BA, AY->BB, BD->AY, BD->BE, BD->BH,
BE->BG, BF->BG, BI->BJ, BI->BK, BJ->BL, BJ->BP, BJ->BQ, BK->BL, BK->BP, BK->BQ,
BL->BN, BL->BO, BM->BN, BM->BO, BR->BS, BR->BT, BR->BU, BR->BV, BR->BW, BS->BX,
BS->BY, BS->BZ, BS->CA, CB->BX, CB->BY, CB->BZ, CB->CA
```

Title: [6/18/2012] Challenge #66 [intermediate]
Text: Maxiphobic heaps are a variant of leftist heaps. Like leftist heaps, maxiphobic heaps are represented as binary trees arranged according to the heap property that every element is less than or equal to its two children. Find-min looks at the root of the tree, delete-min discards the root and merges its two children, and insert merges the existing tree with a singleton tree containing the new element.

The key to leftist trees and maxiphobic trees is the merge operation. In leftist trees, the rank of each left child is always less than the rank of its sibling, where the rank of a node is defined as the length of its right spine, and the merge operation enforces this invariant.

In maxiphobic trees, the merge operation is implemented by comparing the roots of the two trees. The smaller root survives as the root of the merged tree. That leaves three sub-trees: the tree that lost in the comparison of the two roots, and the child sub-trees of the winner. They are merged by first merging the two smaller trees, where the size of a tree is determined as the number of elements it contains, then attaching that merged tree along with the remaining tree as the children of the new root.

The name maxiphobic, meaning “biggest avoiding,” refers to the merge of the two smaller sub-trees.

Your task is to write functions that implement maxiphobic trees.

Tit**le: [6/20/2012] Challenge #67 [intermediate]**
Text: You are given a list of 999,998 integers, which include all the integers between 1 and 1,000,000 (inclusive on both ends) in some unknown order, with the exception of two numbers which have been removed. By making only *one pass* through the data and using only a constant amount of memory (i.e. O(1) memory usage), can you figure out what two numbers have been excluded?

Note that since you are only allowed one pass through the data, you are not allowed to sort the list!

EDIT: clarified problem

Title: [6/22/2012] Challenge #68 [intermediate]
Text: You are given numbers N and H. H=floors of the building N=number of telephones. Your must find the MINIMUM amount of throws you need(A) to find out the highest floor from which the telephone will not break when thrown. For example when you have one phone and 10 floors, you start from the lowest floor and start throwing your phone- if it breaks the highest floor is 1, if not we throw it from the second floor-if it breaks the highest floor is 2....and so on. The problem asks you to find out the MINIMUM amount of throws it will take to find that floor. If N=1, then the answer is always H-because you start from the bottom and go up throwing the phone from every floor till it breaks.

Now when N>1 then that's a whole different story. If you have 2 phones you can throw one from the middle of the building, if it breaks, you only need to check floors 1-(middle-1) with your remaining phone, if it doesn't break you need to check floors (middle+1)-top floor.

Title: [6/26/2012] Challenge #69 [intermediate]
Text: During World War I, the German army used a very clever pen and paper cipher called the [ADFGVX cipher](http://en.wikipedia.org/wiki/ADFGVX_cipher), and your task today is to implement functions to both encrypt and decrypt messages using this cipher. What follows is a rather lengthy description of how it works (you can also find a description in that wikipedia link), but in essence it is actually quite simple.

Here is how it works:

The cleartext (the message that is to be encrypted) could consist of characters selected from an alphabet of 36 characters. For the purposes of today's problem, that alphabet will be:

    "ABCDEFGHIKLMNOPQRSTUVWXYZ0123456789 "

That is, it will be the regular uppercase alphabet except for the letter J (if there's a J in the cleartext, replace it with an I), ten numbers, and a space character. That makes 25 + 10 + 1 = 36 different characters.

The ciphertext will consist of only 6 different characters, namely the characters "ADFGVX". Supposedly these were selected because they are quite unlike each other in morse code, lessening the risk for errors in transmission.

The key for the encryption and decryption consists of two parts: first one scrambled version of the cleartext alphabet (i.e. some permutation of "ABCDEFGHIKLMNOPQRSTUVWXYZ0123456789 "), called the *substition key*. Second we need a *transposition key* which is a just a codeword of some sort.

Lets illustrate the encryption of the cleartext "Brake me out of jail on the 21st." using the substitution key "R3FLMX7KWQ69D4Y5NOZ STV2EH8AP1ICBGU0" and the transposition key "PROGRAMMER"

Encryption proceeds as follows: we begin by putting the cleartext in A suitable format, so that it only contains characters from the alphabet. Our cleartext then becomes "BRAKE ME OUT OF IAIL ON THE 21ST". As you can see, all characters have been put into uppercase, the "J" have been replaced by an "I", and all characters not in the alphabet (in this example, only the period ".") have been removed.

Next we put the substitution key into a 6x6 table with the cipher chars "ADFGVX" as row and column headers, like this:

```
    A D F G V X
   +------------
  A | R 3 F L M X
  D | 7 K W Q 6 9
  F | D 4 Y 5 N O
  G | Z   S T V 2
  V | E H 8 A P 1
  X | I C B G U 0
```

Each letter of the cleartext now gets replaced by two letters representing the row and column of the character in this square. So for instance, 'A' becomes 'VG' (because it's in the V row and the G column), 'B' becomes 'XF', 'C' becomes 'XD', etc. This is called "fractioning" the text. If we convert our cleartext using this method it becomes:

```
  B  R  A  K  E     M  E     O  U  T     O  F
  XF AA VG DD VA GD AV VA GD FX XV GG GD FX AF GD

  I  A  I  L     O  N     T  H  E     2  1  S  T
  XA VG XA AG GD FX FV GD GG VD VA GD GX VX GF GG
```

Note that the space character is encoded as GD.

Next, this fractioned text is put into a table with the transposition key as headers, as follows:

```
  P R O G R A M M E R
  -------------------
  X F A A V G D D V A
  G D A V V A G D F X
  X V G G G D F X A F
  G D X A V G X A A G
  G D F X F V G D G G
  V D V A G D G X V X
  G F G G F G F A D F
```

The last row didn't quite fit (it was six letters short), so we add in some random characters, in this case "FGFADF", to fill it out. Now the columns are sorted in alphabetical order of the header characters:

```
  A E G M M O P R R R
  -------------------
  G V A D D A X F V A
  A F V G D A G D V X
  D A G F X G X V G F
  G A A X A X G D V G
  V G X G D F G D F G
  D V A G X V V D G X
  G D G F A G G F F F
```

As you can see, the sorting is "stable", i.e. when there are two or more characters are identical in the transposition key, they keep the original order they had. So in this example, there are three R's and two M's, and they are in the same order relative to each other both before and after the transposition.

Now, finally, we simply read off the table *column by column* to get our ciphertext. This is the final result:

```
    GADGVDGVFAAGVDAVGAXAGDGFXGGFDDXADXAAAGXFVGXGXGGVGFDVDDDFVVGVFGFAXFGGXF
```

To decrypt, reverse the operations described here.

Title: [6/29/2012] Challenge #70 [intermediate]
Text: Implement the [hyperoperator](http://en.wikipedia.org/wiki/Hyperoperation#Definition) as a function hyper(n, a, b), for non-negative integers n, a, b.

hyper(1, a, b) = a + b, hyper(2, a, b) = a * b, hyper(3, a, b) = a ^ b, etc.

Bonus points for efficient implementations.

Title: [7/2/2012] Challenge #71 [intermediate]
Text: Before I get to today's problem, I'd just like to give a warm welcome to our two new moderators, [nooodl](http://www.reddit.com/user/nooodl) and [Steve132](http://www.reddit.com/user/Steve132)! We decided to appoint two new moderators instead of just one, because rya11111 has decided to a bit of a break for a while.

I'd like to thank everyone who applied to be moderators, there were lots of excellent submissions, we will keep you in mind for the next time. Both nooodl and Steve132 have contributed some excellent problems and solutions, and I have no doubt that they will be excellent moderators.

Now, to today's problem! Good luck!

***

The famous Fibonacci sequence is defined as follows: starting with 0 and 1, each number in the sequence is defined as the sum of the previous two numbers. The sequence starts:

```
    0,1,1,2,3,5,8,13,21,34,55,89,144,233,377,...
```

The list here is zero-based, so f(0) = 0, f(1) = 1, f(2) = 1, and so on.

Less famous is the so-called tribonacci sequence, which is like the Fibonacci sequence, except that it starts with 0,0,1 and every new number is defined as the sum of the previous three numbers. It starts:

```
    0,0,1,1,2,4,7,13,24,44,81,149,274,504,927,...
```

Continuing the pattern, there are also tetranacci sequence (which sums the four previous numbers) and the pentanacci sequence (which sums the five previous numbers). They begin:

```
    0,0,0,1,1,2,4,8,15,29,56,108,208,401,773,...

    0,0,0,0,1,1,2,4,8,16,31,61,120,236,464,...
```

These sequences are usually referred to as "higher order Fibonacci sequences". Note that if the order of the sequence is K (i.e. when K = 2 you get the standard Fibonacci numbers, and when K = 3, you get the tribonacci numbers), then the sequence starts out with K - 1 zeroes and then one 1.

Your task is to implement a function f(K, N) which returns the N^th fibonacci number of the order K. That is, f(2, N) would return values in the regular Fibonacci sequence, f(3, N) returns values in the tribonacci sequence, and so on.

What is f(100, 10000) mod 10^8 ?

Bonus: What is f(3^13 , 5^10) mod 10^8 ?

Title: [7/4/2012] Challenge #72 [intermediate]
Text: An X-ray illuminator is the bright plate that doctors put filters over in order to view x-ray images.

In our problem, we are going to place various sizes of red and blue tinted cellophane randomly on top of a finite x-ray illuminator.

If a given part of the illuminator is covered by only red filters, then the light is red. If it is covered by only blue filters, then the light is blue.
If it is covered by a mixture of red and blue filters, the light will be a shade of purple.

Given some set of red and blue sheets, what is the total area of all the purple regions?

Specification:
Each piece of cellophane is guaranteed to be an positive integer number of centimeters wide and tall, and will be placed at an integer coordinate
on the illuminator.

The input file will contain the following:
First, an integer n <= 1024 specifying how many pieces of cellophane there are

Then n lines for each piece of cellophane, where each line contains a character 'R' or 'B' for the color of the cellophane sheet, then two positive integers x,y for the position of the upper-left corner of the sheet, then two positive integers w,h for the width and height of the sheet.

IMPORTANT: Here are the constraints on the dimensions: 1 <= x+w <= 4096,1<=y+h<=4096,1<=w<=4095,1<=h<=4095...in other words, a sheet should always lie within the boundry of the 4k by 4k board.

Here is an example input and output

input file:

```
    3
    R 0 0 5 5
    R 10 0 5 5
    B 3 2 9 2
```

Here is an ascii art example visualizing that input:

```
    RRRRR     RRRRR
    RRRRR     RRRRR
    RRRPPBBBBBPPRRR
    RRRPPBBBBBPPRRR
    RRRRR     RRRRR
```

expected program output:

```
    8
```

Write a program to count the number of purple blocks given an input file.

For testing, here are some test files I generated:

* http://codepad.org/5HtVUwCT
* http://codepad.org/2KXIrWlj
* http://codepad.org/Weyka1Pp

I am a fallible mod, but I believe the correct answer for those files should be 13064038,15822641,15666634 respectively.

Title: [7/6/2012] Challenge #73 [intermediate]
Text: Write a program that, given an ASCII binary matrix of 0's and 1's like this:

```
0000000000000000
0000000000000000
0000011001110000
0000001111010000
0000011001110000
0000011011100000
0000000000110000
0000101000010000
0000000000000000
0000000000000000
0000000000000000
```

Outputs the smallest cropped sub-matrix that still contains all 1's (that is, remove all borders of 0's):

```
01100111
00111101
01100111
01101110
00000011
10100001
```

Title: [7/9/2012] Challenge #74 [intermediate]
Text: In [his paper describing the so-called "Dancing Links" algorithm](http://arxiv.org/abs/cs/0011047) for solving exact cover problems (press the PDF link to see the full paper), Donald Knuth describes a rather fascinating data-structure, basically a [sparse binary matrix](http://en.wikipedia.org/wiki/Sparse_matrix) implemented using [doubly linked lists](http://en.wikipedia.org/wiki/Linked_list#Singly.2C_doubly.2C_and_multiply_linked_lists). The linked lists are two-dimensional, so instead of just having "left" and "right" links, it has "up" and "down" links as well).

In other words, if you are given a matrix of ones and zeroes, like this:

```
0 0 1 0 1 1 0
1 0 0 1 0 0 1
0 1 1 0 0 1 0
1 0 0 1 0 0 0
0 1 0 0 0 0 1
0 0 0 1 1 0 1
```

You create a data-structure that looks something [like this](http://i.imgur.com/RXAvI.png).

The link that's marked "h" is the head link, indicating the "head" of the data-structure. The links to the right of the head link are the column headers, indicating the columns. The number that's in the column header indicates how many ones there are in that column (and thus how many links there are in that particular column). Storing those numbers is entirely optional.

The rest of the structure is created from the 0s and 1s of the matrix. If there's a 1 in the input matrix, there's a link in the data structure, if there's a 0 in the input matrix, then there's no link.

As an example, you'll notice in the input matrix that there are 1s in the third, fifth and sixth columns, so in the finished data structure, there are links in the third, fifth and sixth columns. Each link in the matrix has left and right links (to the previous and next items in the same row) and up and down links (to the previous and next items in the same column). If there are no links in the left/right/up/down, the link "wraps around" to the other side of the row or column.

While this data-structure might look huge and daunting at first glance, it turns out that it is actually quite nimble. Once constructed, rows and columns can be removed and put back with surprising ease and elegance. Indeed, when you visualize that happening in your head, it really seems as if the links are dancing.

Your task today is this: given a matrix of ones and zeroes, construct this complicated linked list data-structure. You may assume that no row or column in the original input matrix of 1s and 0s consist entirely of 0s: there's always going to be at least one 1 in every row or column.

As a bonus, you may also implement any number of the following functions that operate on the data structure:

* remove_column(X): If X is a link in the matrix data-structure, completely remove X's column from the matrix, while still maintaining the correct structure (i.e. fix all the links so they're pointing where they should point when the column is removed). Note that X can be any link in a column or a column header.

* remove_row(X): Same thing as the previous function, only this time it removes the row instead of the column

* restore_column(X): If X is a link in a column that was previously removed using remove_column(X), this function restores the column to the matrix. That is, if X is a link in the matrix, first calling "remove_column(X)" and then "restore_column(X)" should leave the matrix as it was originally.

* restore_row(X): Same thing as the previous function, only this time it restores a row that had previously been removed using remove_row(X).

For the last two functions, you may want to check Knuth's paper for a neat trick that restores previously deleted items in linked lists.

EDIT: For the remove/restore functions, you can assume that if you've removed some number of rows/columns by calling remove_row/remove_column, that the calls to restore_row/restore_column will be called *in the reverse order*. I.e. if you first removed column 1, then column 2, then column 3, you would first restore column 3, then column 2, then column 1.

Also, just to be clear: if you can't get your head around the remove/restore functions, remember that they are just a bonus and totally optional. Constructing the data-structure is the main point of this excercise.

Title: [7/12/2012] Challenge #75 [intermediate] (Build System)
Text: First off, I'd like to apologize for posting this 12 hours late, I'm a little new to my mod responsibilities. However, with your forgiveness, we can go onward!

Everyone on this subreddit is probably somewhat familiar with the C programming language.
Today, all of our challenges are C themed! Don't worry, that doesn't mean that you have to solve the challenge in C.

In C, that the compiler is usually invoked on each of the source files in a project to produce an object file for each source file.
Then, these object files are linked together to produce a binary or a dynamic or static library. Usually, something like a Makefile or an IDE does this job. By specifying
the source code and project settings in some kind of configuration file, the user tells the build system tools how to make the final executable from code.

Your job is to implement your own build system that will read a project description file and then build a project. Use the simple build-system description language I've come up with below, and extend it as you see fit! Here's how it works:

Each line of the input file is treated as a seperate command, where each command modifies or changes the build system in some way. Trailing or leading whitespace or blank lines do not matter. Commands are as follows:

Commands to set the target:

```
exe <file>
lib <file>
```

This says that the current build target is an executable named <file>, or a static lib named <file>. All subsequent commands affect this build target until it is changed.

Commands to set flags:

```
ldflags <flag1> <flag2> <flag3> ... <flagn>
cflags <flag1> <flag2> <flag3> ... <flagn>
```

ldflags appends <flags> to the linker flags for the current build target
cflags appends <flags> to the compiler flags for the current build target

Commands to set dependencies:

```
link <file>
```

This says to append <file> to the list of linked libraries for the current build target. This is used for dependency resolution.

Commands to set source files.
If a line does not contain a command and is not blank, then that line is interpreted as the filename of a C source file to add to the current build target's source list.

Here is an example input file:

```
lib libhello.a
    cflags -O3 -DHELLO_POSIX
    hello.c
    hello_win32.c
    hello_posix.c

exe hello
    cflags -O3
    hello_main.c
    link libhello.a
```

This should compile and link a library libhello.a from the three source files, with HELLO_POSIX as a compile definition, and then compile and link ./hello using that library.

BONUS POINTS:
You get major bonus points if your tool does minimal rebuilds...that is, if it only compiles out-of-date files and goes in dependency order instead of file layout order.

Title: [7/13/2012] Challenge #76 [intermediate] (Probability graph)

Text: Write a function `graph(f, low, high, tests)` that outputs a probability graph of the function `f` from range `low` to `high` (inclusive) over `tests` tests (i.e., counting the frequencies of `f()` outputs). `f` takes no arguments and returns an integer, `low`, `high` and `tests` are all integer values. For example, a function `f` that simulates two-dice rolls:

```
def two_dice():
    return random.randint(1, 6) + random.randint(1, 6)
```

Then `graph(f, 2, 12, 10000)` should output something roughly like:

```
  2: ##
  3: #####
  4: #######
  5: ###########
```

```
 6: #############
 7: #################
 8: #############
 9: ###########
10: ########
11: #####
12: ##
```

For bonus points, output the graph with the numbers on the bottom and the bars drawn vertically.

Title: [7/16/2012] Challenge #77 [intermediate] (Last digit of factorial)
Text: The factorial of 10 is 3628800. The last non-zero digit of that factorial is 8.

Similarly, the last non-zero digit of the factorial of 10^3 is 2.

Compute the last non-zero digit of the factorial of 10^9 .

***

Bonus: Compute the last non-zero digit of the factorial of 10^100 .

Title: [7/18/2012] Challenge #78 [intermediate] (Dependency Planner)
Text: Working on planning a large event (like a wedding or graduation) is often really difficult, and requires a large number of dependant tasks. However, doing all the tasks linearly isn't always the most efficient use of your time. Especially if you have multiple individuals helping, sometimes multiple people could do some tasks in parallel.

We are going to define an input set of tasks as follows. The input file will contain a number of lines, where each line has a task name followed by a colon, followed by any number of dependencies on that task. A task is an alphanumeric string with underscores and no whitespace

For example, lets say we have to eat dinner. Eating dinner depends on dinner being made and the table being set. Dinner being made depends on having milk, meat and veggies. Having milk depends on going to the fridge, but meat and veggies depend on buying them at the store.
buying them at the store depends on having money, which depends on depositing ones paycheck.... this scenario would be described in the following input file. Note task definitions can appear in any order and do not have to be defined before they are used.

```
eat_dinner: make_dinner set_table
        make_dinner: get_milk get_meat get_veggies
        get_meat: buy_food
        buy_food: get_money
        get_veggies: buy_food
        get_money: deposit_paycheck
```

Write a program that can read an input file in this syntax and output all the tasks you have to do, in an ordering that no task happens before one of its dependencies.

Title: [7/18/2012] Challenge #79 [intermediate] (Plain PGM file viewer)
Text: Write a program that converts a ["plain" `.pgm` file](http://netpbm.sourceforge.net/doc/pgm.html) passed from stdin to an ASCII representation easily viewable in a terminal. If you're too lazy to read through the specification, the format should be simple enough to reverse-engineer from an example file:

```
P2
# feep.pgm
24 7
```

```
15
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  3  3  3  3  0  0  7  7  7  7  0  0 11 11 11 11  0  0 15 15 15 15  0
0  3  0  0  0  0  0  7  0  0  0  0  0 11  0  0  0  0  0 15  0  0 15  0
0  3  3  3  0  0  0  7  7  7  0  0  0 11 11 11  0  0  0 15 15 15 15  0
0  3  0  0  0  0  0  7  0  0  0  0  0 11  0  0  0  0  0 15  0  0  0  0
0  3  0  0  0  0  0  7  7  7  7  0  0 11 11 11 11  0  0 15  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

* The top line, `P2`, is there to **identify** the file as a plain .pgm file.
* Lines with a **#** in front of them are **comments**, and should be ignored.
* The first two numbers in the file are the **width** and **height**.
* The third number, 15 here, is the **maximum grayscale value** in the image: here, this means 15 is full white, and lower numbers are darker, 0 being pure black.
* Thereafter, a **(*width* x *height*) grid** specifying the image itself follows.

Your program should use ASCII symbols to represent different grayscale values. Assuming the text is black on a white background, you could use a gradient like this one:

```
" .:;+=%$#"
```

Converted, the example image would look something like this:

```
 ....  ;;;;  ====  ####
 .     ;     =     #  #
 ...   ;;;   ===   ####
 .     ;     =     #
 .     ;;;;  ====  #
```

Title: [7/23/2012] Challenge #80 [intermediate] (Poker hands)
Text: Your intermediate task today is to write a program that can identify a hand in poker.

Let each hand be represented as a string composed of five different cards. Each card is represented by two characters, "XY", where X is the rank of the card (A, 2, 3, 4, 5, 6, 7, 8, 9, T, J, Q or K) and Y is the suit of the card (H, D, C or S).

So, for instance, "AH" would be the Ace of Hearts, "2C" would be the 2 of Clubs, "JD" would be the Jack of Diamonds, "TS" would be the Ten of Spades, and so on. Then a hand with a full house could be represented as "2C 2H TS TH TC" (a pair of twos and three tens).

Write a program that takes a string like this and prints out what type of hand it is. So, for instance, given "2C 2H TS TH TC" it would print out "Full house". Note that the cards will not necessarily be in any kind of particular order, "2C 2H TS TH TC" is the same hand as "TC 2C 2H TS TH".

For reference, here are the different possible hands in poker, from most valuable to least valuable. Your program should be able to recognize all of these:

* Royal flush: a hand with a Ten, Jack, Queen, King and Ace in the same suit
* Straight flush: a hand with five cards of consecutive rank in the same suit
* Four of a kind: a hand with four cards of the same rank
* Full house: a hand with a pair and a three of a kind
* Flush: a hand with all cards the same suit
* Straight: a hand with five cards of consecutive rank
* Three of a kind: a hand with three cards of the same rank

* Two pair: a hand with two pairs
* Pair: and hand with two cards of the same rank
* High card: a hand with nothing special in it

Obviously, any one hand can qualify for more than one of these; every royal flush is obviously also a straight flush, and every straight flush is obviously also a flush. But you should only print out the kind with the most value, so "2H 3H 4H 5H 6H" should print out "Straight flush", not "Flush".

***

Bonus: write a function that given two different poker hands tells you which hand is the winner. When there are apparent ties, standard poker rules apply: if both players have a pair, the player with the highest pair wins. If both have pairs of the same rank, the player with the highest card not in the pair wins (or second highest, or third highest, if there are more ties). Note that poker hands can be absolute ties: for instance, if two players both have flushes in different colors but with identical ranks, that's an absolute tie, and your function should return with that result.

Title: [7/25/2012] Challenge #81 [intermediate] (Local Minimization)
Text: For a lot of the questions today we are going to be doing some simple numerical calculus. Don't worry, its not too terrifying.

Write a function fmin that can take in a real-valued function f(x) where x is a vector in 3D space (bonus points for N-D).

xout=fmin(f,x0) should return a local minimum of f close to x0.

Example in Python

```
        %f is a function with 1 3-vector
    def f(x):
            return x[0]**2+x[1]**2+x[2]**2

        %find the local minimum of f, starting at (1,1,1)
    print fmin(f,[1.0,1.0,1.0])
```

should print out

```
        [0.0,0.0,0.0]  %because (0,0,0) is the global minimum of f(x,y,z)=x^2+y^2+z^2
```

EDIT: To make this a little easier, I decided that it is acceptable for your implementation to require that fmin have additional arguments for the derivatives of f

Title: [7/27/2012] Challenge #82 [intermediate] (Broken Roman Numerals)
Text: Many ancient buildings and scriptures use [Roman numerals](http://en.wikipedia.org/wiki/Roman_numerals) to express numbers or years. Let's say you discover a bunch of formulas using these numerals, but some of the letters used in them are unreadable, like this:

```
    LX?I
```

That ? could be an I, V, or X, giving you these possibilities:

```
    LXII = 62
    LXVI = 66
    LXXI = 71
```

Write a function `guess_roman` that outputs all possibilities for an input string containing any number of question marks. For example, `guess_roman("X??I")` outputs:

```
XIII = 13
XVII = 17
XXII = 22
XXVI = 26
XXXI = 31
XLII = 42
XLVI = 46
XCII = 92
XCVI = 96
```

* What is the output of `guess_roman("D??I")`?
* How many unique seven-letter Roman numerals are there (that is, how many strings does `guess_roman("???????")` return?)
* What is their sum?

Title: [7/30/2012] Challenge #83 [intermediate] (Indexed file search)
Text: For this challenge, write two programs:

* 'index file1 file2 file3 ...' which creates an index of the words used in the given files (you can assume that they are plain text)
* 'search word1 word2 ...' which prints the name of every file in the index that contains all of the words given. This program should use the index previously built to find the files very quickly.

The speed of the "index" program doesn't matter much (i.e. you don't need to optimize it all that much), but "search" should finish very quickly, almost instantly. It should also scale very well, it shouldn't take longer to search an index of 10000 files compared to an index of 100 files. Google, after all, can handle the same task for billions/milliards* of documents, perhaps even trillions/billions!

*(***see easy problem for explanation)*

Index a folder where you have a lot of text files, which on my computer would probably mean the folders where I store the programs I've written. If you don't have a lot text files, head over to Project Gutenberg and go nuts.

Good luck!

Title: [8/1/2012] Challenge #84 [intermediate] (Recursive Song)
Text: Like many people who program, I got started doing this because I wanted to learn how to make video games.

As a result, my first ever 'project' was also my first video game. It involved a simple text adventure I called "The adventure of the barren moor"

Now that I'm an adult, I've decided to put some money into actually producing it as a real game (not really). I've hired a team of singers to sing the theme song.

The theme song is very simple: Its a rhyming ditty called "The barren moor" with a repeating recursive verses similar to the twelve days of christmas. We shamelessly ripped off the lyrics
of [The rattlin bog](http://en.wikipedia.org/wiki/The_Rattlin'_Bog) except instead of "Hi Ho the rattlin bog" we say "Hi Ho the barren moor", etc, and replace "moor" for "bog" everywhere else it's appropriate.
Also, instead of "A rare X, a rattlin' X" we have "A bare X, a barren X" in each verse.

Write a program that can print the full text the song "The barren moor".

Title: [8/3/2012] Challenge #85 [intermediate] (3D cuboid projection)
Text: Write a program that outputs simple 3D ASCII art for a cuboid in an [oblique](http://en.wikipedia.org/wiki/Oblique_projection) perspective, given a length, height, and depth, like this:

```
$ python 3d.py 20 10 3
   :::::::::::::::::::/
  :::::::::::::::::::/+
 :::::::::::::::::::/++
####################+++
####################+++
####################+++
####################+++
####################+++
####################+++
####################+++
####################++
####################+
####################
```

(The characters used for the faces (here `#`, `:`, and `+`) are fully up to you, but make sure you don't forget the `/` on the top-right edge.)

Title: [8/8/2012] Challenge #86 [intermediate] (Weekday calculations)
Text: Today's intermediate challenge comes from user [nagasgura](http://www.reddit.com/r/dailyprogrammer_ideas/comments/xx3cq/intermediate_calculate_the_day_of_the_week_on_any/)

>Calculate the day of the week on any date in history

>You could use the [Doomsday rule](http://en.wikipedia.org/wiki/Doomsday_rule) to program it. It should take in a day, month, and year as input, and return the day of the week for that date.

Title: [8/10/2012] Challenge #87 [intermediate] (Chord lookup)
Text: For this challenge, your task is to write a program that takes a musical chord name from input (like `Gm7`) and outputs the notes found in that chord (`G A# D F`). If you're no musician, don't worry -- the progress is quite simple. The first thing you need to know about is the 12 notes of the chromatic scale:

```
C C# D D# E F F# G G# A A# B
```

The intervals between two notes is expressed in semitones. For example, there are three semitones between the `D` and the `F` on this scale. Next, you'll need to know about the different kinds of chords themselves:

chord	symbol	tones
major	(nothing)	[0, 4, 7]
minor	m	[0, 3, 7]
dom. 7th	7	[0, 4, 7, 10]
minor 7th	m7	[0, 3, 7, 10]
major 7th	maj7	[0, 4, 7, 11]

To find out the notes in a chord, take the base note, then select the tones from the chromatic scale relative to the numbers in the list of tone intervals. For example, for `F7`, we look up the chord:

```
7 → dom. 7th → [0, 4, 7, 10]
```

Then we step `[0, 4, 7, 10]` semitones up from `F` in the scale, wrapping if necessary:

```
[F+0, F+4, F+7, F+10] → [F, A, C, D#]
```

Those are the notes in our chord.

If you know a thing or two about music theory: for extra credit, tweak your program so that it...

* outputs the chords "correctly", using `b` and `bb` and `x` where necessary

* supports more complex chords like `A9sus4` or `Emadd13`.

Title: [8/13/2012] Challenge #88 [intermediate] (Printing out a calendar)
Text: Make a program that given a certain month in a certain year, it prints out a calendar for that month in a nice calendar format.

For instance, for January 2012, it should print out something like:

```
+--------------------+
|      January       |
+--------------------+
|M |T |W |T |F |S |S |
+--------------------+
|  |  |  |  |  |  | 1|
| 2| 3| 4| 5| 6| 7| 8|
| 9|10|11|12|13|14|15|
|16|17|18|19|20|21|22|
|23|24|25|26|27|28|29|
|30|31|  |  |  |  |  |
+--------------------+
```

It doesn't have to look exactly like this, this is just an example. For instance, where I come from, the week on a calendar starts on Monday, but many other places it starts on Sunday - either way is fine. It also doesn't need all these fancy borders and stuff, you can just print out a row with the weekdays and under that the dates.

Note that this challenge is not about developing your own routines for handling dates, so you are perfectly allowed to use whatever date/time libraries you want. Most programming languages come with them built in. Of course, if you want to, you can use the results from [Challenge #86](http://www.reddit.com/r/dailyprogrammer/comments/xx97s/882012_challenge_86_intermediate_weekday/).

As a bonus, write the program so that it prints out the calendar for a whole year in a nice 3 by 4 grid. [Here's an example of how that might look](http://pastebin.com/41EnYKHq) (remember to check for leap years!). Again, the design is up to you.

Title: [8/20/2012] Challenge #89 [intermediate] (Printing strings in Brainf*)'**
Text: [A while ago](http://www.reddit.com/r/dailyprogrammer/comments/ti5ji/5112012_challenge_51_intermediate/) we had some fun with the very peculiar [Brainfuck programming language](http://en.wikipedia.org/wiki/Brainfuck), which (despite its limited set of commands and character set) is actually Turing complete, meaning that any computation you can do in any other programming language, you can do in Brainfuck.

That doesn't make it easy to use, though. Even as simple a task as printing out a string requires quite lengthy code. Today, we will simplify that task quite a bit!

Your task today is to write a program that takes a string as input and outputs Brainfuck code that, when run, will print out that string. That is, given "Hello World!", it will print out something that looks like [Wikipedia's example Hello World program](http://en.wikipedia.org/wiki/Brainfuck#Hello_World.21) (though not necessarily exactly, of course).

Use your program to create a Brainfuck program that prints out [The Raven](http://pastebin.com/v8AbQRFv), by Edgar Allen Poe.

Bonus: Try to optimize your program in such a way as to make the brainfuck code as short as possible. [Here, for instance, is a 34500 character long Brainfuck program that I made](http://pastebin.com/Jdmrf4hM) which prints out "The Raven". Can you beat me and write a program that generates shorter Brainfuck code?

Remember, this bonus is optional, even if your generated program is very big, you are still free to submit code.

Title: [8/22/2012] Challenge #90 [intermediate] (Scientific Units Calculator)
Text: In the SI system, measurements of scientific quantities are expressed in terms of 7 standard 'base' units for various quantities:

the "second" for time, the "meter" for length, "kilogram" for mass, the "ampere" for current, the "kelvin" for temperature, the "mole" for amount of substence, and the
"candela" for light intensity.

These base units and exponents of them fully describe any measurable quantity. For example, lets say we wanted to describe force. Force is defined as mass * acceleration.
accelleration is defined as velocity per second. velocity is defined as length per second. Therefore, force is mass*length per second per second, so force is defined as
m kg s^-1 s^-1 in SI units.

Write a program that can read in a units expression involving multiplying and dividing units and output the correct expression of those units in SI base units. Furthermore, you should make it so that your program ALSO accepts SI derived units as well, such as "watts" or "pascals" (there is a list of SI derived units and their base definitions [here](http://en.wikipedia.org/wiki/SI_derived_units)). If you can, you should also include some simple aliases that aren't even base units, such as 'mass' is 'kg' and 'velocity' is m/s.

Examples (input,output):

```
    m/s*m*cd -> s^-1 m^2 cd
    newton/m -> s^-2 kg
    watt/velocity -> s^-2 m kg
```

BONUS: Make it so, when printing, if there is a simpler name for the quanity output than the base name, then it also prints that as well. For example, s^-2 m kg is also
the definition of force in newtons, so when it prints watt/velocity it should output

```
    s^-2 m kg (Newtons)
```

SUPER BONUS: Correctly parse and handle Metrix Prefixes, like giga,micro,nano, etc. So we could have
kilo-watt/mega-joule -> kilo-second

Title: [8/24/2012] Challenge #91 [intermediate] (Cantor's fractions)
Text: Famous mathematician Georg Cantor once thought of a cool way to enumerate strictly positive rational numbers (i.e. x > 0): in an infinite 2D matrix of all rationals where A[x,y] = (x / y), start counting from A[1,1] and then zig-zag your way out like this:

http://www.homeschoolmath.net/teaching/rationals-countable.gif

Using this enumeration, write a program that outputs the first 1000 **distinct** strictly positive fractions as decimal numbers. (The repeated ones are crossed out in the above image, e.g. 2/2 = 1/1, so skip 2/2.)

Title: [8/27/2012] Challenge #92 [intermediate] (Rubik's cube simulator)
Text: Your intermediate task today is to build a simple simulator of a [Rubik's Cube](http://en.wikipedia.org/wiki/Rubik%27s_Cube). The cube should be represented by some sort of structure, which you can give a list of moves which it will execute and show you how the cube will look as a result.

A Rubik's Cube has six sides, which are traditionally known as Up, Down, Front, Back, Left and Right, abbreviated as U, D, F, B, L and R respectively. Color the sides of the cube as follows: Up <- white, Down <- yellow, Front <- green, Back <- blue, Left <- orange and Right <- red.

Taking advantage of the style of [problem #85](http://www.reddit.com/r/dailyprogrammer/comments/xq2ao/832012_challenge_85_intermediate_3d_cuboid/), the basic solved cube should then look something like this:

```
  W W W /
 W W W / R
W W W / R R
G G G|R R R
G G G|R R
G G G|R
```

Here showing the top side, the front side and the right side.

To list moves you can make on a Rubik's Cube, you use something called [Singmaster's notation](http://en.wikipedia.org/wiki/Rubik%27s_Cube#Move_notation), which works like this: to indicate a clockwise turn of any one side, you use the abbreviation of the side. So "R" means to spin the right side clockwise 90 degrees. If there's a prime sympol (i.e. a ') after the letter, that means to spin it counter-clockwise 90 degrees. If there's a "2" after the letter, it means you should spin that side 180 degrees. There is an extended notation for advanced moves (such as spinning a middle slice, or spinning two slices), but we'll ignore those for this challenge.

So, for instance, executed the sequence

```
R U B' R B R2 U' R' F R F'
```

on a totally solved cube, it should result in the following configuration:

```
  O O R /
 B W G / W
R R O / W R
W W G|W R R
G G G|R R
G G G|R
```

[See here for a step by step demonstration](http://alg.garron.us/?alg=R_U_B-_R_B_R2_U-_R-_F_R_F-).

Make a program that can execute a sequence of moves like these, and then print out what the cube looks like as a result, either in the cuboid form I've used here, or just print out the sides one by one.

What is the result of executing the following series of moves on a solved cube?

```
F' B L R' U' D F' B
```

Title: [8/30/2012] Challenge #93 [intermediate] (Z-Order Encryption)
Text: Write a program that implements the following encryption scheme:

It reads in some string of data of length N. Then, lay out that string in the smallest possible perfect power of two square that can fit the data.

For example, "My country, tis of thee" is 23 characters long. Therefore, it fits into a 5x5 square 25 characters long like this:

```
My co
untry
, tis
 of t
hee
```

However, when we constrain it to be a power of two, instead we end up with an 8x8 square, and laying it out looks like

```
My count
ry, tis
of thee
```

However, the encrytion part happens when, instead of laying out letters of the square from left to right as above, you lay out the square using a [Z-order code](http://en.wikipedia.org/wiki/Z-order_curve) instead, like so.

```
Myouofhe
 cnt te
ryti
, s
```

Write a program that reads a string from standard input and can encrypt to a z-order square, and vice-versa

Title: [9/01/2012] Challenge #94 [intermediate] (Base64 conversion)
Text: Create a function which accepts a byte array and outputs a [Base64](http://en.wikipedia.org/wiki/Base64) equivalent of the parameter. Write a second function that reverses the progress, decoding a Base64 string.

Obviously, you can't use a library function for the encode/decode.

Title: [9/03/2012] Challenge #95 [intermediate] (Filler text)
Text: Your intermediate task today is to write a function that can create "filler text", i.e. text that doesn't actually mean anything, but from a distance could plausibly look like a real language. This is very useful, for instance, if you're a designer and want to see what a design would look like with text in it, but you don't actually want to write the text yourself.

The rules are:

* The argument to function is the approx number of words.
* The text is made up of sentences with 3-8 words
* Each word is made up of 1-12 chars
* Sentences have first word capitalized and a period at the end
* After each sentence there is a 15% chance of a linebreak and an additional 50% chance of this line break being a paragraph break.

An example of what the text might look like [can be found here](http://pastebin.com/iLwDxghy).

***

Bonus: Make it so that the character frequency roughly matches the English language. I.e. more e's and t's than x's and z's. Also, modify your code so that it will insert commas, exclamation points, question marks and the occassional number (as a separate word, obviously).

Title: [9/06/2012] Challenge #96 [intermediate] (Parsing English Values)
Text: In intermediate problem #8 we did a number to english converter. Your task this time is to write a function that can take in a string like "One-Hundred and Ninety-Seven" or "Seven-Hundred and Forty-Four Million", parse it, and return the integer that it represents.

The definition of the exact input grammar is somewhat non-standard, so interpret it how you want and implement whatever grammar
you feel is reasonable for the problem. However, try to handle at least up to one-billion, non-inclusive. Of course, more is good too!

```
    parseenglishint("One-Thousand and Thirty-Four")->1034
```

Title: [9/08/2012] Challenge #97 [intermediate] (Sierpinski carpet)
Text: Write a function that accepts an integer *n* and returns a (3^*n* × 3^n) boolean matrix containing a *n*th-iteration [Sierpinski carpet](http://en.wikipedia.org/wiki/Sierpinski_carpet) fractal.

* How many 1 bits are there in `carpet(7)`?
* What is the largest value of `n` for which the matrix returned by `carpet(n)` fits in a terabyte?

For bonus points, write a general function `center_iter(d, n)` that generates fractals like the Sierpinski carpet in *d* dimensions. (`center_iter(1, n)` is the [Cantor set](http://en.wikipedia.org/wiki/Cantor_set), center_iter(2, n) the Sierpinski carpet, `center_iter(3, 1)` a 3x3x3 cube with the center piece removed, etc.)

Title: [9/15/2012] Challenge #98 [intermediate] (Multiple cycling)
Text: Write a function that takes two arguments: a limit, *lim*, and a list of integers, *x*. The function counts up from 0 by cycling through *x* and skipping numbers until we find the next number that's a multiple of *x[i]*. For example, when x is the list [5, 7, 3], start counting from 0:

1. Next multiple of 5 is 5
* Next multiple of 7 is 7
* Next multiple of 3 is 9
* Next multiple of 5 is 10
* Next multiple of 7 is 14
* Next multiple of 3 is 15

When the count reaches *lim* or a number above it, return the number of steps it took to reach it. (`multiple_cycle(15, [5, 7, 3])` would return 6.)

What is the result of `multiple_count(1000000000, [5395, 7168, 2367, 9999, 3])`?
Title: [9/17/2012] Challenge #99 [intermediate] (Unemployment map of the United States)
Text: [A little while ago](http://www.reddit.com/r/dailyprogrammer/comments/yj38u/8202012_challenge_89_difficult_coloring_the/) we took advantage of a [very useful blank map](http://en.wikipedia.org/wiki/File:Blank_US_Map.svg) hosted at Wikipedia. The advantage of this map is that it is very easy to assign different colors to each state (for details on how to do this, [see the previous problem](http://www.reddit.com/r/dailyprogrammer/comments/yj38u/8202012_challenge_89_difficult_coloring_the/)). We only had some silly fun with it, but it can also obviously be very useful in visualizing information about the country.

[Here is a text-file with unemployment data](https://gist.github.com/3740029) for all US states for each month from January 1980 to June 2012, stored in [CSV format](http://en.wikipedia.org/wiki/Comma-separated_values). The first column is the dates, then each column is the data for each state (the order of which is detailed in the headers). I got this information from the Federal Reserve Bank of St. Louis [FRED database](http://research.stlouisfed.org/fred2/), which has excellent API access (good work, St. Louis Fed!).

Using this table, make a program that can draw a map of unemployment across the United States at a given date. For instance, [here is a map of unemployment for July 2005](http://i.imgur.com/O4LP2.png). As you can see, I edited the map slightly, adding a scale to the left side and a header that includes the date. You can do that too if you wish, but it is not necessary in any way.

Your map doesn't need to look anything like mine. You can experiment with different colors and different styles. I selected the colors linearly based on unemployment, but you may want to use a different function to select colors, or perhaps color all states within a certain range the same (so that all states with 0%-2% are the same color, as are the states with 2%-4%, 4%-6%, etc). Experiment and see what you like.

Create a map which shows unemployment for February 1995.

Title: [9/20/2012] Challenge #100 [intermediate] ("Bad" Word Filter)

Text: Write a program called 'censor' that takes in one argument on the command line. This argument is the filename of a newline-separated wordlist of profanity such as

http://urbanoalvarez.es/blog/2008/04/04/bad-words-list/ or

http://www.bannedwordlist.com/SwearWordResources.htm

The program should then read a text from standard in, and print it out again, but replacing every instance of a word in the wordlist with a 'censored' version.
The censored version of a word has the same first character as the word, and the rest of the characters are replaced with '*'.

For example, the 'censored' version of 'peter' would be 'p****'

Example:

```
  >echo 'You jerkface!' | censor badwords.txt
  You j***face!
```

Title: [9/27/2012] Challenge #101 [intermediate] (Image Stenography)

Text: This challenge is loosely inspired by user [skeeto](/u/skeeto)

In this challenge, you are to implement any kind of [digital stenography you want](http://en.wikipedia.org/wiki/Steganography#Digital)
, but it has to be based on an image.

Write a program that takes in two command-line arguments, one of which is an input image and the other is a data file to hide in the image.
You can use the [netpbm](http://en.wikipedia.org/wiki/Netpbm_format) file format if you want for simplicity, but if your language has another format built-in, you can use that.

The point is that, whatever you choose to do, it has to be non-obvious upon casual inspection that the data file is embedded in the image.
If the data file is too big to store in the image given, your program can output an error.

For example, the algorithm I implemented for this challenge is very similar to the one on wikipedia: that is, for every 2 bits of data in the
data file, I replace the low-order two bits of a color channel of a pixel.

Implement this algorithm, or come up with your own!

Title: [9/30/2012] Challenge #102 [intermediate] (n-character-set strings)
Text: Write a function that takes a string *s* and an integer *n*, and returns whether or not the string *s* contains **at most** *n* different characters.

For example, `ncset("aacaabbabccc", 4)` would return `true`, because it contains only 3 different characters, 'a', 'b', and 'c', and 3 ≤ 4.

For how many English words (yes, it's time for [this dictionary](http://code.google.com/p/dotnetperls-controls/downloads/detail?name=enable1.txt) again!) does `ncset(word, 4)` hold?

Title: [10/13/2012] Challenge #103 [easy-difficult] (Text transformations)
Text: ###Easy
Back in the 90s (and early 00s) people thought it was a cool idea to \\/\\/|2][73 |_1|<3 7H15 to bypass text filters on BBSes. They called it [Leet (or 1337)](http://en.wikipedia.org/wiki/Leet), and it quickly became popular all over the internet. The habit has died out, but it's still quite interesting to see the various replacements people came up with when transforming characters.

Your job's to write a program that translates normal text into Leet, either by hardcoding a number of translations (e.g. A becomes either 4 or /-\\, randomly) or allowing the user to specify a random translation table as an input file, like this:

```
A   4 /-\
B   |3 [3 8
C   ( {
(etc.)
```

Each line in the table contains a single character, followed by whitespace, followed by a space-separated list of possible replacements. Characters should have some non-zero chance of not being replaced at all.

###Intermediate

Add a `--count` option to your program that counts the number of possible outcomes your program could output for a given input. Using the entire translation table from [Wikipedia](http://en.wikipedia.org/wiki/Leet#Orthography), how many possible results are there for `./leet --count "DAILYPROG"`? (Note that each character can also remain unchanged.)

Also, write a translation table to convert ASCII characters to hex codes (`20` to `7E`), i.e. `"DAILY" -> "4441494C59"`.

###Difficult

Add a `--decode` option to your program, that tries to reverse the process, again by picking any possibility randomly: `/\/\/` could decode to `M/`, or `NV`, or `A/V`, etc.

Extend the `--count` option to work with `--decode`: how many interpretations are there for a given input?

Title: [10/18/2012] Challenge #104 [Intermediate] (Bracket Racket)
Text: **Description:**

Write a function, where given a string of arbitrary characters, returns true if all brackets ([defined as parentheses, square-brackets, curly-braces, and chevrons](http://en.wikipedia.org/wiki/Bracket)) are correctly paired and ordered. This is to say that all brackets, if they enclose other brackets, enclose both the paired opening and closing characters.

Formal Inputs & Outputs:

Input Description:

string data - a given string that may or may not have correctly formed brackets.

Output Description:

Return True or False - true if the given string is correctly bracket formed.

Sample Inputs & Outputs:

"123", "(abc)", "()abc()", and "([<{abc123abc}>])" should return true, but "(abc[123)abc]" (wrong pairing) and "(abc>" (not closed) should return false.

Notes:

This is a very easy problem if you use a specific primitive data-structure.

Title: [10/20/2012] Challenge #105 [Intermediate] (Boolean logic calculator)
Text: [Boolean logic](http://en.wikipedia.org/wiki/Boolean_algebra) is something all programmers have to deal with, whether we like it or not. Why not automate the task to make it easier?

Your objective, if you choose to accept it, is to make a boolean logic calculator that can parse boolean logic statements. Given:

| = or
* = and
\^ = xor
! = not

Take input of 1s and 0s (or T and F) and output the evaluation of that statement. Try not to use statement evaluators built into your language of choice, like [eval](http://php.net/eval). Your parser should be able to evaluate statements in parentheses as well

Title: [10/23/2012] Challenge #106 [Intermediate] (Jugs)
Text: There exists a problem for which you must get exactly 4 gallons of liquid, using only a 3 gallon jug and a 5 gallon jug. You must fill, empty, and/or transfer liquid from either of the jugs to acquire exactly 4 gallons.

The solution to this particular problem is the following:

- Fill the 5 gallon jug

- Transfer from the 5 gallon jug to the 3 gallon jug, leaving 2 gallons in the 5 gallon jug and 3 gallons in the 3 gallon jug

- Empty the 3 gallon jug

- Transfer from the 5 gallon jug to the 3 gallon jug, leaving 0 gallons in the 5 gallon jug and 2 gallons in the 3 gallon jug

- Fill the 5 gallon jug

- Transfer from the 5 gallon jug to the 3 gallon jug, leaving 4 gallons in the 5 gallon jug and 3 gallons in the 3 gallon jug

- Empty the 3 gallon jug (optional)

The challenge will be to output a set of instructions to achieve an arbitrary final volume given 2 arbitrary sized gallon jugs. Jugs should be sized in a whole number (integer) amount of gallons. The program must also determine if the desired volume is impossible with this method (i.e. 2 - 4 gallon jugs to make 3 gallons). The program should be deterministic, meaning that running with the same inputs always produces the same solution (preventing a random pouring algorithm). The program should also consider outputting the most optimal set of instructions, but is not necessary.

Title: 10/25/2012] Challenge #107 [Intermediate] (Infinite Monkey Theorem)
Text: Verify the [Infinite Monkey Theorem](http://en.wikipedia.org/wiki/Infinite_monkey_theorem).

Well that's a bit hard, so let's go with this. Using any method of your choice, generate a random string of space-separated words. (The simplest method would be to randomly choose, with equal probability, one of the 27 characters including letters and space.) Filter the words using a [word list](http://code.google.com/p/dotnetperls-controls/downloads/detail?name=enable1.txt) of your choice, so that only words in the word list are actually output.

That's all you need for the basic challenge. For extra points, run your program for a few minutes and find the most interesting string of words you can get. The longer the better. For style, see if you can "train your monkey" by modifying either the random character generator or the word list to output text that's more Shakespearean in less time.

Title: [10/27/2012] Challenge #108 [Intermediate] (Minesweeper Generation)
Text: For the intermediate challenge, you will have to generate a Minesweeper game. Minesweeper boards have three attributes, length, width, and number of mines. Given the input below, output a correct gameboard.

Minesweeper games have two types of pieces, mines, and non-mines. The non-mines have a number, which is the number of mines adjacent to it.

For example: [Here's an image of a Minesweeper game.](http://upload.wikimedia.org/wikipedia/commons/7/7b/Minesweeper_end_Kmines.png)

Your input is...

* Height: 15
* Width: 15
* Mines: 20

Title: [10/30/2012] Challenge #109 [Intermediate]
Text: **Description:**

A palindrome is a string of characters that are read the same way both ways (forward and backwards). Given two range of integers (a_start, a_end and b_start, b_end), if at least one of the products between the two ranges is a palindrome, print the integer-pair.

For example, if the first range of integers is [90,99] and the second is [90,99], there is at least one palindrome because 91 x 99 = 9009, which is read the same forward and backward. Thus, "91, 99" should br printed.

Formal Inputs & Outputs:

Input Description:

Integer a_start - The starting range of the integer a

Integer a_end - The ending range of the integer a

Integer b_start - The starting range of the integer b

Integer b_end - The ending range of the integer b

Output Description:

Print an integer pair if their product is a palindrome.

Sample Inputs & Outputs:

Let a_start and a_end be 10, 11, and let b_start and b_end be 10, 11. Your code, given these arguments, should print "11, 11", since 11 * 11 is 121, and is a palindrome.

Notes:

This problem is of an easy-intermediate difficulty level; a brute-force solution works well enough, but think of what happens when given a large range of numbers. What is the computational complexity? What can you do to optimize palindrome verification?

Title: [11/3/2012] Challenge #110 [Intermediate] Creepy Crawlies
Text: **Description:**

The web is full of creepy stories, with Reddit's /r/nosleep at the top of this list. Since you're a huge fan of not sleeping (we are programmers, after all), you need to amass a collection of creepy stories into a single file for easy reading access! Your goal is to write a web-crawler that downloads all the text submissions from the top 100 posts on /r/nosleep and puts it into a simple text-file.

Formal Inputs & Outputs:

Input Description:

No formal input: the application should simply launch and download the top 100 posts from /r/nosleep into a special file format.

Output Description:
Your application must either save to a file, or print to standard output, the following format: each story should start with a title line. This line is three equal-signs, the posts's name, and then three more equal-signs. An example is "=== People are Scary! ===". The following lines are the story itself, written in regular plain text. No need to worry about formatting, HTML links, bullet points, etc.

Sample Inputs & Outputs:

If I were to run the application now, the following would be examples of output:

=== Can I use the bathroom? ===

Since tonight's Halloween, I couldn't... (your program should print the rest of the story, I omit that for example brevity)

=== She's a keeper. ===

I love this girl with all of my... (your program should print the rest of the story, I omit that for example brevity)

Title: [11/6/2012] Challenge #111 [Intermediate] The First Sudoku
Text: Find the lexicographically first valid sudoku. A valid sudoku consists of a 9x9 grid of the numbers 1-9 such that no number appears twice in any row or column, or in any of the 9 major 3x3 sub-grids. Two sudokus can be compared to determine which is lexicographically first like this: append the rows for each of the two sudokus together. Find the _first_ number where they differ. Whichever sudoku has a smaller number in that position is lexicographically first.

Here's the solution you should get:

`[1, 2, 3, 4, 5, 6, 7, 8, 9]`
`[4, 5, 6, 7, 8, 9, 1, 2, 3]`
`[7, 8, 9, 1, 2, 3, 4, 5, 6]`
`[2, 1, 4, 3, 6, 5, 8, 9, 7]`
`[3, 6, 5, 8, 9, 7, 2, 1, 4]`
`[8, 9, 7, 2, 1, 4, 3, 6, 5]`
`[5, 3, 1, 6, 4, 2, 9, 7, 8]`

`[6, 4, 2, 9, 7, 8, 5, 3, 1]`
`[9, 7, 8, 5, 3, 1, 6, 4, 2]`

If you want more of a challenge, find the lexicographically first valid 16x16 sudoku, or even larger.

Title: [11/14/2012] Challenge #112 [Intermediate]Date Sorting
Text: **Description:**

Your boss has just given you a list of dates that need to be sorted. Since it's Friday night, and you don't want to come in on Saturday, you want to write a little application that sorts these dates for you! Fortionatly, these dates are all written in a text file, where all of the date strings are on individual lines (un-sorted), and follow a standard format. All dates are guaranteed to be in the format of "YYYY MM DD hh:mm:ss", where "YYYY" is a year from 1800 to 2020, "MM" is from 01 to 12, "DD" is from 01 to 31, "hh" is from 00 to 23, "mm" is from 01 to 60, and "ss" is from 01 to 60.

Formal Inputs & Outputs:

Input Description:

String InputData - A string of the input file, where each date is on its own separate line, following the above date-time standard.

Output Description:

Your application must print all dates sorted, from the oldest date (i.e. 1800's) the modern dates (i.e. 2012), to future dates (i.e. 2020s").

Sample Inputs & Outputs:

The following is the string given to your function

2012 12 02 23:02:12
1899 03 02 14:04:42
1969 07 20 02:25:30
2019 11 02 00:13:01

The following is the output from the above input:

1899 03 02 14:04:42
1969 07 20 02:25:30
2012 12 02 23:02:12
2019 11 02 00:13:01

Notes:

Most solutions may either write a comparison function that keeps comparing elements from left-to-right until different values are found, or a date can be converted into a standard integer value and have that value used for comparison.

Title: [11/20/2012] Challenge #113 [Intermediate] Text Markup
Text: **Description:**

Many technologies, notably user-edited websites, take a source text with a special type of mark-up and output HTML code. As an example, Reddit uses a special formatting syntax to turn user texts into bulleted lists, web-links, quotes, etc.

Your goal is to write a function that specifically implements the Reddit markup language, and returns all results in appropriate HTML source-code. The actual HTML features you would like to implement formatting (i.e. using CSS bold vs. the old <b> tag) is left up to you, though "modern-and-correct" output is highly desired!

[Reddit's markup description is defined here](http://www.reddit.com/help/commenting). You are required to implement all 9 types found on that page's "Posting" reference table.

Formal Inputs & Outputs:

Input Description:

String UserText - The source text to be parsed, which may include multiple lines of text.

Output Description:

You must print the HTML formatted output.

Sample Inputs & Outputs:

The string literal `*Test*` should print <b>Test</b> or <div style="font-weight:bold;">Test</div>

Title: [12/4/2012] Challenge #114 [Intermediate] Shortest word ladder
Text: Given any two words from [this list of 3,807 four-letter words](http://pastebin.com/zY4Xt7iB), output a [word ladder](http://www.reddit.com/r/dailyprogrammer/comments/149kec/1242012_challenge_114_easy_word_ladder_steps/) between them that's as short as possible, using words from the list. (Note that the word list was chosen so that it's possible to form a ladder between any pair of words in the list.) Sample input:

```
    look
    leap
```

Sample output (any valid ladder of 5 words from `look` to `leap` would also work):

```
    look
    loon
    loan
    lean
    leap
```

__Bonus__: There are 8 pairs of words that require a ladder of 18 words to join. Find these 8 pairs of words. (Hint: a certain word appears in each of the 8 pairs.)

Title: [1/3/2013] Challenge #115 [Intermediate] Sum-Pairings
Text: # [](#IntermediateIcon) *(Intermediate)*: Sum-Parings

Let the term "sum-pair" be a pair of integers A and B such that the sum of A and B equals a given number C. As an example, let C be 10. Thus, the pairs (5, 5), (1, 9), (2, 8), etc. are all sum-pairs of 10.

Your goal is to write a program that, given an array through standard input (console), you echo out all sum-pairs of a given integer C.

# Formal Inputs & Outputs

## Input Description:

On the console, you will be first given an integer N. This is the number of following integers that are part of the array. After the N integers, you will be given an integer C which represents the sum-pair you are attempting to match.

## Output Description

Your program must print all unique pair of integers in the given list, where the sum of the pair is equal to integer C.

# Sample Inputs & Outputs

## Input (Through Console)

```
    4
    1 -3 4 10aw
    5
```

## Output (Through Console)

```
    1, 4
```

Note that there is only one pair printed to the console since there is only one unique pair (1, 4) that has the sum of C (5).

# Challenge Input

We will show the solution of this problem data-set in 7-days after the original submission.

```
    14
    10 -8 2 1 4 -9 6 1 9 -10 -5 2 3 7
    7
```

# Note

(*Awesome points awarded to /u/drigz for getting some important information into my thick-skull: there are linear-time solutions!*)

This is a common interviewing problem for programming jobs, so treat it as such! There is a very trivial solution, but the trivial solution will run in [O(N^2)](http://en.wikipedia.org/wiki/Big_O_notation) time. There are a few other known solutions: one that runs in O(N Log(N)) time (hint: takes advantage of sorting), and another in linear time (hint: dictionary).

Title: [01/09/13] Challenge #116 [Intermediate] Sort r/DailyProgrammer!

Text: # [](#IntermediateIcon) *(Intermediate)*: Sort r/DailyProgrammer!

Some users of r/DailyProgrammer want a list of URLs ordered from our very first challenge to the easiest challenge. Your goal is to crawl r/DailyProgrammer, automatically generate two types of these lists, and that's it!

Author: nint22

# Formal Inputs & Outputs

## Input Description

No formal input is required

## Output Description

You must print out two lists: one sorted by number, then category, and the other list sorted by category, then number. For each list, there should be N lines where N is the number of total challenges published. For each line, the challenge difficulty, ID, title, and URL must be placed in the following format:

[Easy / Medium / Hard] #<ID>: "<Title>" <URL>

To clarify on the two lists required, the first must be like the following:

```
    ...
    [Easy] #101: "Some Title" http://www.reddit.com/...
    [Intermediate] #101: "Some Title" http://www.reddit.com/...
    [Hard] #101: "Some Title" http://www.reddit.com/...
    ...
```

List two:

```
...
[Easy] #101: "Some Title" http://www.reddit.com/...
[Easy] #102: "Some Title" http://www.reddit.com/...
[Easy] #103: "Some Title" http://www.reddit.com/...
...
```

# Sample Inputs & Outputs
## Sample Input
None needed
## Sample Output
None needed
# Challenge Input
None needed
## Challenge Input Solution
None needed
# Note
Google around for the Reddit API documentation and related crawler libraries. It might save you quite a bit of low-level parsing!

Title: [01/16/13] Challenge #117 [Intermediate] Mayan Long Count
Text: # [](#IntermediateIcon) *(Intermediate)*: Mayan Long Count
The [Mayan Long Count](http://en.wikipedia.org/wiki/Maya_calendar#Long_Count) calendar is a counting of days with these units: "* The Maya name for a day was k'in. Twenty of these k'ins are known as a winal or uinal. Eighteen winals make one tun. Twenty tuns are known as a k'atun. Twenty k'atuns make a b'ak'tun.*". Essentially, we have this pattern:

 * 1 **kin** = 1 day

 * 1 **uinal** = 20 kin

 * 1 **tun** = 18 uinal

 * 1 **katun** = 20 tun

 * 1 **baktun** = 20 katun

The long count date format follows the number of each type, from longest-to-shortest time measurement, separated by dots. As an example, '12.17.16.7.5' means 12 baktun, 17 katun, 16 tun, 7 uinal, and 5 kin. This is also the date that corresponds to January 1st, 1970. Another example would be December 21st, 2012: '13.0.0.0.0'. This date is completely valid, though shown here as an example of a "roll-over" date.

Write a function that accepts a year, month, and day and returns the Mayan Long Count corresponding to that date. You must remember to take into account leap-year logic, but only have to convert dates after the 1st of January, 1970.

Author: skeeto
# Formal Inputs & Outputs
## Input Description
Through standard console, expect an integer N, then a new-line, followed by N lines which have three integers each: a day, month, and year. These integers are guaranteed to be valid days and either on or after the 1st of Jan. 1970.
## Output Description
For each given line, output a new line in the long-form Mayan calendar format: <Baktun>.<Katun>.<Tun>.<Uinal>.<Kin>.
# Sample Inputs & Outputs
## Sample Input
```
3
1 1 1970
20 7 1988
12 12 2012
```
## Sample Output

12.17.16.7.5
12.18.15.4.0
12.19.19.17.11
# Challenge Input
None needed
## Challenge Input Solution
None needed
# Note

* Bonus 1: Do it without using your language's calendar/date utility. (i.e. handle the leap-year calculation yourself).

* Bonus 2: Write the inverse function: convert back from a Mayan Long Count date. Use it to compute the corresponding date for `14.0.0.0.0`.

Title: [01/23/13] Challenge #118 [Intermediate] Canon Timing
Text:
[](#IntermediateIcon) *(Intermediate)*: Canon Timing
Naval ships typically separate their shells, explosives, and cannons in different compartments. This is all done for the safety of the ship and control of the explosive materials. [Check out this great animation from Wikipedia on how some ships load cannons!](http://en.wikipedia.org/wiki/File:Animated_gun_turret.gif)

Your job, as a first-class naval programmer, is to simulate how many shells can be fired given how long you want the system to work and how many seconds each sub-system does "work". Assume your system only has three components (shells, propellant, and the cannon), with each component having a different "work" times and with the cannon having a dependency on the shells and propellant loading.

The simulation should implement the following behaviors:

* Shell and propellant do work independently, that is to say they are not dependent on one another.

* Shell and propellant can only start re-working once they pass their materials to the cannon if, and only if, the canon is not firing.

* The cannon can only fire if both shell and propellant are given to it. This is to say there is no "queue" where the cannon is that can keep a small stock of shells and propellant; it must only have one of each while firing.

* Upon simulation initialization, all sub-systems must start their timers from 0. (i.e. the simulation starts in a default position of no work having been done)

* Note that when firing the cannon, you can count a "shot fired" immediately, but can only reload once the work-time has been consumed.

As an example, let shell and propellant retrieval take two seconds each. Let gun firing take one second. Your simulation will first take two seconds to get both the shell and propellant to the cannon. The cannon can then fire, consuming one second of work. During this one second of work, your shell and propellant retrievals can start, such that it only takes one more second for the cannon to wait before being able to fire again. Thus if you only simulated for
three seconds, your cannon would only fire once, but if you simulated for five seconds, it would fire twice, if simulated for seven seconds, it would fire thrice, etc.

Author: nint22
# Formal Inputs & Outputs
## Input Description
Expect four decimal numbers (up to two decimal points in precision, so a format like "<some integers or zero>.<two integer decimals, or double-zero>") on standard input (console) delimited by a space character. Let these four decimals, in order, be N A B and C. N is the time you want the firing system to be simulated. A and B are, respectively, the work times for shell and propellant retrieval. Finally, C is the time it takes to fire the cannon.
## Output Description

Simply print, as an integer, how many times the cannon can successfully fire in the given simulation time. Please note that a cannon's shot can be counted before the cannon's work time is past.

# Sample Inputs & Outputs

## Sample Input

```
5.00 2.00 2.00 1.00
```

## Sample Output

2

# Challenge Input

```
99.99 1.23 3.21 5.01
```

## Challenge Input Solution

Not yet posted (Check back in a few days)

# Note

This challenge is not as trivial as it appears, since simulating non-integer times will require internal abstraction of the time mechanism. Casting the input to floats or doubles will lead to errors when given large simulation times.

Title: [01/30/13] Challenge #119 [Intermediate] Find the shortest path

Text:

[](#IntermediateIcon) *(Intermediate)*: Find the shortest path

Given an ASCII grid through standard console input, you must find the shortest path from the start to the exit (without walking through any walls). You may only move up, down, left, and right; never diagonally.

Author: liloboy

# Formal Inputs & Outputs

## Input Description

The first line of input is an integer, which specifies the size of the grid in both dimensions. For example, a 5 would indicate a 5 x 5 grid. The grid then follows on the next line. A grid is simply a series of ASCII characters, in the given size. You start at the 'S' character (for Start) and have to walk to the 'E' character (for Exit), without walking through any walls (indicated by the 'W' character). Dots / periods indicate open, walk-able space.

## Output Description

The output should simply print "False" if the end could not possibly be reached or "True", followed by an integer. This integer indicates the shortest path to the exit.

# Sample Inputs & Outputs

## Sample Input

```
5
S....
WWWW.
.....
.WWWW
....E
```

Check out this link for many more examples! http://pastebin.com/QFmPzgaU

## Sample Output

```
True, 16
```

# Challenge Input

```
8
S...W...
.WW.W.W.
.W..W.W.
......W.
WWWWWWW.
E...W...
WW..WWW.
........
```

## Challenge Input Solution

True, 29

# Note

As a bonus, list all possible shortest paths, if there are multiple same-length paths.

Title: [02/06/13] Challenge #120 [Intermediate] Base Conversion Words
Text:
[](#IntermediateIcon) *(Intermediate)*: Base Conversion Words
Given as input an arbitrary string and base (integer), your goal is to convert the base-encoded string to all bases from 2 to 64 and try to detect all English-language words.

Author: aredna
# Formal Inputs & Outputs
## Input Description
On the console, you will be first given an arbitrary string followed by an integer "Base". This given string is base-encoded, so as an example if the string is "FF" and base is "16", then we know that the string is hex-encoded, where "FF" means 255 in decimal.
## Output Description
Given this string, you goal is to re-convert it to all bases, between 2 (binary) to 64. Based on these results, if any English-language words are found within the resulting encodings, print the encoded string, the encoding base, and on the same line have a comma-separated list of all words you found in it.

It is ** extremely** important to note this challenge's encoding scheme: unlike the "Base-64" encoding scheme, we will associate the value 0 (zero) as the character '0', up to value '9' (nine), the value 10 as the character 'a' up to 35 as the character 'z', the value 26 as 'A', then the value 61 as 'Z', and finally 62 as '+' (plus) and 63 as '/' (division). Essentially it is as follows:

Values 0 to 9 maps to '0' through '9'
Values 10 to 35 maps to 'a' through 'z'
Values 36 to 61 maps to 'A' through 'Z'
Value 62 maps to '+'
Value 63 maps to '/'
# Sample Inputs & Outputs
## Sample Input
E1F1 22
## Sample Output
Coming soon!
# Challenge Input
None given
## Challenge Input Solution
None given
# Note
None

Title: [03/06/13] Challenge #121 [Intermediate] Bytelandian Exchange 2
Text: # [](#IntermediateIcon) *(Intermediate)*: Bytelandian Exchange 2
This problem uses the same money-changing device from Monday's Easy challenge.

Bytelandian Currency is made of coins with integers on them. There is a coin for each non-negative integer (including 0). You have access to a peculiar money changing machine. If you insert a N-valued coin, it pays back 3 coins of the value N/2,N/3 and N/4, rounded down. For example, if you insert a 19-valued coin, you get three coins worth 9, 6, and 4. If you insert a 2-valued coin, you get three coins worth 1, 0, and 0.

This machine can potentially be used to make a profit. For instance, a 20-valued coin can be changed into three coins worth 10, 6, and 5, and 10+6+5 = 21. Through a series of exchanges, you're able to turn a 1000-valued coin into a set of coins with a total value of 1370.

Starting with a single N-valued coin, what's the maximum value you could get using this machine? Be able to handle very large N.

Author: Thomas1122
# Formal Inputs & Outputs
## Input Description

The value N of the coin you start with
## Output Description
The maximum total value of coins you can potentially exchange that coin for.
# Sample Inputs & Outputs
## Sample Input
1000
## Sample Output
1370
# Challenge Input
10000000000 (aka 10^10 aka 10 billion)
## Challenge Input Solution
???
# Note
Hint: use recursion!

Title: [03/13/13] Challenge #121 [Intermediate] Bytelandian Exchange 2
Text:
[](#IntermediateIcon) *(Intermediate)*: Bytelandian Exchange 2
This problem uses the same money-changing device from Monday's Easy challenge.

Bytelandian Currency is made of coins with integers on them. There is a coin for each non-negative integer (including 0). You have access to a peculiar money changing machine. If you insert a N-valued coin, it pays back 3 coins of the value N/2,N/3 and N/4, rounded down. For example, if you insert a 19-valued coin, you get three coins worth 9, 6, and 4. If you insert a 2-valued coin, you get three coins worth 1, 0, and 0.

This machine can potentially be used to make a profit. For instance, a 20-valued coin can be changed into three coins worth 10, 6, and 5, and 10+6+5 = 21. Through a series of exchanges, you're able to turn a 1000-valued coin into a set of coins with a total value of 1370.

Starting with a single N-valued coin, what's the maximum value you could get using this machine? Be able to handle very large N.

Author: Thomas1122
# Formal Inputs & Outputs
## Input Description
The value N of the coin you start with
## Output Description
The maximum total value of coins you can potentially exchange that coin for.
# Sample Inputs & Outputs
## Sample Input
1000
## Sample Output
1370
# Challenge Input
10000000000 (aka 10^10 aka 10 billion)
## Challenge Input Solution
???
# Note
Hint: use recursion!

Title: [03/27/13] Challenge #121 [Intermediate] Path to Philosophy
Text:
[](#IntermediateIcon) *(Intermediate)*: Path to Philosophy
Clicking on the first link in the main text of a Wikipedia article not in parentheses or italics, and then repeating the process for subsequent articles, usually eventually gets you to the Philosophy article. As of

May 26, 2011, 94.52% of all articles in Wikipedia lead eventually to the article Philosophy. The rest lead to an article with no wikilinks or with links to pages that do not exist, or get stuck in loops. [Here's a Youtube video demonstrating this phenomenon](http://www.youtube.c\
om/watch?v=vehDe2lSptU).

Your goal is to write a program that will find the path from a given article to the Philosophy article by following the first link (not in parentheses, italics or tables) in the main text of the given article. Make sure you have caching implemented from the start so you only need to fetch each page once.

You will then extend the program to do a depth-first search in search of the Philosophy article, backtracking if you get stuck and quitting only when you know there is no such path. The last thing you will do is generalise it to do a DFS towards *any* goal article.

Hint: Yes, there is a Wikipedia API. Feel free to use it.

The original formulation of this problem is found in the alternative text to [XKCD: Extended Mind](http://www.youtube.com/watch?v=vehDe2lSptU).

Author: nagasgura

# Formal Inputs & Outputs

## Input Description

Two strings, both which are names of existing Wikipedia articles (in the Wikipedia language of your choice).

## Output Description

A path of Wikipedia articles, each linked from the previous one, that leads from the start article to the end article.

* Links in parentheses, italics and tables should not be considered
* Links leading outside the main article namespace should not be considered
* Links are to be considered in the order they appear in an article
* The path should be created in a depth-first fashion
* You must implement article caching early on

You choose the output datastructure yourself, or print to standard-out.

# Sample Inputs & Outputs

## Sample Input

* From: Molecule
* To: Philosophy

## Sample Output

* Molecule
* Atom
* Matter
* Invariant mass
* Energy
* Kinetic energy
* Physics
* Natural philosophy
* Philosophy

# Challenge Input

* From: Asperger syndrome
* To: Logic

## Challenge Input Solution

* Asperger syndrome
* Autism spectrum
* Pervasive developmental disorder
* Mental disorder
* Psychology
* Applied science
* Natural science
* Science
* Knowledge
* Fact
* Proof (truth)
* Necessity and sufficiency
* Logic

Title: [04/01/13] Challenge #122 [Intermediate] User-Space Threading
Text: # [](#IntermediateIcon) *(Intermediate)*: User-Space Threading
This challenge is more coding-focused than maths-focused.

[Threading](http://en.wikipedia.org/wiki/Thread_(computing\)) is a computational model that allows the execution of small sections of instructions from different sources (i.e. threads of code), one after another, that it gives users a perception of code running in parallel. It is essentially a light-weight process that can be launched, managed, or terminated by the owning process.

Your goal is to implement an *efficient* and *dynamic* user-level threading library. You may implement this in any language and on any platform, but you may *not* use any existing threading code or implementation, such as the Win32 threading code or the UNIX pthreads lib. You may call system functions (such as interrupts and signals), but again cannot defer any thread-specific work to the operating system.

The term *efficient* means that when switching the thread of execution, you must do so as quickly as possible (big bonus points for actually measuring this). The term *dynamic* means that you provide a way (through either static variables, functions, config file, etc.) to allow end-users to change how fast you switch and what kind of algorithm you use for timing.

To help you get started, try to implement the following functions: (written in C-style for clarity)

* _ThreadID CreateThread(void (*threadFunction)(void*))_ // Returns a positive, non-zero, thread ID on success. Returns 0 on failure. Starts a thread of execution of the given thread function (for those confused, this is a C-style function being passed as an argument)
* _bool DestroyThread(ThreadID givenThreadId)_ // Destroys a thread of execution, based on the given thread ID

Title: [05/08/13] Challenge #123 [Intermediate] Synchronizing Calendars
Text:
[](#IntermediateIcon) *(Intermediate)*: Synchronizing Calendars
You're trying to plan out your family's Easter dinners for the next few centuries.

Your grandparents use the Lunar calendar, but your parents use the Julian calender, so you only have dinner with your grandparents when the calendars synchronize.

To help you figure that out, you're going to need to compute when M Julian years has the same amount of days as N Lunar months. As it turns out, these calendars synchronize with cycles of certain numbers of years.

Some information you will need:

* The time between full moons is 29.53059 days, so that is the length of one Lunar month.

* A Julian year is 365 days for three years, the fourth year is a leap year of 366 days, and then the cycle repeats.

* When taking the days in a number of Lunar months, you will likely get a decimal answer. _Round to the nearest day._

Author: Zamarok

# Formal Inputs & Outputs

## Input Description

You will be given two numbers `(M, N)`, where
`M` is the number of Julian years, and
`N` is the number of Lunar months.

You need to confirm that the number of days in `M` Julian years is equal to the number of days in `N` Lunar months.

## Output Description

You will take `M` and `N` and discover if the calendars synchronize after `M` Julian years and `N` Lunar months.

When looking at how many days `N` Lunar months will have, round to the nearest day.

If they do synchronize with the given input, print out the number of days that will pass before this occurs.

If the calendars don't synchronize with the given input, print `0`.

# Sample Inputs & Outputs

## Sample Input

```
   38, 470
```

## Sample Output

13879

# Challenge Input

```
   114, 2664
   30, 82
```

## Challenge Input Solution

```
   41638
   0
```

# Note

This was a problem in my homework for an astronomy class. I decided to code a solution to generate solutions, rather than figuring out it by hand. Turned out to be a good problem to solve, and I learned a bunch while doing it. It's difficult enough to provide a good challenge and to make you think about how to approach the problem from different angles.

Let me know if anyone wants to see the original homework assignment, or my solution (about 5 lines of Haskell).

# Extra Credit (optional):

Right now your program just confirms when the calendars will synchronize. You can modify your program to generate `(M, N)` to sequentially discover solutions. Find the largest solution for `M` where `M` is less than `500`.

For even more extra credit, point out the number of years that it takes for one cycle, a cycle being the time between when these calendars synchronize. There are multiple correct answers here.

Title: [05/08/13] Challenge #124 [Intermediate] Circular Graphs
Text: # [](#IntermediateIcon) *(Intermediate)*: Circular Graphs

A classic problem in computer science & [graph-theory](http://en.wikipedia.org/wiki/Graph_theory) is to detect if there are any [circular paths](http://en.wikipedia.org/wiki/Cycle_(graph_theory\)) in a given directed graph (sometimes called a cycle). Your goal is to write a program that takes in a series of edges, which defines a graph, and then print all sets of cycles onto a console or text file.

For the sake of clarity, we define a cycle as a set of vertices that have at least one incoming edge and one outgoing edge, where each node is only directly connected to at most two other nodes within the list.

Author: nint22

# Formal Inputs & Outputs
## Input Description

You will first be given an integer N, which represents the number of edges that will be given on each following new-line. Edges are defined as two integer numbers, where the direction of the edge always goes from the left vertex to the right vertex.

## Output Description

Simply print all vertices in a directed cycle; make sure that the cycle is closed (see sample output).

# Sample Inputs & Outputs
## Sample Input

```
4
1 2
2 3
3 1
3 4
```

## Sample Output

```
1 2 3 1
```

# Note

As usual with these kind of problems, the challenge isn't in writing a solution, but writing a *fast*-solution. If you post a solution, please discuss the big-O notation of your search function. Good luck, and have fun programming!

Title: [05/22/13] Challenge #125 [Intermediate] Halt! It's simulation time!
Text: # [](#IntermediateIcon) *(Intermediate)*: Halt! It's simulation time!

The [Halting Problem](http://en.wikipedia.org/wiki/Halting_problem), in computational theory, is the challenge of determining if a given program and data, when started, will actually finish. In more simple terms: it is essentially impossible to determine if an arbitrary program will ever complete because of how quickly a program's complexity can grow. One could attempt to partially solve the program by attempting to find logical errors, such as infinite loops or bad iteration conditions, but this cannot verify if complex structures ever halt. Another partial solution is to just simulate the code and see if it halts, though this fails for any program that becomes reasonably large. For this challenge, you will be doing this last approach:

Your goal is to simulate a given program, written in a subset of common assembly instructions listed below, and measure how many instructions were executed before the program halts, or assume the program never halts after executing 100,000 instructions. The fictional computer architecture that runs these instructions does so one instruction at a time, starting with the first and only stopping when the "HALT" instruction is executed or when there is no next instruction. The memory model is simple: it has 32 1-bit registers, indexed like an array. Memory can be treated conceptually like a C-style array named M: M[0], M[1], ..., M[31] are all valid locations. All memory should be initialized to 0. Certain instructions have arguments, which will always be integers between 0 to 31 (inclusive).

The instruction set only has 10 instructions, as follows:

Instruction	Description
AND a b	M[a] = M[a] bit-wise and M[b]
OR a b	M[a] = M[a] bit-wise or M[b]
XOR a b	M[a] = M[a] bit-wise xor M[b]
NOT a	M[a] = bit-wise not M[a]
MOV a b	M[a] = bit-wise M[b]
SET a c	M[a] = c
RANDOM a	M[a] = random value (0 or 1; equal probability distribution)
JMP x	Start executing instructions at index x
JZ x a	Start executing instructions at index x if M[a] == 0
HALT	Halts the program

Note that memory and code reside in different places! Basically you can modify memory, but cannot modify code.

Special thanks to the ACM collegiate programming challenges group for giving me the initial idea [here](https://icpcarchive.ecs.baylor.edu/index.php?option=com_onlinejudge&Itemid=8&category=12&page=show_problem&problem=945). Please note that one cannot actually solve the Halting problem, and that this is strictly a mini-simulation challenge.

# Formal Inputs & Outputs
## Input Description

You will first be given an integer N, which represents the number of instructions, one per line, that follows. Each of these lines will start with an instruction from the table above, with correctly formed arguments: the given program will be guaranteed to **never** crash, but are **not** guaranteed to ever halt (that's what we are testing!).

## Output Description

Simply run the program within your own simulation; if it halts (runs the HALT instruction) or ends (goes past the final instruction), write "Program halts!" and then the number of instructions executed. If the program does not halt or end within 100,000 instruction executions, stop the simulation and write "Unable to determine if application halts".

# Sample Inputs & Outputs
## Sample Input

```
    5
    SET 0 1
    JZ 4 0
    RANDOM 0
    JMP 1
    HALT
```

## Sample Output

```
    "Program halts! 5 instructions executed."
```

Title: [05/30/13] Challenge #126 [Intermediate] Perfect P'th Powers
Text: # [](#IntermediateIcon) *(Intermediate)*: Perfect P'th Powers

An integer X is a "perfect square power" if there is some integer Y such that Y^2 = X. An integer X is a "perfect cube power" if there is some integer Y such that Y^3 = X. We can extrapolate this where P is the power in question: an integer X is a "perfect p'th power" if there is some integer Y such that Y^P = X.

Your goal is to find the highest value of P for a given X such that for some unknown integer Y, Y^P should equal X. You can expect the given input integer X to be within the range of an unsigned 32-bit integer (0 to 4,294,967,295).

Special thanks to the ACM collegiate programming challenges group for giving me the initial idea [here](http://uva.onlinejudge.org/index.php?option=onlinejudge&page=show_problem&problem=1563).

# Formal Inputs & Outputs
## Input Description

You will be given a single integer on a single line of text through standard console input. This integer will range from 0 to 4,294,967,295 (the limits of a 32-bit unsigned integer).

## Output Description

You must print out to standard console the highest value P that fits the above problem description's requirements.

# Sample Inputs & Outputs
## Sample Input

Note: These are all considered separate input examples.

```
17

1073741824

25
```

## Sample Output

Note: The string following the result are notes to help with understanding the example; it is NOT expected of you to write this out.

```
1 (17^1)

30 (2^30)

2 (5^2)
```

Title: [06/09/13] Challenge #127 [Intermediate] Call Forwarding
Text: # [](#IntermediateIcon) *(Intermediate)*: Call Forwarding

A call forwarding service is a system that allows any incoming phone calls to a phone number be forwarded to a secondary phone number. This system is helpful in the case of a person taking a vacation (so that if Alice is out of the office, Bob receives all her customer's calls). It is possible, with such a system, that the secondary receiver (Bob in this case) also goes on vacation and also setups call forwarding to another person (Carol). Thus in such a situation, if someone calls Alice, it gets forwarded to Bob who in turn has the system re-forward to Carol.

Your job is to implement such a system, take in people's vacation times, and return how many call forwards are implemented at a given time and how "deep" the forwarding goes.

Special thanks to the ACM collegiate programming challenges group for giving me the initial idea [here](http://uva.onlinejudge.org/index.php?option=onlinejudge&Itemid=8&page=show_problem&problem=316). Also, based on recent world news, please consider donating to the [EFF](https://www.eff.org/) and make sure to write good code that protects your users. This subreddit is not the right place for a political discussion; I leave it up to the reader to think about why/how this subject may be important to you. At least consider that software you write in your "real-world job" may be used by an international audience, and such an audience may be targeted by unscrupulous people/governments. Protect people's lives by protecting their digital data: we programmers are the few people who can actually protect our users. </preachy paragraph>

# Formal Inputs & Outputs
## Input Description

You will be given an integer N on its own line that represents the number of vacation schedule descriptions that follow (each on their separate line). For each vacation description, you will be given four integers: the first is the person's regular 4-digit phone number, then the 4-digit phone number they choose to forward to, then when the vacation starts (measured in days) and how long the vacation lasts (measured in days). On the final line of input, which is line N + 1, you will be given a day to test the properties of the call-forwarding system (as defined in the output description).

Note that the date/time system here is based on a day index system. 1 represents the first day, 2 represents the second day, etc. Days do not respect the concept of months or years, so expect to simulate any given schedule up to the day 4,294,967,295. (32-bit unsigned integer max value)

Note that the input's forwarding chain will be guaranteed to *not* have circular forwarding: you can expect that Carol, in the challenge description, will never re-forward back to Alice while she is on vacation. As a secondary challenge, if you *can* detect such a failure, in that case simply print the chain in question that fails the call forwarding.

## Output Description

For the given day you want to test the system (the last integer from the input format), you must print both how many call forwarding are in place and the largest forwarding chain. A forwarding chain is the relationship as described in the challenge description where Alice forwards to Bob, who in turn forwards to Carol (this chain has a value of two, for the two call forwards).

# Sample Inputs & Outputs
## Sample Input

```
3
0000 0001 1 3
0001 4964 2 1
4964 0005 2 3
2
```

## Sample Output

```
3 call forwardings set up on day 2
3 call forwardings are the longest chain on day 2
```

Title: [06/12/13] Challenge #128 [Intermediate] Covering Potholes
Text: # [](#IntermediateIcon) *(Intermediate)*: Covering Potholes

Matrix city currently has very poor road conditions; full of potholes and are in dire need of repair. The city needs your help figuring out which streets (and avenues) they should repair. Chosen streets are repaired fully, no half measures, and are end-to-end. They're asking you to give them the minimum number of roads to fix such that all the potholes are still patched up. (They're on a very limited budget.)

Fortunately, the city was planned pretty well, resulting in a grid-like structure with its streets!

Original author: /u/yelnatz

# Formal Inputs & Outputs
## Input Description

You will be given an integer N on standard input, which represents the N by N matrix of integers to be given next. You will then be given N number of rows, where each row has N number of columns. Elements in a row will be space delimited. Any positive integer is considered a good road without problems, while a zero or negative integer is a road with a pot-hole.

## Output Description

For each row you want to repair, print "row X repaired", where X is the zero-indexed row value you are to repair. For each column you want to repair, print "column X repaired", where X is the zero-indexed column value you are to repair.

# Sample Inputs & Outputs
## Sample Input

```
5
0 4 0 2 2
1 4 0 5 3
2 0 0 0 1
2 4 0 5 2
2 0 0 4 0
```

## Sample Output

```
Row 0 repaired.
Row 2 repaired.
Row 4 repaired.
Column 2 repaired.
```

Based on this output, you can draw out (with the letter 'x') each street that is repaired, and validate that all pot-holes are covered:

```
x x x x x
1 4 x 5 3
x x x x x
2 4 x 5 2
x x x x x
```

I do not believe this is an NP-complete problem, so try to solve this without using "just" a brute-force method, as any decently large set of data will likely lock your application up if you do so.

Title: [07/10/13] Challenge #129 [Intermediate] N-Dimensional Vectors
Text: # [](#IntermediateIcon) *(Intermediate)*: N-Dimensional Vectors

N-Dimensional [vectors](http://en.wikipedia.org/wiki/Euclidean_vector) are vectors with n-components; it can be interpreted as a point in n-dimensional space. 2-dimensional (2D) vectors can be seen as a line on paper. 3D vectors can be seen as a line (direction with length) in regular space. You can represent higher n-dimensions in many different ways, but what we're interested in is the three common vector operations: length, normilization, and dot-product.

You are to implement code that first accepts a few vectors, the operations you want to perform on them, and their results.

Note: this Friday's upcoming *[Hard]* challenge will be to implement the cross-product computation (for only 3-dimensions). You are encouraged to bring the code you write for this solution as a starting point for the associated [Hard]-level challenge!

Original author: /u/nint22

# Formal Inputs & Outputs
## Input Description

You will be given an integer N on standard input, which represents the N-following number of lines of text. The start of each line will be a positive non-zero integer A, where A is the following number of space-delimited [Real number](http://en.wikipedia.org/wiki/Real_number) (floating-point in many languages). These numbers representing a vector of A-dimensions (or an A-component vector). After these N-lines of text, expect a single line with an integer M, which represents the M-following number of lines of text. Each line will start with the characters 'l', 'n', or 'd', representing the function you are to compute.

After that, you can expect one or two space-delimited integers. These integers represent the index of the above-defined vectors; the indexing scheme starts at zero (0). An 'l' and 'n' line will be given a single integer, while a 'd' will be given two space-delimited integers.

## Output Description

For each line that defines the function ('l' for length, 'n' for normalize, and 'd' for dot-product) and operands (the vector values based on the given indices), you are to print the result of the appropriate computation on standard console output. The length-function must compute the given vector's [Euclidean space](http://en.wikipedia.org/wiki/Euclidean_space#Distance) length. The normalize-function must compute the given vector's [Unit vector](http://en.wikipedia.org/wiki/Unit_vector). Finally, the Dot-product function must compute the *two* given vector's, well... [Dot Product](http://en.wikipedia.org/wiki/Dot_product)! When printing your result, you may choose however you print the result (regular float, or scientific notation), but you must be accurate with 5 decimals.

# Sample Inputs & Outputs
## Sample Input

```
    5
    2 1 1
    2 1.2 3.4
    3 6.78269 6.72 6.76312
    4 0 1 0 1
    7 84.82 121.00 467.05 142.14 592.55 971.79 795.33
    7
    l 0
    l 3
    l 4
    n 1
    n 2
    n 3
    d 0 1
```

## Sample Output

```
    1.4142
    1.4142
    1479.26
    0.33282 0.94299
    0.579689 0.574332 0.578017
    0 0.707107 0 0.707107
    4.6
```

Title: [07/17/13] Challenge #130 [Intermediate] Foot-Traffic Generator
Text: # [](#IntermediateIcon) *(Intermediate)*: Foot-Traffic Generator

This week's [Easy] challenge was [#133: Foot-Traffic Analysis](http://www.reddit.com/r/dailyprogrammer/comments/1iambu/071513_challenge_133_easy_foottraffic_analysis/): part of the challenge was to parse foot-traffic information and print out some room-usage information. What if we wanted to test this program with our own custom data-set? How can we generate a custom log to test our Foot-Traffic Analysis tool with? Real-world programming requires you to often write your own test-generating code! Your goal in this challenge is to do exactly that: write a foot-traffic generator!

Read up on the original [[Easy] challenge here](http://www.reddit.com/r/dailyprogrammer/comments/1iambu/071513_challenge_133_easy_foottraffic_analysis/), and take a look at the input-data format as well as the important data-consistency rules. It's very important to understand the previous challenge's input format, as your output here will have to match it!

Original author: /u/nint22

Note: This is not a particularly difficult challenge, but it is a great example of real-world programming! Make sure you actually test your generator with the previous challenge's program you wrote.

# Formal Inputs & Outputs
## Input Description

On standard console input, you will be given one line of five space-delimited integers: E (for the number of events to generate), V (for the number of visitors), R (for the number of rooms), I (for the time at which the earliest event can occur), and O (for the time at which the last event can occur).

## Output Description

You must output a data-set that follows the input format for the Foot-Traffic Analysis challenge. You must print off E x2 lines (the x2 is because one "event" is defined as both a visitor going into a room and then eventually out of it), only referring to user indices 0 to V (inclusive) and room indices 0 to R (inclusive). Make sure that the time for any and all events are within the range of I and O (inclusive)! Remember that time is defined as the minute during the day, which will always be between 0 and 24H x 60 minutes (1440).

Though your data set can randomly pick the visitor and room index, you must make sure it is logically valid: users that enter a room eventually have to leave it, users cannot enter a room while being in another room, and must always enter a room first before leaving it. Note that we do not enforce the usage of all visitor or room indices: it is possible that with a small E but a large R and V, you only use a small subset of the room and visitor indices.

Make sure to [seed your random-number generator](http://en.wikipedia.org/wiki/Random_seed)! It does not matter if your resulting list is ordered (on any column) or not.

# Sample Inputs & Outputs
## Sample Input

```
18 8 32 300 550
```

## Sample Output

```
36
0 11 I 347
1 13 I 307
2 15 I 334
3 6 I 334
4 9 I 334
5 2 I 334
6 2 I 334
7 11 I 334
8 1 I 334
0 11 O 376
1 13 O 321
2 15 O 389
3 6 O 412
4 9 O 418
5 2 O 414
6 2 O 349
7 11 O 418
8 1 O 418
0 12 I 437
1 28 I 343
2 32 I 408
```

```
   3 15 I 458
   4 18 I 424
   5 26 I 442
   6 7 I 435
   7 19 I 456
   8 19 I 450
   0 12 O 455
   1 28 O 374
   2 32 O 495
   3 15 O 462
   4 18 O 500
   5 26 O 479
   6 7 O 493
   7 19 O 471
   8 19 O 458
```

Title: [08/08/13] Challenge #131 [Intermediate] Simple Ray-Casting
Text: # [](#IntermediateIcon) *(Intermediate)*: Simple Ray-Casting

[Ray Casting](http://en.wikipedia.org/wiki/Ray_casting) is a method of [rendering 3D computer graphics](http://en.wikipedia.org/wiki/Rendering_(computer_graphics\)), popular in the early/mid 90's. Famous games like [Wolfenstein](http://en.wikipedia.org/wiki/Wolfenstein_3D) and [Doom](http://en.wikipedia.org/wiki/Doom_(video_game\)) are great examples of ray-casting based graphics. Real-time computer graphics today are based on hardware-accelerated polygon [rasterization](http://en.wikipedia.org/wiki/Rasterisation), while film-quality computer graphics are based on [ray-tracing](http://en.wikipedia.org/wiki/Ray_tracing_(graphics\)) (a more advanced and finer-detailed ray-casting derivative).

Your goal is to implement a single ray-cast query within a 2D world: you will be given the ray's origin and direction, as well as a top-down view of a tile-based world, and must return the position of the first wall you hit. The world will be made of a grid of tiles that are either occupied (as defined by the 'X' character), or empty (as defined by the space ' ' character). **[Check out these graphics as a visualization of example 1; it should help clarify the input data](http://imgur.com/a/pUOKb).** Real ray-casting applications do many of these wall-collision hits, generally one per column of pixels you want to render, but today you only have to solve for a single ray!

Original author: /u/nint22

# Formal Inputs & Outputs
## Input Description

On standard console input you will be given two integers, N and M. N is the number of columns, while M is the number of rows. This will be followed by M rows of N-characters, which are either 'x' or ' ' (space), where 'x' is a wall that you can collide with or ' ' which is empty space. After this world-definition data, you will be given three space-delimited floating-point values: X, Y, and R. X and Y are world positions, following [this coordinate system description](http://imgur.com/a/pUOKb), with R being a radian-value degree representing your ray direction (using the unit-circle definition where if R is zero, it points to the right, with positive R growth rotation counter-clockwise). R is essentially how much you rotate the ray from the default position of X+ in a counter-clockwise manner.

## Output Description

Simply print the collision coordinate with three-digit precision.

# Sample Inputs & Outputs
## Sample Input

Note that this input is rendered and explained in more detail [here](http://imgur.com/a/pUOKb).

```
10 10
xxxxxxxxxx
x  x x   x
x  x x   x
x    x xxx
xxxx     x
x  x     x
x        x
x  x     x
x  x    xx
xxxxxxxxxx
6.5 6.5 1.571
```

## Sample Output

```
6.500 1.000
```

Title: [08/08/13] Challenge #132 [Intermediate] Tiny Assembler
Text: # [](#IntermediateIcon) *(Intermediate)*: Tiny Assembler

Tiny, a very simple fictional computer architecture, is programmed by an assembly language that has 16 [mnemonics](http://en.wikipedia.org/wiki/Assembly_language#Opcode_mnemonics_and_extended_mnemonics), with 37 unique op-codes. The system is based on [Harvard architecture](http://en.wikipedia.org/wiki/Harvard_architecture), and is very straight-forward: program memory is different from working memory, the machine only executes one instruction at a time, memory is an array of bytes from index 0 to index 255 (inclusive), and doesn't have any relative addressing modes. Instructions are multibyte, much like the X86 architecture. Simple instructions like HALT only take one byte, while complex instructions like JLS (Jump if Less-than) take four bytes.

Your goal will be to write an [assembler](http://en.wikipedia.org/wiki/Assembler_(computing\)#Assembler) for Tiny: though you don't need to simulate the code or machine components, you must take given assembly-language source code and produce a list of hex op-codes. You are essentially writing code that converts the lowest human-readable language to machine-readable language!

The following are all mnemonics and associated op-codes for the Tiny machine. Note that brackets mean "content of address-index" while non-brackets mean literals. For example, the instruction "AND [0] 1" will set the contents of the first element (at index 0) of memory to 1 if, and only if, the original contents at that element are equal to the given literal 1.

Google Documents of the below [found here](https://docs.google.com/document/d/1lDk_1dLz82iwc-1hESNFSepcz4Swyaf9P1j5hvR2iHw).

Group	Instruction	Byte Code	Description
1. Logic	AND a b	2 Ops, 3 bytes:	M[a] = M[a] bit-wise and M[b]
		0x00 [a] [b]	
		0x01 [a] b	
	OR a b	2 Ops, 3 bytes:	M[a] = M[a] bit-wise or M[b]
		0x02 [a] [b]	
		0x03 [a] b	
	XOR a b	2 Ops, 3 bytes:	M[a] = M[a] bit-wise xor M[b]
		0x04 [a] [b]	
		0x05 [a] b	
	NOT a	1 Op, 2 bytes:	M[a] = bit-wise not M[a]
		0x06 [a]	
2. Memory	MOV a b	2 Ops, 3 bytes:	M[a] = M[b], or the literal-set M[a] = b
		0x07 [a] [b]	
		0x08 [a] b	

```
|	3. Math	|	RANDOM a	|	2 Ops, 2 bytes:	|	M[a] = random value (0 to 25; equal probability
distribution)	|
|		|		|	0x09 [a]	|		|
|		|	ADD a b	|	2 Ops, 3 bytes:	|	M[a] = M[a] + b; no overflow support	|
|		|		|	0x0a [a] [b]	|		|
|		|		|	0x0b [a] b	|		|
|		|	SUB a b	|	2 Ops, 3 bytes:	|	M[a] = M[a] - b; no underflow support	|
|		|		|	0x0c [a] [b]	|		|
|		|		|	0x0d [a] b	|		|
|	4. Control	|	JMP x	|	2 Ops, 2 bytes:	|	Start executing instructions at index of value M[a]
(So given a is zero, and M[0] is 10, we then execute instruction 10) or the literal a-value	|
|		|		|	0x0e [x]	|		|
|		|		|	0x0f x	|		|
|		|	JZ x a	|	4 Ops, 3 bytes:	|	Start executing instructions at index x if M[a] == 0 (This is a
nice short-hand version of )	|
|		|		|	0x10 [x] [a]	|		|
|		|		|	0x11 [x] a	|		|
|		|		|	0x12 x [a]	|		|
|		|		|	0x13 x a	|		|
|		|	JEQ x a b|	4 Ops, 4 bytes:	|	Jump to x or M[x] if M[a] is equal to M[b] or if M[a] is equal to
the literal b.	|
|		|		|	0x14 [x] [a] [b]	|		|
|		|		|	0x15 x [a] [b]	|		|
|		|		|	0x16 [x] [a] b	|		|
|		|		|	0x17 x [a] b	|		|
|		|	JLS x a b	|	4 Ops, 4 bytes:	|	Jump to x or M[x] if M[a] is less than M[b] or if M[a] is less
than the literal b. |
|		|		|	0x18 [x] [a] [b]	|		|
|		|		|	0x19 x [a] [b]	|		|
|		|		|	0x1a [x] [a] b	|		|
|		|		|	0x1b x [a] b	|		|
|		|	JGT x a b|	4 Ops, 4 bytes:	|	Jump to x or M[x] if M[a] is greater than M[b] or if M[a] is
greater than the literal b.	|
|		|		|	0x1c [x] [a] [b]	|		|
|		|		|	0x1d x [a] [b]	|		|
|		|		|	0x1e [x] [a] b	|		|
|		|		|	0x1f x [a] b	|		|
|		|	HALT	|	1 Op, 1 byte:	|	Halts the program / freeze flow of execution	|
|		|		|	0xff	|		|
|	5. Utilities	|	APRINT a	|	4 Ops, 2 byte:	|	Print the contents of M[a] in either ASCII (if
using APRINT) or as decimal (if using DPRINT). Memory ref or literals are supported in both instructions.	|
|		|	DPRINT a	|	0x20 [a] (as ASCII; aprint)	|		|
|		|		|	0x21 a (as ASCII)	|		|
|		|		|	0x22 [a] (as Decimal; dprint)	|		|
|		|		|	0x23 a (as Decimal)	|		|
```

Original author: /u/nint22

# Formal Inputs & Outputs
## Input Description

You will be given the contents of a file of Tiny assembly-language source code. This text file will only contain source-code, and no meta-data or comments. The source code is not case-sensitive, so the instruction "and", "And", and "AND" are all the same.

## Output Description

Print the resulting op-codes in hexadecimal value. Formatting does not matter, as long as you print the *correct* hex-code!

# Sample Inputs & Outputs
## Sample Input

The following Tiny assembly-language code will multiply the numbers at memory-location 0 and 1, putting the result at memory-location 0, while using [2] and [3] as working variables. All of this is done at the lowest 4 bytes of memory.

```
    Mov [2] 0
    Mov [3] 0
    Jeq 6 [3] [1]
    Add [3] 1
    Add [2] [0]
    Jmp 2
    Mov [0] [2]
    Halt
```

## Sample Output

```
    0x08 0x02 0x00
    0x08 0x03 0x00
    0x15 0x06 0x03 0x01
    0x0B 0x03 0x01
    0x0A 0x02 0x00
    0x0F 0x02
    0x07 0x00 0x02
    0xFF
```

# Challenge Bonus

If you write an interesting Tiny-language program and successfully run it against your assembler, you'll win a silver medal! If you can formally prove (it won't take much effort) that this language / machine is Turing Complete, you'll win a gold medal!

Title: [09/11/13] Challenge #133 [Intermediate] Chain Reaction
Text: # [](#IntermediateIcon) *(Intermediate)*: Chain Reaction

You are a physicists attempting to simulate a discrete two-dimensional grid of elements that cause chain-reactions with other elements. A chain-reaction is when an element at a position becomes "active" and spreads out and activates with other elements. Different elements have different propagation rules: some only can react with directly-adjacent elements, while others only reacting with elements in the same column. Your goal is to simulate the given grid of elements and show the grid at each interaction.

Original author: /u/nint22

# Formal Inputs & Outputs
## Input Description

On standard console input, you will be given two space-delimited integers N and M, where N is the number of element types, and M is the grid size in both dimensions. N will range inclusively between 1 and 20, while M ranges inclusively from 2 to 10. This line will then be followed by N element definitions.

An element definition has several space-delimited integers and a string in the form of "X Y R D". X and Y is the location of the element. The grid's origin is the top-left, which is position (0,0), where X grows positive to the right and Y grows positive down. The next integer R is the radius, or number of tiles this element propagates outwardly from. As an example, if R is 1, then the element can only interact with directly-adjacent elements. The string D at the end of each line is the "propagation directions" string, which is formed from the set of characters 'u', 'd', 'l', 'r'. These represent up, down, left, right, respectively. As an example, if the string is "ud" then the element can only propagate R-number of tiles in the up/down directions. Note that this string can have the characters in any order and should not be case-sensitive. This means "ud" is the same as "du" and "DU".

Only the first element in the list is "activated" at first; all other elements are idle (i.e. do not propagate) until their positions have been activated by another element, thus causing a chain-reaction.

## Output Description

For each simulation step (where multiple reactions can occur), print an M-by-M grid where elements that have had a reaction should be filled with the 'X' character, while the rest can be left blank with the space character. Elements not yet activated should always be printed with upper-case letters, starting with the letter 'A', following the given list's index. This means that the first element is 'A', while the second is 'B', third is 'C', etc. Note that some elements may not of have had a reaction, and thus your final simulation may still contain letters.

Stop printing any output when no more elements can be updated.

# Sample Inputs & Outputs
## Sample Input

```
4 5
0 0 5 udlr
4 0 5 ud
4 2 2 lr
2 3 3 udlr
```

## Sample Output

```
Step 0:
A   B

   C
  D

Step 1:
X   B

   C
  D

Step 2:
X   X

   C
  D

Step 3:
X   X

   X
  D
```

# Challenge Bonus

1: Try to write a visualization tool for the output, so that users can actually see the lines of propagation over time.

2: Extend the system to work in three-dimensions.

Title: [11/12/13] Challenge #135 [Intermediate] De Morgan's Law
Text: # [](#IntermediateIcon) *(Intermediate)*: De Morgan's Law

[De Morgan's Law](http://en.wikipedia.org/wiki/De_Morgan's_laws) is a neat law of Boolean propositional logic, helpful in every-day programming. It helps with inverting a boolean expression to get the opposite expression. [Wolfram Mathworld](http://mathworld.wolfram.com/deMorgansLaws.html) has a much more correct definition using set theory, being equivalent to [Boolean expressions](http://en.wikipedia.org/wiki/Boolean_algebra).

That's important since programming is all about Boolean-expressions! If you want to invert some "if" conditional code (that is, to compute the test for the "else" fall-through case), you need to apply the Law's two transformations:

 * The not (of expression-A and expression-B) is-equivalent-to not expression-A or not expression-B
 * The not (of expression-A or expression-B) is-equivalent-to not expression-A and not expression-B

An interpretation of these rules, helpful for applying it, is that you first apply the *not*-operator on all expressions, then replace all *and*s with *or*s, and vice-versa. This gets much more tricky and complex when order of operations come in play with parentheses (nested expressions).

Your goal is to take a C-like language's Boolean expression, and apply De Morgan's Law on it. You may choose to simplify the resulting expression as much as possible for epic bonus poinst; consider reading into [Karnaugh maps](http://en.wikipedia.org/wiki/Karnaugh_map) as one approach. "Simplified" is measured in the least-amount of variables and operators required (not counting parentheses).

# Formal Inputs & Outputs
## Input Description

The grammar of this C-like language is English-language space-delimited words, uses parentheses for nested expressions, with variables strictly being lower-case alpha-numeric. The reserved key-word for logical-and is "AND", with logical-or "OR", and logical-not "NOT". The expression will be in [Infix-notation](http://en.wikipedia.org/wiki/Infix_notation).

Note that the "NOT" is a unary operator, meaning it always applies first to variables on the right of itself, or to a full expression. The expression "NOT a OR b" is equivalent to "(NOT a) OR b", but not "NOT (a or b)".

## Output Description

Given the expression, print it's inverse using the same grammar and given variables. Simplify the expression for epic bonus points.

# Sample Inputs & Outputs
## Sample Inputs

```
    a
    NOT a
    a AND b
    NOT a AND b
    NOT (a AND b)
    NOT (a OR b AND C) OR NOT(a AND NOT b)
```

## Sample Outputs

```
    NOT a
    a
    NOT a or NOT b
    a OR NOT b
    a AND b
    a AND NOT b
```

Title: [11/20/13] Challenge #136 [Intermediate] Ranked Voting System

Text: # [](#IntermediateIcon) *(Intermediate)*: Ranked Voting System

A [Ranked Voting System](http://en.wikipedia.org/wiki/Ranked_voting_systems) is a system that chooses a result based on a ranked-preference rather than a simple majority. A standard ranked ballot generally has multiple choices, only one of which one can be picked. A ranked ballot allows you to choose the order in which you prefer candidates. An example could be that you prefer choice **B** first, then choice **C**, and finally choice **A**.

There are some neat implications on how this differs from conventional voting systems, and is used in many different countries and states (check out the same [article's list of current uses](http://en.wikipedia.org/wiki/Ranked_voting_systems#Use_by_polities\). [CGP Grey has a great explanation](http://www.youtube.com/watch?v=3Y3jE3B8HsE) on the overall system; well worth a watch! The overall difference between the two system is that a more agreed-upon candidate could win during a heavily split election.

Your goal is to take a list of candidates and voter's ballots, implement this voting system (using the [Instant-runoff rules](http://en.wikipedia.org/wiki/Instant-runoff_voting)), and print the results of the fictional election.

# Formal Inputs & Outputs
## Input Description

On standard console input, you will be given two space-delimited integers, N and M. N is the number of votes, while M is the number of candidates. After this line, you will be given the candidates line, which is a space-delimited set of M-number of candidate names. These names are one-word lower-case letters only. This is followed by N-lines of ballots, where each ballot is a list of M-integers, from 0 to M-1, representing the order of preference.

Note that the order of preference for ballots goes from left-to-right. The integers are the index into the candidate list. For the example below, you can map 0: Knuth, 1: Turing, 2: Church. This means that if the ballot row is "1 0 2", that means the voter prefers Turing over Knuth over Church.

## Output Description

Given the candidates and ballots, compute the first-round of successful candidates (e.g. rank them based on all ballot's first choice). If the percentage of votes for any one candidate is more than 50%, print the candidate name as the winner. Else, take all the votes of the least-successful candidate, and use their ballot's 2nd choice, summing again the total votes. If needed (e.g. there is no candidate that has more than 50% of the votes), repeat this process for the 3rd, 4th, etc. choice, and print the winner of the election.

For each round of computation, print the percentage of votes for each candidate, and rank them based on that percentage, using the output format.

# Sample Inputs & Outputs
## Sample Inputs

```
5 3
Knuth Turing Church
1 0 2
0 1 2
2 1 0
2 1 0
1 2 0
```

## Sample Outputs

```
Round 1: 40.0% Turing, 40.0% Church, 20.0% Knuth
Round 2: 60.0% Turing, 40.0% Church
Turing is the winner
```

Title: [11/28/13] Challenge #137 [Intermediate / Hard] Banquet Planning
Text: # [](#IntermediateIcon) *(Intermediate)*: Banquet Planning

You and your friends are planning a big banquet, but need to figure out the order in which food will be served. Some food, like a turkey, have to be served after appetizers, but before desserts. Other foods are more simple, like a pecan pie, which can be eaten any time after the main meal. Given a list of foods and the order-relationships they have, print the banquet schedule. If a given food item cannot be placed in this schedule, write an error message for it.

# Formal Inputs & Outputs
## Input Description

On standard console input, you will be given two space-delimited integers, N and M. N is the number of food items, while M is the number of food-relationships. Food-items are unique single-word lower-case names with optional underscores (the '_' character), while food-relationships are two food items that are space delimited. All food-items will be listed first on their own lines, then all food-relationships will be listed on their own lines afterwards. A food-relationship is where the first item must be served before the second item.

Note that in the food-relationships list, some food-item names can use the [wildcard-character](http://en.wikipedia.org/wiki/Wildcard_character) '*'. You must support this by expanding the rule to fulfill any combination of strings that fit the wildcard. For example, using the items from Sample Input 2, the rule "turkey* *_pie" expands to the following four rules:

```
turkey almond_pie
turkey_stuffing almond_pie
turkey pecan_pie
turkey_stuffing pecan_pie
```

A helpful way to think about the wildcard expansion is to use the phrase "any item A must be before any item B". An example would be the food-relationship "*pie coffee", which can be read as "any pie must be before coffee".

Some orderings may be ambiguous: you might have two desserts before coffee, but the ordering of desserts may not be explicit. In such a case, group the items together.

## Output Description

Print the correct order of food-items with a preceding index, starting from 1. If there are ambiguous ordering for items, list them together on the same line as a comma-delimited array of food-items. Any items that do not have a relationship must be printed with a warning or error message.

# Sample Inputs & Outputs
## Sample Input 1

```
3 3
salad
turkey
dessert
salad dessert
turkey dessert
salad turkey
```

## Sample Output 1

```
1. salad
2. turkey
3. dessert
```

## Sample Input 2

```
8 5
turkey
pecan_pie
salad
crab_cakes
almond_pie
rice
coffee
turkey_stuffing
turkey_stuffing turkey
turkey* *_pie
*pie coffee
salad turkey*
crab_cakes salad
```

## Sample Output 2

```
1. crab_cakes
2. salad
3. turkey_stuffing
4. turkey
5. almond_pie, pecan_pie
6. coffee

Warning: Rice does not have any ordering.
```

# Author's Note:

This challenge has some subtle ordering logic that might be hard to understand at first. Work through sample data 2 by hand to better understand the ordering rules before writing code. Make sure to expand all widecard rules as well.

Title: [12/05/13] Challenge #138 [Intermediate] Overlapping Circles
Text: # [](#IntermediateIcon) *(Intermediate)*: Overlapping Circles

Computing the volume of a circle is pretty straight-forward: Pi x Radius x Radius, or simply Pi x r ^2.

What if we wanted to computer the volume of two circles? Easy, just sum it! Yet, what about two intersecting circles, much like the classic [Venn diagram](http://en.wikipedia.org/wiki/File:Venn0111.svg)?

Your goal is to write a program that takes two unit-circles (radius of one) at given locations, and compute that shape's volume. You must make sure to *not* double-count the intersecting volume! (i.e. you must not sum this [red area](http://en.wikipedia.org/wiki/File:Venn0001.svg) twice).

As a starting point, check out how to compute [circle segments](http://en.wikipedia.org/wiki/Circular_segment).

# Formal Inputs & Outputs
## Input Description

On standard input you will be given four floating-point space-delimited values: x y u w. x and y are the first circle's position in [Cartesian coordinates](http://en.wikipedia.org/wiki/Cartesian_coordinate_system). The second pair u and w are the second circle's position.

Note that the given circles may not actually intersect. If this is the case, return the sum of both circles (which will always be Pi x 2 since our circles are unit-circles).

## Output Description

Print the summed volume of the two circles, up to an accuracy of 4 digits after the decimal place.

# Sample Inputs & Outputs
## Sample Input

```
    -0.5 0 0.5 0
```

## Sample Output

```
    5.0548
```

Title: [12/1/13] Challenge #139 [Intermediate] Telephone Keypads
Text: # [](#IntermediateIcon) *(Intermediate)*: Telephone Keypads

[Telephone Keypads](http://en.wikipedia.org/wiki/Telephone_keypad) commonly have both digits and characters on them. This is to help with remembering & typing phone numbers (called a [Phoneword](http://en.wikipedia.org/wiki/Phoneword)), like 1-800-PROGRAM rather than 1-800-776-4726. This keypad layout is also helpful with [T9](http://en.wikipedia.org/wiki/T9_(predictive_text\)), a way to type texts with word prediction.

Your goal is to mimic some of the T9-features: given a series of digits from a telephone keypad, and a list of English words, print the word or set of words that fits the starting pattern. You will be given the number of button-presses and digit, narrowing down the search-space.

# Formal Inputs & Outputs
## Input Description

On standard console input, you will be given an array of digits (0 to 9) and spaces. All digits will be space-delimited, unless the digits represent multiple presses of the same button (for example pressing 2 twice gives you the letter 'B').

Use the modern [Telephone Keypads](http://en.wikipedia.org/wiki/Telephone_keypad) digit-letter layout:

```
    0 = Not used
    1 = Not used
    2 = ABC
    3 = DEF
    4 = GHI
    5 = JKL
    6 = MNO
    7 = PQRS
    8 = TUV
    9 = WXYZ
```

You may use any source for looking up English-language words, though this [simple English-language dictionary](http://www.curlewcommunications.co.uk/wordlist.html) is complete enough for the challenge.

## Output Description

Print a list of all best-fitting words, meaning words that start with the word generated using the given input on a telephone keypad. You do not have to only print words of the same length as the input (e.g. even if the input is 4-digits, it's possible there are many long words that start with those 4-digits).

# Sample Inputs & Outputs
## Sample Input

```
    7777 666 555 3
```

## Sample Output

```
    sold
    solder
    soldered
    soldering
    solders
    soldier
    soldiered
    soldiering
    soldierly
    soldiers
    soldiery
```

## Challenge++

If you want an extra challenge, accomplish the same challenge but without knowing the number of times a digit is pressed. For example "7653" could mean sold, or poke, or even solenoid! You must do this efficiently with regards to [Big-O complexity](http://en.wikipedia.org/wiki/Big_O_notation).

Title: [12/18/13] Challenge #140 [Intermediate] Adjacency Matrix
Text: # [](#IntermediateIcon) *(Intermediate)*: Adjacency Matrix

In [graph theory](http://en.wikipedia.org/wiki/Graph_theory), an [adjacency matrix](http://en.wikipedia.org/wiki/Adjacency_matrix) is a data structure that can represent the edges between nodes for a [graph](http://en.wikipedia.org/wiki/Graph_(mathematics\)) in an N x N matrix. The basic idea is that an edge exists between the elements of a row and column if the entry at that point is set to a valid value. This data structure can also represent either a [directed graph](http://en.wikipedia.org/wiki/Directed_graph) or an [undirected graph](http://en.wikipedia.org/wiki/Graph_(mathematics\)#Undirected_graph), since you can read the rows as being "source" nodes, and columns as being the "destination" (or vice-versa).

Your goal is to write a program that takes in a list of edge-node relationships, and print a directed adjacency matrix for it. Our convention will follow that rows point to columns. Follow the examples for clarification of this convention.

[Here's a great online directed graph editor](http://bl.ocks.org/rkirsling/5001347) written in Javascript to help you visualize the challenge. Feel free to post your own helpful links!

# Formal Inputs & Outputs
## Input Description

On standard console input, you will be first given a line with two space-delimited integers N and M. N is the number of nodes / vertices in the graph, while M is the number of following lines of edge-node data. A line of edge-node data is a space-delimited set of integers, with the special "->" symbol indicating an edge. This symbol shows the edge-relationship between the set of left-sided integers and the right-sided integers. This symbol will only have one element to its left, or one element to its right. These lines of data will also never have duplicate information; you do not have to handle re-definitions of the same edges.

An example of data that maps the node 1 to the nodes 2 and 3 is as follows:

```
    1 -> 2 3
```

Another example where multiple nodes points to the same node:

```
    3 8 -> 2
```

You can expect input to sometimes create cycles and self-references in the graph. The following is valid:

```
    2 -> 2 3
    3 -> 2
```

Note that there is no order in the given integers; thus "1 -> 2 3" is the same as "1 -> 3 2".

## Output Description

Print the N x N adjacency matrix as a series of 0's (no-edge) and 1's (edge).

# Sample Inputs & Outputs
## Sample Input

```
    5 5
    0 -> 1
    1 -> 2
    2 -> 4
    3 -> 4
    0 -> 3
```

## Sample Output

```
    01010
    00100
    00001
    00001
    00000
```

Title: [12/23/13] Challenge #140 [Intermediate] Graph Radius
Text: # [](#IntermediateIcon) *(Intermediate)*: Graph Radius

In [graph theory](http://en.wikipedia.org/wiki/Graph_theory), a [graph's radius](http://en.wikipedia.org/wiki/Distance_(graph_theory\)) is the minimum eccentricity of any vertex for a given graph. More simply: it is the minimum distance between all possible pairs of vertices in a graph.

As an example, the [Petersen graph](http://en.wikipedia.org/wiki/Petersen_graph) has a radius of 2 because any vertex is connected to any other vertex within 2 edges.

On the other hand, the [Butterfly graph](http://en.wikipedia.org/wiki/Butterfly_graph) has a radius of 1 since its middle vertex can connect to any other vertex within 1 edge, which is the smallest eccentricity of all vertices in this set. Any other vertex has an eccentricity of 2.

# Formal Inputs & Outputs
## Input Description

On standard console input you will be given an integer N, followed by an [Adjacency matrix](http://en.wikipedia.org/wiki/Adjacency_matrix). The graph is not directed, so the matrix will always be reflected about the [main diagonal](http://en.wikipedia.org/wiki/Main_diagonal).

## Output Description

Print the radius of the graph as an integer.

# Sample Inputs & Outputs
## Sample Input

```
    10
    0 1 0 0 1 1 0 0 0 0
    1 0 1 0 0 0 1 0 0 0
```

```
0 1 0 1 0 0 0 1 0 0
0 0 1 0 1 0 0 0 1 0
1 0 0 1 0 0 0 0 0 1
1 0 0 0 0 0 0 1 1 0
0 1 0 0 0 0 0 0 1 1
0 0 1 0 0 1 0 0 0 1
0 0 0 1 0 1 1 0 0 0
0 0 0 0 1 0 1 1 0 0
```

## Sample Output

```
2
```

Title: [02/26/14] Challenge #150 [Intermediate] Re-emvoweler 1
Text: # _(Intermediate)_: Re-emvoweler 1

In [this week's Easy challenge](http://www.reddit.com/r/dailyprogrammer/comments/1ystvb/022414_challenge_149_easy_disemvoweler/), series of words were disemvoweled into vowels, and non-vowel letters. Spaces were also removed. Your task today is, given the two strings produced via disemvowelment, output _one possibility_ for the original string.

1. Your output must be such that if you put it through the solution to this week's Easy challenge, you'll recover exactly the input you were given.
2. You don't need to output the same string as the one that was originally disemvoweled, just _some_ string that disemvowels to your input.
3. Use [the Enable word list](http://code.google.com/p/dotnetperls-controls/downloads/detail?name=enable1.txt), or some other reasonable English word list. Every word in your output must appear in your word list.
4. For the sample inputs, all words in originally disemvoweled strings appear in Enable. In particular, I'm not using any words with punctuation, and I'm not using the word "a".
5. As before, ignore punctuation and capitalization.

# Formal Inputs & Outputs

## Input description

Two strings, one containing only non-vowel letters, and one containing only vowels.

## Output description

A space-separated series of words that could be disemvoweled into the input, each word of which must appear in your word list.

# Sample Inputs & Outputs

## Sample Input 1

```
wwllfndffthstrds
eieoeaeoi
```

## Sample Output 1

There are, in general, many correct outputs. Any of these is valid output for the sample input (using the Enable word list to verify words):

```
we wile lo fen daff et host rids
we wile lo fend aff eths tor ids
we wile lo fen daff the sot rids
```

```
    we will fend off eths tare do si
    we will fend off the asteroids
```

## Sample Input 2

```
    bbsrshpdlkftbllsndhvmrbndblbnsthndlts
    aieaeaeieooaaaeoeeaeoeaau
```

## Sample Outputs 2

```
    ab bise ars he ae pi ed look fa tab all sned hove me ar bend blob ens than adults
    ai be base rash pe die look fat bal la sned hove me ar bend blob ens than adults
    babies ae rash pe die loo ka fat balls end ho vee mar bend blob ens than adults
    babies rash pedal kef tie bolls nod aah ave omer bendable bones than adults
    babies are shaped like footballs and have more bendable bones than adults
```

## Sample Input 3

```
    llfyrbsshvtsmpntbncnfrmdbyncdt
    aoouiaeaeaoeoieeoieaeoe
```

# Notes

Thanks to /u/abecedarius for inspiring this challenge on /r/dailyprogrammer_ideas!

Think you can do a better job of re-emvoweling? Check out this week's Hard challenge!

Title: [4-19-2014] Challenge #154 [Intermediate] Gorellian Alphabet Sort
Text: #**Description:**
The Gorellians, at the far end of our galaxy, have discovered various samples of English text from our electronic transmissions, but they did not find the order of our alphabet. Being a very organized and orderly species, they want to have a way of ordering words, even in the strange symbols of English. Hence they must determine their own order.

For instance, if they agree on the alphabetical order:

UVWXYZNOPQRSTHIJKLMABCDEFG

Then the following words would be in sorted order based on the above alphabet order:

WHATEVER
ZONE
HOW
HOWEVER
HILL
ANY
ANTLER
COW
***
#**Input:**

The input will be formatted to enter the number of words to sort and the new Alphabet ordering and a list of words to sort. n should be > 0. The alphabet is assumed to be 26 letters with no duplicates and arranged in the new order. Also assumed there are n strings entered.

n (new alphabet ordering)

(word 1 of n)

(word 2 of n)
....

(word n of n)

##Example input 1:

```
8 UVWXYZNOPQRSTHIJKLMABCDEFG
ANTLER
ANY
COW
HILL
HOW
HOWEVER
WHATEVER
ZONE
```

***
#**Output:**

The list of words in sorted order based on the new order of the alphabet. The sort order should be based on the alphabet (case insensitive) and the words should be output to appear as the words were entered.

##Example of output for input 1:

```
WHATEVER
ZONE
HOW
HOWEVER
HILL
ANY
ANTLER
COW
```

***
#**Notes:**

The sorting should be case insensitive. Meaning that you do not sort it based on the ASCII value of the letters but by the letters. Your solution should handle an alphabet order that might be typed in upper/lower case. It will sort the words by this order and output the words as they were typed in.

##Example Input 2:

```
5 ZYXWVuTSRQpONMLkJIHGFEDCBa

go
aLL
ACM
teamS
Go
```

##Example output 2:

```
teamS
go
Go
aLL
ACM
```

***
#**Extra Challenge:**

Error check the input.

***
If the alphabet is missing letters it returns an error message and listing letters missing.

##Input for this:
4 abcdfghijklmnopsuvxz

error
checking
is
fun

##Output for this:

Error! Missing letters: e q r t w y

***
If the alphabet has duplicate letters it returns an error message listing all the duplicate letters used in the alphabet.

##Input for this:

4 abcdefaghijklmnoepqrstiuvwoxuyz

oh
really
yah
really

##Output for this:

Error! Duplicate letters found in alphabet: a e i o u

Title: [26/3/2014] Challenge #155 [Intermediate] We're about to score!
Text: # **Description**

One of the ways that chess games are tracked during play is to assign values to each piece and then look at the pieces that remain on the board for each player. After several moves where pieces have been taken, one can quickly determine who has an advantage.

Pieces are assigned standard valuations:

* pawns are worth one point each.
* Knights and bishops 3 points each
* A Rook is worth 5
* The Queen is worth 9 points.
* The Kings true value is infinite but you shouldn't need to worry about this

More info on chess values can be seen [HERE](http://en.wikipedia.org/wiki/Chess_piece_relative_value)

# **Input Description**

Each line of input will be given in standard chess algebraic notation:

Here's a picture of the notation to give you an idea : [Image](http://home.comcast.net/~danheisman/images/Record_board.jpg)

* columns are given a-h and rows are given 1-8 (starting with white's back row). For reference the queens are on d1 (white) and d8 (black).
* Pieces (except for pawns) have a capital letter associated with them:

King = K; Knight = N; Queen = Q; Rook = R; Bishop = B; None = pawns, they are just specified by their file.
* Captures are marked with an "x":

e.g. "Qxe5" for "queen captures the piece on square e5"; pawn captures are given by file, for example "exd5".

* Castling is indicated as such: O-O for kingside, O-O-O Queenside. Check is indicated by a "+" and checkmate is given by "mate" or "#".

For more help on chess notation see [HERE](http://home.comcast.net/~danheisman/Articles/recording_chess.htm)

# **Formal Input Description**

Three values per line: move number, then white's move, then black's move using chess algebraic notation.

Example:

1. e4 e5 <-- White's pawn to e4, Black's pawn moves to e5
2. Nf3 Nc6 <-- White's Knight moves to f3, Black's Knight moves to c6
3. Bb5 Nf6 <-- White's Bishop moves to b5, Black's Knight moves to f6
4. d3 Bc5 <-- White's Pawn moves to d3, Black's Bishop moves to c5

etc...

# **Formal Output Description**

Your program should emit two values, one for white and one for black, at the end of the series of moves (for an incomplete game).

# **Sample Input**

This is actually Anand v Carlsen from the Zurich Chess Challenge 2014, round 5 play.

1. e4 e5
2. Nf3 Nc6
3. Bb5 Nf6
4. d3 Bc5
5. Bxc6 dxc6
6. h3 Nd7
7. Be3 Bd6
8. Nbd2 O-O
9. O-O Re8
10. Nc4 Nf8
11. d4 exd4
12. Qxd4 c5
13. Qd3 b6
14. Nxd6 Qxd6
15. Qxd6 cxd6
16. Rfd1 Bb7
17. Rxd6 Bxe4

18. Ne1 Rad8
19. Rad1 Ne6
20. Rxd8 Rxd8
21. Rxd8+ Nxd8
22. f3 Bd5
23. a3 Nc6
24. Kf2 f6
25. Nd3 Kf8
26. Ke2 Ke7
27. Kd2 Kd7
28. Nf4 Bf7
29. b3 Ne7
30. h4 Nd5

# **Sample output**

12-12

# **Challenge Input**

This is actually Aronian vs So from the 2014 76th Tata Steel Masters round 6. Aronian would go on to win.

1. c4 Nf6
2. Nf3 g6
3. Nc3 d5
4. cxd5 Nxd5
5. e4 Nxc3
6. bxc3 Bg7
7. Be2 c5
8. O-O Nc6
9. Qa4 Bd7
10. Qa3 Qa5
11. Rd1 O-O
12. Rb1 b6
13. d4 Qxa3
14. Bxa3 Bg4
15. dxc5 Bxc3
16. Ba6 Rab8
17. Rdc1 Bxf3
18. gxf3 Bd2
19. Rd1 Bc3
20. Kg2 bxc5
21. Bxc5 Bb4
22. Be3 Bd6
23. Rbc1 Nb4
24. Bc4 Rfc8
25. f4 Kf8
26. a3 Nc6
27. Ba6 Bxa3

Title: [4/2/2014] Challenge #156 [Intermediate] Managing Workers
Text: # [](#IntermediateIcon) _(Intermediate)_: Managing Workers

After yesterday's April Fools shenanigans, management worldwide must work at full pace to make up for lost productivity from the innumerable ThinkGeek pranks aimed at coworkers. You've been hired by some random company to create a program which lets them organise their workers to do a set of given tasks in a project as efficiently as possible.

Each task is described by its duration (in days). Each worker can only do one task at once, and tasks must be done as a whole - ie. you can't do one half at one point and then another half later on. However any number of tasks can be performed concurrently by different workers. You will also be given the maximum length of time, in days, that the overall project can go on for.

The catch is - some tasks depend on other tasks to be fully completed before they themselves can be started. If Task A needs Task B and C to be completed before it can begin, then Tasks B and C are *dependencies* of Task A.

Your challenge is to try and find a way of scheduling the workers such that the number of workers (and idle time of each worker) is minimised.

# Formal Inputs and Outputs

## Input Description

On the console, you will be given numbers N and T, separated by commas. **N** represents the number of tasks in the project, and **T** represents the maximum time the project may go on for. Next you will be given a list of tasks, in the format:

```
Name, D [, Dependency]
```

Where **Name** is the name of the task, and **D** is its duration. The number of dependencies may be zero or one. There will be no circular dependencies. Dependencies are referenced by name.

## Output Description

You must print the total number of workers assigned. Then the assigned tasks for each worker, and starting on which day, in the format:

```
N, Name, S
```

Where **N** is the worker number (starting from 1, eg. Worker 1, Worker 2, Worker 3, etc.), **Name** is the name of the task, and **S** is the starting day (starting from Day 1.)

Finally you must print the total number of idle (not working) worker days, **I**. So if Worker 1 has 3 off days and Worker 2 has 5, then print 8.

# Sample Inputs & Outputs

## Sample Input

```
6,12
Lights,2,Wiring
Windows,3
Insulation,4
Painting,4
Wiring, 6
Cleaning,7,Painting
```

## Sample Output

```
3
1,Painting,1
```

```
1,Cleaning,5
2,Windows,1
2,Wiring,4
2,Lights,10
3,Insulation,1
10
```

# Challenge

## Challenge Input

```
13,17
Preparation,2,Planning
Hiring,3
Legal,3
Briefing,4,Preparation
Advertising,4
Paperwork,5,Legal
Testing,5,Frontend
API,6
Backend,6
Planning,7
Frontend,8
Mobile,8
Documentation,9,API
```

## Possible Challenge Output

```
5
1,Frontend,1
1,Paperwork,9
1,Advertising,14
2,Hiring,1
2,Mobile,4
2,Testing,12
3,Legal,1
3,Backend,4
3,Preparation,10
3,Briefing,12
4,API,1
4,Documentation,7
5,Planning,1
15
```

# Hint

This can be partly solved using bin-packing.

Title: [4/9/2014] Challenge #157 [Intermediate] Puzzle Cube Simulator
Text: # [](#IntermediateIcon) _(Intermediate)_: Puzzle Cube Simulator

You may be aware of puzzles such as the [Rubik's Cube](http://img1.wikia.nocookie.net/__cb20130909182419/maditsmadfunny/images/e/ee/Rubik%27s_Cube_cropped.jpg). They work by having pieces with coloured faces which can rotate around the centers. You may also be aware of higher-order puzzles such as the [Professor's Cube](http://upload.wikimedia.org/wikipedia/commons/f/fe/Professors_cube.jpg). These work in exactly the same way, with the exception of having more pieces. For the purposes of this challenge, an **n**-cube is a puzzle with **n** pieces along an edge - the Rubik's cube would be a 3-cube, and the Professor's cube a 5-cube.

To make it easier to see exactly what people are doing, there is a standard set of what is called *Move Notation*, which tells you exactly how the puzzle was turned. For the purpose of this challenge, the [notation defined in Article 12 of the WCA regulations](https://www.worldcubeassociation.org/regulations/#article-12-notation) will be used. In a nutshell:

* There are 6 faces. U (up, the top face). D (down, the bottom face). L (left). R (right). F (front). B (back).
* Each face is turned like you were looking at it from the front.
* A notation such as `X` means you turn the X face clockwise 90'. So `R L` means turn the right face clockwise 90' (from its perspective), then the left face clockwise 90' (from its perspective).
* A notation such as `X'` (pronounced *prime*) means you turn the X face anticlockwise 90'. So `R U'` means turn the right face clockwise 90', then the top face anticlockwise 90'.
* A notation such as `X2` means you turn the X face 180'.

This lets you signify a sequence of moves, such as `R U R' U'  R' F  R2 U' R' U  R U R' F'` - which lets you know exactly what happened to the puzzle.

Your challenge is, given a **3**-cube (the standard cube) and a sequence of moves, to simulate the turning of a puzzle and print the output state at the end. (you don't have to solve it - phew!)

Assume a standard colour scheme. That is, start with white on the bottom (D), yellow on the top (U), red on the front (F), green on the right (R), orange on the back (B) and blue on the left (L).

# Formal Inputs and Outputs

## Input Description

You will be given, on one line (and separated by spaces), a sequence of moves in WCA standard notation. This will be arbitrarily long, within sensible limits.

## Output Description

You must print out the front face *only* of a cube that has been turned in the way described by the input (as if you were looking at it from the front of the cube.) Each colour will be represented by its first letter (r, o, y, g, b, w) and the face shall be represented as a printed square.
For example:

```
  rrb
  rrw
  oww
```

# Sample Inputs & Outputs

## Sample Input

```
        U2 R' D2 R F L' U2 R
```

## Sample Output

```
  rrb
  rrw
  oww
```

# Challenge

## Challenge Input

```
        R U2 F2 D' F' U L' D2 U2 B' L R2 U2 D
```

## Challenge Output

```
    bbo
    yrb
    oow
```

# Hint

Multidimensional arrays will be useful here. Try to visualise the way pieces are moved around when you turn a face.

Title: [4/16/2014] Challenge #158 [Intermediate] Part 1 - The ASCII Architect
Text: #Description

In the far future, demand for pre-manufactured housing, particularly in planets such as Mars, has risen very high. In fact, the demand is so much that traditional building planning techniques are taking too long, when faced with the "I want it now!" mentality of the denizens of the future. You see an opportunity here - if you can cheaply generate building designs, you are sure to turn a huge profit.

You decide to use ASCII to design your buildings. However, as you are lazy and wish to churn out many designs quickly, you decide to simply give the computer a string, and have the computer make the building for you.

#Formal input & output

Input

Input will be to STDIN, or read from a file input.txt located in the working directory of the operating system. Input consists of one line between 1 to 231-1 length. The line can be assumed to only contain the lowercase letters from a to j, and numbers from 1 to 9. It can also be assumed that a number will not immediately follow another number in the string (i.e. if the 4th character is a number, the 5th character is guaranteed to be a letter, not a number.)

Output

Output will be to STDOUT, or written to a file output.txt in the working directory. For each non-number character of input, the output will contain a vertical line composed as shown [here](http://i.imgur.com/twPajPG.png):

A letter can also be prefixed by a number n, where n is an integer between 1 and 9. In this case, n whitespaces must be at the bottom of the vertical line. For example, 3b would output

```
+
+
S
S
S
```

Where spaces are identified with a capital S (In your actual output, it should be actual spaces).
Sample Inputs and Outputs

#Sample input 1 (Bridge)

j3f3e3e3d3d3c3cee3c3c3d3d3e3e3f3fjij3f3f3e3e3d3d3c3cee3c3c3d3d3e3e3fj

#Sample output

```
   .              . .              .
   .*             **...**          *.
   .***        ****...****        ***.
   *_____     ______***______     _____*
   *_______  ________***________  _______*
   *+++++++**++++++++***++++++++**+++++++*
   -+++++++--++++++++---++++++++--+++++++-
   -       --       ---       --       -
   +       ++       +++       ++       +
   +       ++       +++       ++       +
```

#Notes

Try making your own buildings as well instead of just testing the samples. Don't forget to include your favourite ASCII construction with your solution!

Title: [4/23/2014] Challenge #159 [Intermediate] Rock Paper Scissors Lizard Spock - Part 2 Enhancement
Text: #Theme Week:

We continue our theme week challenge with a more intermediate approach to this game. We will be adding on to the challenge from monday. Those who have done monday's challenge will find this challenge a little easier by just modifying what they have done from monday.

[Monday's Part 1 Challenge]
(http://www.reddit.com/r/dailyprogrammer/comments/23lfrf/4212014_challenge_159_easy_rock_paper_scissors/)

#Description:

We are gonna upgrade our game a bit. These steps will take the game to the next level.

Our computer AI simply randoms every time. We can go a step further and implement a basic AI agent that learns to create a better way in picking. Please add the following enhancements from monday's challenge.

* Implement a Game Loop. This should be a friendly menu that lets the player continue to play matches until they pick an option to quit.
* Record the win and tie record of each player and games played.
* At termination of game display games played and win/tie records and percentage (This was the extra challenge from monday)
* Each time the game is played the AI agent will remember what the move of the opponent was for that match.
* The choice of what move the computer picks in future games will be based on taking the top picks so far and picking from the counter picks. In the case of a tie for a move the computer will only random amongst the counter moves of those choices and also eliminate from the potential pool of picks any moves it is trying to counter to lessen the chance of a tie.

Example of this AI.

Game 1 - human picks rock

Game 2 - human picks paper

Game 3 - human picks lizard

Game 4 - human picks rock

For game 5 your AI agent detects rock as the most picked choice. The counter moves to rock are Spock and Paper. The computer will randomized and pick one of these for its move.

Game 5 - human picks lizard.

For game 6 your AI agent sees a tie between Rock and Lizard and then must decide on a move that counters either. The counters could be Spock, Paper, Rock, Scissors. Before picking eliminate counters that match any of the top picks. So since Rock was one of the top picks so far we eliminate it as a possible counter to prevent a tie. So random between Spock, Paper and Scissors.

if for any reason all choices are eliminated then just do a pure random pick.

#Input:

Design a menu driven or other interface for a loop that allows the game to play several games until an option/method is used to terminate the game.

Design and look is up to you.

#Output:

Similar to monday. So the moves and winner. On termination of the game show the number of games played. For each player (human and computer) list how many games they won and the percentage. Also list how many tie games and percentage.

#For Friday:

Friday we will be kicking this up further. Again I suggest design solutions so that you can pick which AI you wish to use (Either a pure random or this new AI for this challenge) as the Bot for making picks.

#Extra Challenge:

The menu system defaults to human vs new AI. Add a sub-menu system that lets you define which computer AI you are playing against. This means you pick if you are human vs random AI (from monday) or you can do human vs Learning AI (from this challenge).

Play 10 games against each AI picking method and see which computer AI has the better win rate.

#Note on the AI:

Friday will have a few steps. One is make your AI that is better than this one. The intent of this AI was to either give guidance to those who don't wish to develop their own AI and also to test to see if it is better than a true random pick. It was not intended to be good or bad.

Those who wish to develop their own AI for the intermediate I would encourage you to do so. It has to be more complex than just simply doing a pure random number to pick. Doing so will get you a step ahead.

Title: [4/30/2014] Challenge #160 Intermediate Part 2 - Damage Control
Text: [Part 1](http://www.reddit.com/r/dailyprogrammer/comments/236va2/4162014_challenge_158_intermediate_part_1_the/)

#Introduction

The new building techniques are a massive success, and soon it is adopted all across the far future society. However, suddenly a great swarm of high-tech termites are predicted to strike - and worse, due to a bug in /u/1337C0D3R's code, the design of the buildings are shoddy and are prone to being destroyed easily. If the buildings are destroyed by the army of termites this could lead to a crisis.

The slightly incompetent government of the future has realized that it is incumbent for them to act. They can provide you with a number of Reinforcement Kits 3000tm that when placed on a building, prevents the building from being destroyed. However, the Reinforcement Kit 3000tm is expensive to produce, so you decide to design an algorithm to use the least number of kits, and save the most money.
Description

The threatened buildings are placed in a straight line, numbered from 1 to N. Each building shares a wall with the buildings next to them - the adjacent buildings are known as 'neighbours'. This is an example of how the buildings would be set up for N = 12:

```
----------------------------------------------------
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
----------------------------------------------------
```

Each day the termites will start at one building and completely, irreversibly destroy it. After having destroyed the building, the termites will then spread to, but not destroy yet, all buildings that can be reached from the building that they started at. They cannot pass through buildings that are already destroyed. In other words, the termites cover all the area of a flood-fill from the starting building, with destroyed buildings as the boundary.

The termites will destroy the buildings that they have spread to unless a Reinforcement Kittm is placed on the building. After the termites have spread fully, you may begin placing kits. A Reinforcement Kittm will kill all termites in the building it is placed in. However, they only have an effect for one day; if on the next day the building again has termites another Reinforcement Kit must be used.

Given a list of P buildings that will be destroyed in P days, find the minimum number of Reinforcement Kits required, given that the buildings may be destroyed in any order. (The government has also given you Termite Bait which lets you choose the order in which the buildings in the list are destroyed).

#Formal Inputs and Outputs
Input Description

Input will be given on STDIN, or read from a file input.txt located in the working directory of the operating system. There will be exactly 2 lines of input. The first line contains two integers that are space separated, N and P. N is the number of buildings in the line. P is the number of buildings that will be destroyed in P days.

The second line consists of space-separated integers. The total number of integers will be equal to P. These are the indexes of the buildings which are to be destroyed.
Output Description

Output will be to STDOUT, or written to a file output.txt in the working directory. Output will contain a single integer consisting of the minimum number of Reinforcement Kits required.
#Sample Inputs and Outputs
#Sample Input 1

8 1

3

#Sample Output 1

7

#Sample Input 2

20 3

3 6 14

#Sample Output 2

35

Title: [5/7/2014] Challenge #161 [Medium] Appointing Workers
Text: # [](#IntermediateIcon) _(Intermediate)_: Appointing Workers

In the past, we've already tackled the challenge of deciding in which order to do certain jobs. However, now you need to work out which worker gets which job. What if some workers are only qualified to do certain jobs? How do you ensure there are no jobs or workers left out? Your challenge now is (given some jobs that need to be done, and some workers and the jobs they're allowed to do) compute who should be given which job, so no-one is doing a job they are not qualified for.

# Formal Inputs and Outputs

## Input Description

On the console, you will be given numbers N. **N** represents the number of jobs that need to be done, and the number of workers.^(see footnote) To keep this challenge at an Intermediate level, the number of workers and jobs will always be the same.

You will then be given a list of **N** jobs (on separate lines), followed by **N** workers and the jobs they're allowed to do (separated by commas, one worker per line).

Note that there may be more than one possible assignment of workers.

## Output Description

You must print the list of workers, along with the job each worker is assigned to.

# Sample Inputs & Outputs

## Sample Input

```
5
Wiring
Insulation
Plumbing
Decoration
Finances
Alice Wiring,Insulation,Plumbing
Bob Wiring,Decoration
Charlie Wiring,Plumbing
David Plumbing
Erin Insulation,Decoration,Finances
```

## Sample Output

```
Alice Insulation
Bob Decoration
Charlie Wiring
David Plumbing
Erin Finances
```

# Challenge

## Challenge Input

```
6
GUI
Documentation
Finances
Frontend
Backend
Support
Alice GUI,Backend,Support
Bill Finances,Backend
Cath Documentation,Finances
Jack Documentation,Frontend,Support
Michael Frontend
Steve Documentation,Backend
```

## Challenge Output

Note that this is just one possible solution - there may be more.

```
Alice GUI
Bill Backend
Cath Finances
Jack Support
Michael Frontend
Steve Documentation
```

# Hint

This problem is called the Matching problem in usual terms.

Title: [5/14/2014] Challenge #162 [Intermediate] Novel Compression, pt. 2: Compressing the Data
Text: # [](#IntermediateIcon) _(Intermediate)_: Novel Compression, pt. 2: Compressing the Data

Welcome to Part 2 of this week's Theme Week. Today we are (predictably) doing the opposite of Monday's challenge. We will be taking uncompressed data, running it through a compression algorithm, and printing compressed data. The grammar and format is exactly the same as last time.

You are still advised to write your program in a way that can be easily adapted and extended later on. A challenge later this week will involve putting all of your work together into a fully featured program!

# Formal Inputs and Outputs

## Input Description

The input will simply be uncompressed textual data. At the end, an EOF symbol is printed (note: in Windows an EOF is entered using Ctrl-Z on the console, and in Linux an EOF is entered using Ctrl-D at a terminal - or alternatively pipe a file containing the input using `cat`.)

## Data Format

[Same rules as before](/r/dailyprogrammer/comments/25clki/5122014_challenge_162_easy_novel_compression_pt_1/). All words must go into a dictionary (just a list of words.)

* If a lower-case word (eg. `stanley`) is encountered, print its index in the dictionary, followed by a space.

* If a capitalised word (first letter is upper-case, eg. `Stanley`) is encountered, print its index in the dictionary, followed by a caret (`^`), followed by a space.

* If an upper-case word (eg. `Stanley`) is encountered, print its index in the dictionary, followed by an exclamation point (`!`), followed by a space.

* If the previous and next words encountered are joined by a hyphen rather than a space (eg. `hunter-gatherer`), print a hyphen (`-`), followed by a space (eg. `44 - 47 `).

* If word is followed by any of the following symbols: `. , ? ! ; :`, print that symbol after it, followed by another space (eg. `44 !`).

* If a new line is encountered, print the letter `R`, followed by a space.

* If the end of the input has been reached, print the letter `E`, followed by a space.

Note: All words will be in the Latin alphabet.

Now for an important bit. If you encounter any of the following:

* A word is capitalised in any other different way than above,

* A word is not alphabetical (eg. has numbers in it),

* A symbol not in `. , ? ! ; :` is encountered,

* Two or more symbols are next to each other like `??1`),

Then you must print an error message and then stop, because our simple basic compression format cannot account for these cases. Normally a practical compression system would handle it more gracefully, but this is just a challenge after all so just drop them.

## Example Data

Therefore, if our input is given as:

```
The quick brown fox jumps over the lazy dog.
Or, did it?
```

Then the output data is:

```
11
the
quick
brown
fox
jumps
over
lazy
dog
or
did
it
0^ 1 2 3 4 5 0 6 7 . R 8^ , 9 10 ? E
```

## Output Description

Print the resultant data from your compression algorithm, using the rules described above.

# Challenge

## Challenge Input

```
I would not, could not, in the rain.
Not in the dark. Not on a train.
Not in a car. Not in a tree.
I do not like them, Sam, you see.
Not in a house. Not in a box.
Not with a mouse. Not with a fox.
I will not eat them here or there.
I do not like them anywhere!
```

## Example Challenge Output

Your output may vary slightly depending on how you populate your word dictionary.

```
30
i
would
not
could
in
the
rain
dark
on
a
train
car
tree
do
like
them
sam
you
see
house
box
with
mouse
fox
will
eat
here
or
there
anywhere
0^ 1 2 , 3 2 , 4 5 6 . R 2^ 4 5 7 . 2^ 8 9 10 . R 2^ 4 9 11 . 2^ 4
9 12 . R 0^ 13 2 14 15 , 16^ , 17 18 . R 2^ 4 9 19 . 2^ 4 9 20 . R 2^ 21 9
22 . 2^ 21 9 23 . R 0^ 24 2 25 15 26 27 28 . R 0^ 13 2 14 15 29 ! R E
```

Title: [5/21/2014] Challenge #163 [Intermediate] Fallout's Hacking Game
Text: #Description:

The popular video games ***Fallout 3*** and ***Fallout: New Vegas*** has a [computer hacking mini game] (http://gamewiki.net/Fallout_3/Hacking_Guide).

This game requires the player to correctly guess a password from a list of same length words. Your challenge is to implement this game yourself.

The game works like the classic game of [Mastermind] (http://en.wikipedia.org/wiki/Mastermind_(board_game\))
The player has only 4 guesses and on each incorrect guess the computer will indicate how many letter positions are correct.

For example, if the password is MIND and the player guesses MEND, the game will indicate that 3 out of 4 positions are correct (M_ND). If the password is COMPUTE and the player guesses PLAYFUL, the game will report 0/7. While some of the letters match, they're in the wrong position.

Ask the player for a difficulty (very easy, easy, average, hard, very hard), then present the player with 5 to 15 words of the same length. The length can be 4 to 15 letters. More words and letters make for a harder puzzle. The player then has 4 guesses, and on each incorrect guess indicate the number of correct positions.

Here's an example game:

```
    Difficulty (1-5)? 3
    SCORPION
    FLOGGING
    CROPPERS
    MIGRAINE
    FOOTNOTE
    REFINERY
    VAULTING
    VICARAGE
    PROTRACT
    DESCENTS
    Guess (4 left)? migraine
    0/8 correct
    Guess (3 left)? protract
    2/8 correct
    Guess (2 left)? croppers
    8/8 correct
    You win!
```

You can draw words from our favorite dictionary file: [enable1.txt] (https://code.google.com/p/dotnetperls-controls/downloads/detail?name=enable1.txt) . Your program should completely ignore case when making the position checks.

#Input/Output:

Using the above description, design the input/output as you desire. It should ask for a difficulty level and show a list of words and report back how many guess left and how many matches you had on your guess.

The logic and design of how many words you display and the length based on the difficulty is up to you to implement.

#Easier Challenge:
The game will only give words of size 7 in the list of words.

Title: [5/28/2014] Challenge #164 [Intermediate] Part 3 - Protect The Bunkers
Text: ##Description
Most of the residential buildings have been destroyed by the termites due to a bug in /u/1337C0D3R's code. All of the civilians in our far-future society now live in bunkers of a curious design - the bunkers were poorly designed using the ASCII Architect and are thus not secure. If the bunkers are breached by a hostile force, it is almost certain that all the civilians will die.

The high-tech termites have developed a taste for human flesh. Confident from their victory at the building lines, they are now staging a full attack on the bunkers. The government has hired you to design protective walls against the termite attacks. However, their supplies are limited, so you must form a method to calculate the minimum amount of walls required.

A map of an area under assault by evil termites can be described as a 2d array of length m and width n. There are five types of terrain which make up the land:

* *: A termite nest. Termites can pass through here. The termites begin their assault here. Protective walls cannot be placed here.
* #: Impassible terrain. Termites cannot pass through here. Protective walls cannot be placed here.
* +: Unreliable terrain. Termites can pass through here. Protective walls cannot be placed here.
* -: Reliable terrain. Termites can pass through here. Protective walls can be placed here.
* o: Bunker. Termites can pass through here. If they do, the civilians die a horrible death. Protective walls cannot be placed here.

Termites will begin their attack from the nest. They will then spread orthogonally (at right angles) through terrain they can pass through.

A map will always follow some basic rules:

* There will only be one nest.
* Bunkers will always be in a single filled rectangle (i.e. a contiguous block).
* A bunker will never be next to a nest.
* There will always be a solution (although it may require a lot of walls).

##Formal Inputs And Outputs
##Input Description

Input will be given on STDIN, read from a file map.txt, or supplied as a command line argument. The first line of input will contain 2 space separated integers m and n. Following that line are n lines with m space seperated values per line. Each value will be one of five characters: *, #, +, -, or o.

Input Limits

```
    1 <= n < 16
    3 <= m < 16
```

##Output Description

Output will be to STDOUT or written to a file output.txt. Output consists of a single integer which is the number of walls required to protect all the bunkers.
##Sample Inputs and Outputs
##Sample Input 1

6 6

#++++*

#-#+++

#--#++

#ooo--

```
#ooo-#

######
```

##Sample Output 1

2

(The walls in this example are placed as follows, with @ denoting walls:

```
#++++*

#@#+++

#--#++

#ooo@-

#ooo-#

######
```

Title: [6/4/2014] Challenge #165 [Intermediate] ASCII Maze Master
Text: # [](#IntermediateIcon) _(Intermediate)_: ASCII Maze Master

We're going to have a slightly more logical puzzle today. We're going to write a program that will find a path through a simple maze.

A simple maze in this context is a maze where all of the walls are connected to each other. Take this example maze segment.

```
# # ### #
# #
# ### B #
#   # B #
# B # B #
# B   B #
# BBBBB #
#       #
#########
```

See how the wall drawn with `B`s isn't connected to any other walls? That's called a floating wall. A simple maze contains no floating walls - ie. there are no loops in the maze.

# Formal Inputs and Outputs

## Input Description

You will be given two numbers **X** and **Y**. After that you will be given a textual ASCII grid, **X** wide and **Y** tall, of walls `#` and spaces. In the maze there will be exactly one letter `S` and exactly one letter `E`. There will be no spaces leading to the outside of the maze - ie. it will be fully walled in.

## Output Description

You must print out the maze. Within the maze there should be a path drawn with askerisks `*` leading from the letter `S` to the letter `E`. Try to minimise the length of the path if possible - don't just fill all of the spaces with `*`!

# Sample Inputs & Output

## Sample Input

```
15 15
###############
#S        #   #
### ### ### # #
#   #   #   # #
# ##### ##### #
#     #   #   #
# ### # ### ###
# #   # #   # #
# # ### # ### #
# # # # # #   #
### # # # # # #
#   #   # # # #
# ####### # # #
#           #E#
###############
```

## Sample Output

```
###############
#S**      #   #
###*### ### # #
#***#   #   # #
#*##### ##### #
#*****#   #   #
# ###*# ### ###
# #***# #   # #
# #*### # ### #
# #*# # # #***#
###*# # # #*#*#
#***#   # #*#*#
#*####### #*#*#
#***********#E#
###############
```

# Challenge

## Challenge Input

```
41 41
#########################################
#   #       #     #           #         #
# # # ### # # ### # ####### ### ####### #
# #S#   # #   #   # #     #           # #
# ##### # ######### # # ############# # #
# #     # #         # #       #   #   # #
# # ##### # ######### ##### # # # # ### #
# #     #   #     #     #   # # # # # # #
# ##### ######### # ##### ### # # # # # #
#   #           #   #     #   # # #   # #
# ### ######### # ### ##### ### # ##### #
#   #   #     # # #   #       # #       #
# # ### # ### # ### ### ####### ####### #
# #     # #   #     #   # #     #     # #
# ####### # ########### # # ##### # ### #
#     # # #   #       #   # #   # #     #
```

```
##### # ##### # ##### ### # ### # #######
#   # #     # #   #   #   # #   #     # #
# ### ### ### ### # ### ### # ####### # #
#   #     #   #   # #   #   # #     #   #
### ##### # ### ### ### # ### # ### ### #
#       # #   # # #   # # #   # # #     #
# ####### ### # # ### ### # ### # #######
#       #   #   #   # #   #     #       #
# ##### ### ##### # # # ##### ### ### ###
#   # # #   #     # # #     # #     #   #
### # # # ### # ##### # ### # # ####### #
# #   #   #   # #     #   # # # #     # #
# ### ##### ### # ##### ### # # # ### # #
#   #       #   # # #   #   # # #   #   #
# # ######### ### # # ### ### # ### #####
# #     #   # # # #   #   # # #   #     #
# ##### # # # # # ### # ### # ######### #
# #   # # # # # #   # #   #             #
# # # # # # # # ### ### # ############# #
# # #     # # #   #   # #       #       #
# ######### # # # ### ### ##### # #######
#     #     # # #   #   # #     # #     #
# ### ####### ### # ### ### ##### # ### #
#   #             #   #     #       #E  #
#########################################
```

# Notes

One easy way to solve simple mazes is to always follow the wall to your left or right. You will eventually arrive at the end.

Title: [6/14/2014] Challenge #166b [Intermediate] Prime Factor Trees
Text: # [](#IntermediateIcon) _(Intermediate)_: Prime Factor Trees

Every number can be represented as the product of its [prime factors](http://en.wikipedia.org/wiki/Prime_factor). These are all of the prime numbers which the number is divisible by - if a number has no prime factors except itself, then it is prime (because it cannot be divided by any other number.) Finding the prime factor representation of a number comes in handy in quite a few ways - one of which is being able to easily find the [Greatest Common Divisor](http://en.wikipedia.org/wiki/Greatest_common_divisor).

One of the first techniques schoolchildren learn to find a number's prime factors is a technique known as factor trees. To create a factor tree, write down the number you are factoring first.

```
60
```

Then, find a number that divides this cleanly, and find the answer - 60 can be divided by 4 to get 15, for example. Once we've done that, write those two numbers under 60 on 'branches', like so:

```
   60
    |
 4--+--15
```

Then, do the same for each of those numbers, too:

```
    60
     |
  4--+--15
  |
2-+-2
```

And finally:

```
    60
     |
  4--+--15
  |      |
2-+-2  3-+-5
```

Once a prime number (such as the bottom row) is created, you can't factor any further, so you stop.

Your challenge is, given a number, generate its factor tree.

# Formal Inputs and Outputs

## Input Description

You will be given a number **N** which you are to generate a factor tree for.

## Output Description

Print the factor tree in a similar format to the ones above.

# Challenge

## Challenge Input

```
1767150
```

## Sample Challenge Output

There are a lot of different ways to display a factor tree for some numbers. Here are some examples.

```
        1767150
           |
   1309-----+-----1350
     |            |
  77-+--17     45---+---30
  |           |       |
 7+-11      9-+--5   6-+--5
           |        |
          3+-3     2+-3

        1767150
             |
      1350----+-----1309
        |             |
   45---+---30      77-+--17
   |        |      |
5-+-9     6-+--5  7+-11
  |     |
 3+-3  2+-3
```

# Notes

If you're having trouble with the tree printing logic, that's fine - you can skip that if you want. Print it a different way that's easier to format.

Title: [6/25/2014] Challenge #168 [Intermediate] Block Count, Length & Area
Text: #Description:

In construction there comes a need to compute the length and area of a jobsite. The areas and lengths computed are used by estimators
to price out the cost to build that jobsite. If for example a jobsite was a building with a parking lot and had concrete walkways and some nice
pavers and landscaping it would be good to know the areas of all these and some lengths (for concrete curbs, landscape headerboard, etc)

So for today's challenge we are going to automate the tedious process of calculating the length and area of aerial plans or photos.

#ASCII Photo:

To keep this within our scope we have converted the plans into an ASCII picture. We have scaled the plans so 1 character is a square with dimensions of 10 ft x 10 ft.

The photo is case sensitive. so a "O" and "o" are 2 different blocks of areas to compute.

#Blocks Counts, Lengths and Areas:

Some shorthand to follow:

* SF = square feet
* LF = linear feet

If you have the following picture.

```
####
OOOO
####
mmmm
```

* # has a block count of 2. we have 2 areas not joined made up of #
* O and m have a block count of 1. they only have 1 areas each made up of their ASCII character.
* O has 4 blocks. Each block is 100 SF and so you have 400 SF of O.
* O has a circumference length of that 1 block count of 100 LF.
* m also has 4 blocks so there is 400 SF of m and circumference length of 100 LF
* # has 2 block counts each of 4. So # has a total area of 800 SF and a total circumference length of 200 LF.

Pay close attention to how "#" was handled. It was seen as being 2 areas made up of # but the final length and area adds them together even thou they not together. It recognizes the two areas by having a block count of 2 (2 non-joined areas made up of "#" characters) while the others only have a block count of 1.

#Input:

Your input is a 2-D ASCII picture. The ASCII characters used are any non-whitespace characters.

##Example:

```
####
@@oo
o*@!
****
```

#Output:

You give a Length and Area report of all the blocks.

##Example: (using the example input)

```
    Block Count, Length & Area Report
    =================================

    #: Total SF (400), Total Circumference LF (100) - Found 1 block
    @: Total SF (300), Total Circumference LF (100) - Found 2 blocks
    o: Total SF (300), Total Circumference LF (100) - Found 2 blocks
    *: Total SF (500), Total Circumference LF (120) - Found 1 block
    !: Total SF (100), Total Circumference LF (40) - Found 1 block
```

#Easy Mode (optional):

Remove the need to compute the block count. Just focus on area and circumference length.

#Challenge Input:

So we have a "B" building. It has a "D" driveway. "O" and "o" landscaping. "c" concrete walks. "p" pavers. "V" & "v" valley gutters. @ and T tree planting.
Finally we have # as Asphalt Paving.

```
    ooooooooooooooooooooooDDDDDooooooooooooooooooooooooooooo
    ooooooooooooooooooooooDDDDDooooooooooooooooooooooooooooo
    ooo##################o#####o#########################ooo
    o@o##################o#####o#########################ooo
    ooo##################o#####o#########################oTo
    o@o##################################################ooo
    ooo##################################################oTo
    o@o############ccccccccccccccccccccccc###############ooo
    pppppppppppppppcOOOOOOOOOOOOOOOOOOOOOc###############oTo
    o@o############cOBBBBBBBBBBBBBBBBBBBOc###############ooo
    ooo####V#######cOBBBBBBBBBBBBBBBBBBBOc###############oTo
    o@o####V#######cOBBBBBBBBBBBBBBBBBBBOc###############ooo
    ooo####V#######cOBBBBBBBBBBBBBBBBBBBOcpppppppppppppppppp
    o@o####V#######cOBBBBBBBBBBBBBBBBBBBOc###############ooo
    ooo####V#######cOBBBBBBBBBBBBBBBBBBBOc######v########oTo
    o@o####V#######cOBBBBBBBBBBBBBBBBBBBOc######v########ooo
    ooo####V#######cOOOOOOOOOOOOOOOOOOOOOc######v########oTo
    o@o####V#######ccccccccccccccccccccccc######v########ooo
    ooo####V#######ppppppppppppppppppppppp######v########oTo
    o@o############ppppppppppppppppppppppp###############ooo
    oooooooooooooooooooooooooooooooooooooooooooooooooooooooo
    oooooooooooooooooooooooooooooooooooooooooooooooooooooooo
```

#FAQ:

Diagonals do not connect. The small example shows this. The @ areas are 2 blocks and not 1 because of the Diagonal.

Title: [7/2/2014] Challenge #169 [Intermediate] Home-row Spell Check
Text: #User Challenge:

Thanks to /u/Fruglemonkey. This is from our idea subreddit.

http://www.reddit.com/r/dailyprogrammer_ideas/comments/26pak5/intermediate_homerow_spell_check/

#Description:

Aliens from Mars have finally found a way to contact Earth! After many years studying our computers, they've finally created their own computer and keyboard to send us messages. Unfortunately, because they're new to typing, they often put their fingers slightly off in the home row, sending us garbled messages! Otherwise, these martians have impeccable spelling. You are tasked to create a spell-checking system that recognizes words that have been typed off-center in the home row, and replaces them with possible outcomes.

#Formal Input:

You will receive a string that may have one or more 'mis-typed' words in them. Each mis-typed word has been shifted as if the hands typing them were offset by 1 or 2 places on a QWERTY keyboard.

Words wrap based on the physical line of a QWERTY keyboard. So A left shift of 1 on Q becomes P. A right shift of L becomes A.

#Formal Output:
The correct string, with corrected words displayed in curly brackets. If more than one possible word for a mispelling is possible, then display all possible words.

#Sample Input:
The quick ntpem fox jumped over rgw lazy dog.

#Sample Output:
The quick {brown} fox jumped over {the} lazy dog.

#Challenge Input:
 Gwkki we are hyptzgsi martians rt zubq in qrsvr.

#Challenge Input Solution:

{Hello} we are {friendly} martians {we} {come} in {peace}

#Alternate Challenge Input:

A oweaib who fprd not zfqzh challenges should mt ewlst to kze

#Alternate Challenge Output:

A {person} who {does} not {check} challenges should {be} {ready} to {act}

#Dictionary:

Good to have a source of words. Some suggestions.

* [enable1.txt] (https://code.google.com/p/dotnetperls-controls/downloads/detail?name=enable1.txt)
* [British English Word List] (http://www.curlewcommunications.co.uk/wordlist.html)

#FAQ:

As you can imagine I did not proof-read this. So lets clear it up.

Shifts can be 1 to 2 spots away. The above only says "1" -- it looks like it can be 1-2 so lets just assume it can be 1-2 away.

If you shift 1 Left on a Q - A - Z you get a P L M -- so it will wrap on the same "Row" of your QWERTY keyboard.

If you shift 2 Left on a W - S - X you get P L M.

If you Shift 1 Right on P L M -- you get Q A Z. If you shift 2 right on O K N - you get Q A Z.

The shift is only on A-Z keys. We will ignore others.

enable1.txt has "si" has a valid word. Delete that word from the dictionary to make it work.

I will be double checking the challenge input - I will post an alternate one as well.

Title: [7/9/2014] Challenge #170 [Intermediate] Rummy Checker
Text: # [](#IntermediateIcon) _(Intermediate)_: Rummy Checker

[Rummy](http://en.wikipedia.org/wiki/Rummy) is another very common card game. This time, the aim of the game is to match cards together into groups (**melds**) in your hand. You continually swap cards until you have such melds, at which point if you have a valid hand you have won. Your hand contains 7 cards, and your hand will contain 2 melds - one that is 3 long and one that is 4 long. A meld is either:

* 3 or 4 cards of the same rank and different suit (eg. 3 jacks or 4 nines) called a **set**
* 3 or 4 cards in the same suit but increasing rank - eg. Ace, Two, Three, Four of Hearts, called a **run**

Ace is played low - ie. before 2 rather than after king.

Your challenge today is as follows. You will be given a Rummy hand of 7 cards. You will then be given another card, that you have the choice to pick up. The challenge is to tell whether picking up the card will win you the game or not - ie. whether picking it up will give you a winning hand. You will also need to state which card it is being replaced with.

## Input Description
First you will be given a comma separated list of 7 cards on one line, as so:

```
Two of Diamonds, Three of Diamonds, Four of Diamonds, Seven of Diamonds, Seven of Clubs, Seven of Hearts, Jack of Hearts
```

Next, you will be given another (**new**) card on a new line, like so:

```
Five of Diamonds
```

## Output Description

If replacing a card in your hand with the new card will give you a winning hand, print which card in your hand is being replaced to win, for example:

```
Swap the new card for the Jack of Hearts to win!
```

Because in that case, that would give you a run (Two, Three, Four, Five of Diamonds) and a set (Seven of Diamonds, Clubs and Hearts). In the event that picking up the new card will do nothing, print:

```
        No possible winning hand.
```

# Notes

You may want to re-use some code for your card and deck structure from your solution to [this challenge](http://www.reddit.com/r/dailyprogrammer/comments/29zut0) where appropriate.

Title: [7/16/2014] Challenge #171 [Intermediate] Zoom, Rotate, Invert Hex Picture
Text: #Description:

This builds off the [Easy #171]
(http://www.reddit.com/r/dailyprogrammer/comments/2ao99p/7142014_challenge_171_easy_hex_to_8x8_bitmap/) Challenge.
We take it to the next level.

We can read in an 8x8 picture from hex values. Once we have that image we can do some fun things to it.

* Zoom - zoom in or out of the image
* Rotate - turn the image 90 degrees clockwise or counter clockwise
* Invert - What was On is Off and what is Off becomes On. It inverts the image

Your challenge is implement these 3 abilities. If you completed Easy #171 then you have a headstart. Otherwise you will need to complete that first.

#Input:

Same as [Easy #171]
(http://www.reddit.com/r/dailyprogrammer/comments/2ao99p/7142014_challenge_171_easy_hex_to_8x8_bitmap/) read in 8 hex values and use it to generate a 8x8 image.

#Zoom:

You will zoom in x2 at a time. So let's look at what a zoom does.
You have this image (using numbers for reference)

```
    12
    34
```

If you perform a zoom in x2 you will generate this image.

```
    1122
    1122
    3344
    3344
```

If you zoom again on this image x2 you will get this:

```
    11112222
    11112222
    11112222
    11112222
    33334444
    33334444
    33334444
    33334444
```

So for example if you have this image:

```
    xxxxxxxx
    x      x
    x xxxx x
    x x  x x
    x x  x x
    x xxxx x
    x      x
    xxxxxxxx
```

If you do a zoom x2 you get this:

```
xxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx
xx            xx
xx            xx
xx  xxxxxxxx  xx
xx  xxxxxxxx  xx
xx  xx    xx  xx
xx  xx    xx  xx
xx  xx    xx  xx
xx  xx    xx  xx
xx  xxxxxxxx  xx
xx  xxxxxxxx  xx
xx            xx
xx            xx
xxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxx
```

Your zoom feature should be able to take the image and go x2. Up to a maximum of x4 (so 8x8 up to 32x32). Your zoom feature should also zoom out and take a 32x32 to a 16x16 and then down to a 8x8. Your zoom should not go out more than x4. (So your images can be only 8x8, 16x16 or 32x32).

#Rotate:

This is very simple. You will rotate clockwise or counterclockwise.

So this image:

```
12
34
```

If you rotate it 90 clockwise:

```
31
42
```

If you rotate it 90 counter clockwise:

```
12
34
```

Your rotations should go either direction and can handle the image being 8x8, 16x16 or 32x32.

#Invert:

In the image if it was turned off it becomes turned on. If it is turned on it becomes turn off.

Example if you have this image: (adding a border of #)

```
##########
#xxxxxxxx#
#x      x#
#x xxxx x#
#x x  x x#
```

```
    #x x  x x#
    #x xxxx x#
    #x      x#
    #xxxxxxxx#
    ##########
```

The invert of it becomes:

```
    ##########
    #        #
    # xxxxxx #
    # x    x #
    # x xx x #
    # x xx x #
    # x    x #
    # xxxxxx #
    #        #
    ##########
```

#Challenge:
Use the same input as the [Easy #171]
(http://www.reddit.com/r/dailyprogrammer/comments/2ao99p/7142014_challenge_171_easy_hex_to_8x8_bitmap/) and do the following operations on them.

* Zoom in x 2
* Rotate Clockwise 90
* Zoom in x 2
* Invert
* Zoom out x 2

Note: Due to the potential size of outputs (and if you elect to show the image inbetween the steps) please use a github or other method to show your output. Thanks!

For speed here are the 4 hex pictures from the Easy 171:

```
   FF 81 BD A5 A5 BD 81 FF
   AA 55 AA 55 AA 55 AA 55
   3E 7F FC F8 F8 FC 7F 3E
   93 93 93 F3 F3 93 93 93
```

Title: [7/25/2014] Challenge #172 [Intermediate] BREACH!
Text: #Description

This is the last time I hire monkeys to do my dirty work. Someone managed to break into our database and access all the data, I went in to inspect the problem and lo and behold, what do I see? Plaintext passwords!?

I hired some newer smarter guy who seemed to know what he was doing, I've spoken to my colleague who performed the code review on his program only to find out I've hired yet another monkey!

The password wasn't in plaintext, it was hashed, but an identical password brought back the same hash. How could I prevent this?

Maybe If I could get a unique hash for each user regardless of the password they enter that would solve the problem? Yes, that'll do...Damn monkeys...

#Formal Inputs & Outputs

##Input description

On standard console input you should enter a password of N length, it may contain any characters, numbers or punctuation.

##Output description

The output will be a reasonably secure hash of the password. The hash should be different even if two passwords are the same. For example

```
peanuts

A2F9CDDA934FD16E07833BD8B06AA77D52E26D39

peanuts

0E18F44C1FEC03EC4083422CB58BA6A09AC4FB2A
```

#Notes/Hints

For this exercise, feel free to use any hashing algorithm you like, built-in or not.

You should probably research into GUID's and how they are used to prevent identical password hashing mistakes.

Here is a good read on this exact topic:

[Password Hashing](http://www.codeproject.com/Articles/425150/Beginners-guide-to-a-secure-way-of-storing-passwor)

#Bonus
Create the hashing algorithm yourself rather than using a built-in SHA-1 etc...

Title: [7/23/2014] Challenge#172 [Intermediate] Image Rendering 101...010101000101
Text: #Description

You may have noticed from our [easy](http://www.reddit.com/r/dailyprogrammer/comments/2ba3g3/7212014_challenge_172_easy/) challenge that finding a program to render the PBM format is either very difficult or usually just a spammy program that no one would dare download.

Your mission today, given the knowledge you have gained from last weeks challenge is to create a Renderer for the PBM format.

For those who didn't do mondays challenge, here's a recap

* a PBM usually starts with 'P1' denoting that it is a .PBM file
* The next line consists of 2 integers representing the width and height of our image
* Finally, the pixel data. 0 is white and 1 is black.

This Wikipedia article will tell you more
http://en.wikipedia.org/wiki/Netpbm_format

#Formal Inputs & Outputs

##Input description
On standard console input you should be prompted to pass the .PBM file you have created from the easy challenge.

##Output description
The output will be a .PBM file rendered to the screen following the conventions where 0 is a white pixel, 1 is a black pixel

#Notes
This task is considerably harder in some languages. Some languages have large support for image handling (.NET and others) whilst some will require a bit more grunt work (C and even Python) .

It's up to you to decide the language, but easier alternatives probably do exist.

#Bonus
Create a renderer for the other versions of .PBM (P2 and P3) and output these to the screen.

Title: [7/30/2014] Challenge #173 [Intermediate] Advanced Langton's Ant
Text: # [](#IntermediateIcon) _(Intermediate)_: Advanced Langton's Ant

If you've done any work or research onto cellular automata, you may have heard of [Langton's Ant](http://en.wikipedia.org/wiki/Langton%27s_ant). It starts with a grid similar to that of [Conway's Game of Life](http://www.reddit.com/r/dailyprogrammer/comments/271xyp/) where a grid cell can be black or white, however this time we have an 'ant' on it. This little metaphorical ant will follow these four rules at every 'step':

* If the current square is white, turn the ant 90' clockwise
* If the current square is black, turn the ant 90' anticlockwise
* Flip the colour of the current square
* Move forward (from the ant's perspective) one cell

With the following starting conditions:

* All cells start white
* The ant starts pointing north

However, being /r/DailyProgrammer, we don't do things the easy way. Why only have 2 colours, black or white? Why not as many colours as you want, where you choose whether ant turns left or right at each colour? Today's challenge is to create an emulator for such a modifiable ant.

If you have more than 2 colours, of course, there is no way to just 'flip' the colour. Whenever the ant lands on a square, it is to change the colour of the current square to the next possible colour, going back to the first one at the end - eg. red, green, blue, red, green, blue, etc. In these cases, at the start of the simulation, all of the cells will start with the first colour/character.

## Input Description

You will be given one line of text consisting of the characters 'L' and 'R', such as:

```
LRLRR
```

This means that there are 5 possible colours (or characters, if you're drawing the grid ASCII style - choose the colours or characters yourself!) for this ant.

In this case, I could choose 5 colours to correspond to the LRLRR:

* White, turn left (anticlockwise)

* Black, turn right (clockwise)

* Red, turn left (anticlockwise)

* Green, turn right (clockwise)

* Blue, turn right (clockwise)

You could also choose characters, eg. `' '`, `'#'`, `'%'`, `'*'`, `'@'` instead of colours if you're ASCII-ing the grid. You will then be given another line of text with a number **N** on it - this is the number of 'steps' to simulate.

## Output Description

You have some flexibility here. The bare minimum would be to output the current grid ASCII style. You could also draw the grid to an image file, in which case you would have to choose colours rather than ASCII characters. I know there are some people who do these sorts of challenges with C/C++ curses or even more complex systems.

# Notes

[More info on Langton's Ant with multiple colours.](http://en.wikipedia.org/wiki/Langton%27s_ant#Extension_to_multiple_colors)

Title: [8/06/2014] Challenge #174 [Intermediate] Forum Avatar Generator
Text: #Description

You run a popular programming forum, Programming Daily, where programming challenges are posted and users are free to show off their solutions. Three of your most prolific users happen to have very similar handles: Sarlik, Sarlek, and Sarlak. Following a discussion between these three users can be incredibly confusing and everyone mixes them up.

The community decides that the best solution is to allow users to provide square avatars to identify themselves. Plus the folks over at the competing /r/dailyprogrammer forum don't have this feature, so perhaps you can use this to woo over some of their userbase. However, Sarlik, Sarlek, and Sarlak are totally old school. They each browse the forum through an old text-only terminal with a terminal browser (lynx, links). They don't care about avatars, so they never upload any.

After sleeping on the problem you get a bright idea: you'll write a little program to procedurally generate an avatar for them, and any other stubborn users. To keep the database as simple as possible, you decide to generate these on the fly. That is, given a particular username, you should always generate the same avatar image.

#Formal Input Description

Your forum's usernames follow the same rules as reddit's usernames (e.g. no spaces, etc.). Your program will receive a single reddit-style username as input.

#Formal Output Description

Your program outputs an avatar, preferably in color, with a unique pattern for that username. The output must always be the same for that username. You could just generate a totally random block of data, but you should try to make it interesting while still being reasonably unique.

##Sample Inputs

Sarlik
Sarlek
Sarlak

##Sample Outputs

    http://i.imgur.com/9KpGEwO.png
    http://i.imgur.com/IR8zxaI.png

```
http://i.imgur.com/xf6h0Br.png
```

#Challenge Input

Show us the avatar for your own reddit username.

Title: [8/13/2014] Challenge #175 [Intermediate] Largest Word from Characters
Text: #Description:

Given a string of words and a string of letters. Find the largest string(s) that are in the 1st string of words that can be formed from the letters in the 2nd string.

* Letters can be only used once. So if the string has "a b c" then words like "aaa" and "bbb" do not work because there is only 1 "a" or "b" to be used.
* If you have tie for the longest strings then output all the possible strings.
* If you find no words at all then output "No Words Found"

#input:

```
(String of words)
(String of characters)
```

##example:

```
abc cca aaaaaa bca
a b c
```

#output:

List of max size words in the first string of words. If none are found "No Words Found" displayed.

##example (using above input):

```
abc bca
```

#Challenge input 1:

```
hello yyyyyyy yzyzyzyzyzyz mellow well yo kellow lellow abcdefhijkl hi is yellow just here to add strings fellow lellow llleow
l e l o h m f y z a b w
```

#Challenge input 2:

```
sad das day mad den foot ball down touch pass play
z a d f o n
```

Title: [8/27/2014] Challenge #177 [Intermediate] .- ..- -.. .. ---
Text: #Description

Morse code is an aural method of transmitting text through the use of silence and tones.

Todays challenge will involve translating your standard english text into morse code, and from there, into an audio file.

##Example

Step 1: "I like cats" - The phrase entered by you for translating

Step 2: .. / .-.. .. -.- . / -.-. .- - ... - The output of the phrase in step 1

Step 3: cats.wav - An audio file containing each dot(.) and dash(-) as an audible tone

#Formal Inputs & Outputs

##Input description

On standard console input, you should enter a phrase of your choosing. This will then be parsed into morse code and finally outputted as stated in the output description.

##Output description

The program should output a valid audio file (WAV, MP3, OGG, as long as it can play it's fine). In that audio should be an audio translation of your input.

Title: [9/03/2014] Challenge #178 [Intermediate] Jumping through Hyperspace ain't like dusting Crops
Text: #Description:

You are navigator aboard the Space Pirate Bob's spaceship the Centennial Condor. Operation of the spaceship requires fuel. Bob wants to calculate a round trip to the deepest planet from his given amount of fuel he is willing to buy for a smuggling run to earn some space credits.

As navigator you need to compute the deepest planet you can make a jump to and back. Space Pirate Bob was too cheap to buy the Mark 2 spaceship navigation package for you. So you will have to improvise and code your own program to solve his problem.

Oh and by the way, the Space Pirate does not like to brack track on his routes. So the jump route to the planet cannot be the same one you take back (The Federation of Good Guy Planets will be patrolling the route you take to the planet to smuggle goods to catch you)

Good Luck, may the Code be with you.

#Star Map:

You will be given a star map in the series of planet letters and fuel cost. If you take the jump route (in any direction) between these planets your spaceship will expend that many units of full. The star map has you start off on Planet A. You will need to see how far from A you can get given your below input of fuel.

The star map has the follow pairs of planets with a jump route between them and the number represents how much fuel you spend if you use it.

```
    A B 1
    A C 1
    B C 2
```

```
   B D 2
   C D 1
   C E 2
   D E 2
   D F 2
   D G 1
   E G 1
   E H 1
   F I 4
   F G 3
   G J 2
   G H 3
   H K 3
   I J 2
   I K 2
```

#input:
A value N that represents how many units the Space Pirate Bob is willing to spend his space credits on to fuel the Centennial Condor for its smuggling run.

##Example:
5

#Output:
The deepest route from A to a planet and back not using the same jump route (planets could be duplicated but the route back has to be unique as the one you use to get to the destination is patrolled) Display the planet and then the To route and Back route.

If no route is found - print an error message. If there is a tie, have your program decide which one to show (only 1 is needed not all)

##example (using the input of 5 above):

```
   Planet D
   To: A-C-D
   Back: D-B-A
```

#Challenge Inputs:

Look for routes for these fuel amounts:

* 5
* 8
* 16

Title: [9/10/2014] Challenge #179 [Intermediate] Roguelike - The traveller Game
Text: #Description:

So I was fooling around once with an idea to make a fun Rogue like game.
If you do not know what a Rogue Like is check out [Wikipedia Article] (http://en.wikipedia.org/wiki/Roguelike) on what it is about.

I got this really weak start at just trying to generate a more graphical approach than ASCII text. If you want to see my attempt. Check out my incomplete project [FORGE] (http://coderd00d.com/Forge/index.html)

For this challenge you will have to develop a character moving in a rogue like environment. So the design requirements.

* 1 Hero character who moves up/down/left/right in a box map.
* Map must have boundary elements to contain it -- Walls/Water/Moutains/Whatever you come up with

* Hero does not have to be a person. Could be a spaceship/sea creature/whatever - Just has to move up/down/left/right on a 2-D map
* Map has to be 20x20. The boundary are some element which prevents passage like a wall, water or blackholes. Whatever fits your theme.
* Your hero has 100 movement points. Each time they move up/down/left/right they lose 1 movement points. When they reach 0 movement points the game ends.
* Random elements are generated in the room. Gold. Treasure. Plants. Towns. Caves. Whatever. When the hero reaches that point they score a point. You must have 100 random elements.
* At the end of the game when your hero is out of movement. The score is based on how many elements you are able to move to. The higher the score the better.
* Hero starts either in a fixed room spot or random spot. I leave it to you to decide.

#input:

Some keyboard/other method for moving a hero up/down/left/right and way to end the game like Q or Esc or whatever.

#output:

The 20x20 map with the hero updating if you can with moves. Show how much movement points you have and score.

At the end of the game show some final score box. Good luck and have fun.

#Example:

ASCII Map might look like this. (This is not 20x20 but yours will be 20x20)

* % = Wall
* $ = Random element
* @ = the hero

A simple dungeon.

```
%%%%%%%%%%
%..$.....%
%......$.%
%...@....%
%....$...%
%.$......%
%%%%%%%%%%
Move: 100
Score: 0
```

#Creative Challenge:

This is a creative challenge. You can use ASCII graphics or bmp graphics or more. You can add more elements to this. But regardless have fun trying to make this challenge work for you.

Title: [9/17/2014] Challenge #180 [Intermediate] Tamagotchi emulator
Text: #Description

You're lonely and bored. Your doritos are stale and no one is online, this loneliness you feel has a cure...A TAMAGOTCHI

For those of you who have never heard of a Tamagotchi, here's a quick summary:

A tamagotchi is a virtual pet whose life you must sustain through various activities including eating, playing, making it sleep, and cleaning its poop. Tamagotchi's go through several life cycles, most notably, egg/infant, teen, adult, elderly. Tamagotchi's can die from lack of attention (in the classic ones, half a day of neglect would kill it) and also from age.

For more information check the wiki

http://en.wikipedia.org/wiki/Tamagotchi

Your job is to create a tamagotchi via command line, gui or any other avenue you'd like.

#Requirements

The tamagotchi must have at **least** the following requirements:

* Capable of being fed
* Capable of being put to bed
* Capable of going to sleep on its own, losing health from hunger and pooping on its own without prompting
* Capable of aging from birth through to death

Like I said, these are the bare minimum requirements, feel free to get quirky and add weird stuff like diseases and love interests.

Title: [09/24/2014] Challenge #181 [Intermediate] Average Speed Cameras
Text: # [](#IntermediateIcon) _(Intermediate)_: Average Speed Cameras

In the UK, a common safety measure on motorways is the so-called [average speed cameras](http://en.wikipedia.org/wiki/SPECS_%28speed_camera%29). These, unlike normal speed cameras which measure a vehicle's speed instantaneously, have several connected cameras at intervals along a motorway. The speed of a vehicle can be determined by dividing the distance between two cameras by the time it takes the vehicle to get from one to another. This can be used to stop vehicles breaking the speed limit over long stretches of roads, rather than allowing vehicles to speed up after they are out of range. The Home Office has contacted you to replace the aging software system in the cameras with something more up to date.

In this challenge, you will be given a number of speed cameras and their positions along a road, along with the speed limit. You will then be given the camera logs for each camera in turn. From this data, you will work out which vehicles are breaking the speed limit.

# Formal Inputs and Outputs

## Input Description

The first section of the input will contain the speed limit and the position of the speed cameras. The speed limit may be in miles per hour or kilometres per hour. The lines will be in the format:

Speed limit is <limit> mph.

OR

Speed limit is <limit> km/h.

The lines describing the positions of the speed cameras will look like:

Speed camera <number> is <distance> metres down the motorway.

Speed camera number 1 will always have a distance of 0.

After this, you will get logs for each speed camera, like this:

```
Start of log for camera <number>:
Vehicle <registration number> passed camera <number> at <time>.
Vehicle <registration number> passed camera <number> at <time>.
...
```

Example inputs and outputs can be found below.

## Output Description

For each vehicle that breaks the speed limit, print a line like so:

```
Vehicle <registration number> broke the speed limit by <amount>.
```

Where `<amount>` is in the local units.

# Sample Inputs and Outputs

## Sample Input

```
Speed limit is 60.00 mph.
Speed camera number 1 is 0 metres down the motorway.
Speed camera number 2 is 600 metres down the motorway.
Speed camera number 3 is 855 metres down the motorway.
Speed camera number 4 is 1355 metres down the motorway.
Start of log for camera 1.
Vehicle G122 IVL passed camera 1 at 09:36:12.
Vehicle H151 KEE passed camera 1 at 09:36:15.
Vehicle U109 FIJ passed camera 1 at 09:36:20.
Vehicle LO04 CHZ passed camera 1 at 09:36:23.
Vehicle I105 AEV passed camera 1 at 09:36:28.
Vehicle J828 EBC passed camera 1 at 09:36:29.
Vehicle WF EP7 passed camera 1 at 09:36:32.
Vehicle H108 KYL passed camera 1 at 09:36:33.
Vehicle R815 FII passed camera 1 at 09:36:34.
Vehicle QW04 SQU passed camera 1 at 09:36:34.
Start of log for camera 2.
Vehicle G122 IVL passed camera 2 at 09:36:42.
Vehicle LO04 CHZ passed camera 2 at 09:36:46.
Vehicle H151 KEE passed camera 2 at 09:36:51.
Vehicle QW04 SQU passed camera 2 at 09:36:53.
Vehicle J828 EBC passed camera 2 at 09:36:53.
Vehicle R815 FII passed camera 2 at 09:36:55.
Vehicle U109 FIJ passed camera 2 at 09:36:56.
Vehicle H108 KYL passed camera 2 at 09:36:57.
Vehicle I105 AEV passed camera 2 at 09:37:05.
Vehicle WF EP7 passed camera 2 at 09:37:10.
Start of log for camera 3.
Vehicle LO04 CHZ passed camera 3 at 09:36:55.
Vehicle G122 IVL passed camera 3 at 09:36:56.
Vehicle H151 KEE passed camera 3 at 09:37:03.
Vehicle QW04 SQU passed camera 3 at 09:37:03.
Vehicle J828 EBC passed camera 3 at 09:37:04.
Vehicle R815 FII passed camera 3 at 09:37:09.
Vehicle U109 FIJ passed camera 3 at 09:37:11.
Vehicle H108 KYL passed camera 3 at 09:37:12.
Vehicle I105 AEV passed camera 3 at 09:37:20.
```

```
        Vehicle WF EP7 passed camera 3 at 09:37:23.
        Start of log for camera 4.
        Vehicle LO04 CHZ passed camera 4 at 09:37:13.
        Vehicle QW04 SQU passed camera 4 at 09:37:24.
        Vehicle J828 EBC passed camera 4 at 09:37:26.
        Vehicle G122 IVL passed camera 4 at 09:37:28.
        Vehicle R815 FII passed camera 4 at 09:37:28.
        Vehicle H151 KEE passed camera 4 at 09:37:29.
        Vehicle H108 KYL passed camera 4 at 09:37:36.
        Vehicle I105 AEV passed camera 4 at 09:37:42.
        Vehicle WF EP7 passed camera 4 at 09:37:44.
        Vehicle U109 FIJ passed camera 4 at 09:37:45.
```

## Sample Output

```
    Vehicle LO04 CHZ broke the speed limit by 3.4 mph.
    Vehicle LO04 CHZ broke the speed limit by 2.1 mph.
    Vehicle QW04 SQU broke the speed limit by 10.6 mph.
    Vehicle R815 FII broke the speed limit by 3.9 mph.
```

# Challenge

## Challenge Input

A long pastebin containing a huge data set is [available here](https://gist.githubusercontent.com/Quackmatic/e75d61c1ecc319f721a2/raw/average-speed-cameras.txt), to stress-test your input if nothing else.

# Notes

You may want to use regular expressions again for this challenge.

Title: [10/01/2014] Challenge #182 [Intermediate] The Data Collator from Jamaica
Text: # [](/IntermediateIcon) **(Intermediate)**: The Data Collator from Jamaica

Often, when given a set of data where one variable is associated with another, we want to find a general rule equating the two variables, with which you can find the closest appropriate match of one to the other.

Say, for example, we have performed an experiment determining the acceleration undergone by an object when subject to a force. [Newton's 2^nd Law of Motion](https://en.wikipedia.org/wiki/Newton%27s_laws_of_motion#Newton.27s_second_law) dictates that *F=ma* - linking the variables `F` (force) and `a` (acceleration) by a constant `m` (mass of the object). If we performed the acceleration we may get the following values:

F (N)	a (m s^(-2))
0.2	0.32
0.4	0.62
0.6	0.97
0.8	1.22
1	1.58
1.2	1.84
1.4	2.17
1.6	2.47
1.8	2.83
2	3.16

[This data can be plotted to see the link between the 2 data sets. Here, `F` is on the horizontal and `a` is on the vertical axis.](http://i.imgur.com/PdlTl6W.png)

To create a line of best-fit or **trend line** for this data, [which looks like this](http://i.imgur.com/W2aFGIx.png), a number of methods can be used, such as the ever-present [least squares](https://en.wikipedia.org/wiki/Least_squares) method. For the purposes of this challenge, the trend line will *always* be linear, and thus the two data sets must be

Your challenge is, given 2 data sets, draw the values on an appropriately-scaled graph (with axes) and find a suitable trend line fitting the data.

# Input and Output Description

## Input

The first line of input will be in the format:

```
<X>:<graph title>:<X label>:<Y label>
```

* **X**: The size of the data sets.
* **graph title**: The title to be displayed at the top of the graph.
* **X label**: The label to be displayed on the x-axis.
* **Y label**: The label to be displayed on the y-axis.

Following that will be precisely **N** further lines of input, in the format:

```
X:Y
```

Where X is the value to be plotted on the X-axis, and Y is the value to be plotted on the Y-axis.

# Output

The output is to be in the form of an image:

* The scale of the axes should be big enough to show every data point on the image, but not too big such that the points are all crammed together.
* The data points are to be plotted onto a graph.
* A linear trend line, fitting the given data, is to be plotted.

# Sample Input

I've created a data set for you to plot yourself.

```
20:Graph of I over V through a resistor:Voltage (V):Current (mA)
0.000:0.000
0.198:0.387
0.400:0.781
0.600:1.172
0.802:1.566
1.003:1.962
1.200:2.349
1.402:2.735
1.597:3.122
1.798:3.505
2.002:3.918
2.202:4.314
2.399:4.681
2.603:5.074
2.800:5.485
```

```
2.997:5.864
3.198:6.256
3.400:6.631
3.597:7.017
3.801:7.435
```

# Tips
Here are some tips to make the most of this /r/DailyProgrammer challenge.

* Try and think of an algorithm or method to find the best-fit line yourself. There are plenty of ways out there, but as a member of /r/DailyProgrammer try and do it from scratch!

* Half of the challenge here is drawing the graph yourself. For that reason it's best to pick a language here that supports graphical output. Using a premade graphing library defeats the point of this challenge so try and DIY.

Title: [10/08/2014] Challenge #183 [Intermediate] Edge Matching Tile Puzzle
Text: #Credit:

Thanks to /u/skeeto for this challenge. As posted on our /r/dailyprogrammer_ideas subreddit.

#Description:

There's a tile puzzle game you might find at your local game store. There are 9 tiles to be arranged in a 3x3 grid. Each of a tile's contains half of some image, to be met up with the appropriate half on another tile. The images are usually animals (cats, beetles). There are 4 kinds of images in total. For example, here's a picture of completed puzzle.

* (http://i.imgur.com/NbLum43.jpg)

Your task is to write a program that finds solutions to a given set of tiles.

#Formal Input Description:

On standard input you'll be given a number, n, indicating the size of the side of the puzzle. For example, for a 3x3 puzzle n = 3. What will follow are n * n lines of 4 letters indicating the edges of each tile. The order of the edges is north, east, south, west (clockwise). Your program should be able to handle up to n = 5.
Instead of images, we'll use the 4 colors Cyan, Magenta, Yellow, and Black (CMYK). The two "halves" are uppercase and lower case. For two tiles to legally touch, an uppercase letter can only touch its lowercase matchin letter on an adjacent tile and vice versa.
For the sake of communication, [the tiles will be labeled A-Z] (http://i.imgur.com/pnVvQ8L.jpg) in the order that they were input. So on a 3x3 puzzle, the tiles are A-I.

#Formal Output Description:

This is where you can get creative. The simplest output could just list the tiles, left to right, top to bottom, and their orientations (N, E, S, W). Or if you're feeling ambitious, output an image showing the completed tile arrangement. For a 3x3 puzzle, there are over 95 billion possible such arrangements (9! * 4^9), though all but a handful of them will be illegal.

You may output just one solution or all solutions. Keep symmetry in mind.

#Sample Input 1

```
3
CYMk
CmKm
cKyM
cYkY
```

```
CMky
ckyM
CYMK
CMKy
CkmY
```

This corresponds to these tiles:

* (http://i.imgur.com/eok9gTt.png)

With these graphics, half circles must be matched up with half squares of the same color. The solution should look like those [cannon bullet things from Super Mario.] (http://i.imgur.com/etCrWXi.jpg)

#Sample Input 2

```
3
ycKC
kKcY
cKMc
mYmk
CCYk
mMyC
MyYk
mKMy
YCMk
```

#Sample Output 1

Simplest output showing one solution:

```
AN CW GE BW FE DS HE IS EN
```

A more graphical output (same solution):

```
+---------+
| C  M  Y |
|kAYyCcCGM|
| M  K  K |
| m  k  k |
|KBCcFyYDY|
| m  M  c |
| M  m  C |
|CHKkIYyEM|
| y  C  k |
+---------+
```

Or drawing the solution:

* (http://i.imgur.com/GJh2eOI.png)

#Challenge Input #1:

```
4
mcYC
MmCk
yYcm
yMYC
Ykcy
```

```
    kkkm
    KKcy
    KMYK
    YMkk
    ymKc
    MyMK
    CmmY
    kMMY
    yCCM
    yccc
    kcck
```

Graphical version (if this helps):

* (http://i.imgur.com/mpO8HGJ.png)

#Challenge Input #2:

```
    5
    cKCk
    yYcc
    YcCK
    kKCM
    CMKc
    cKYC
    kYcm
    KYyY
    Mccm
    yKcm
    mykK
    MMCm
    ckYC
    ycmm
    MmKM
    kymc
    KMMK
    KcyM
    kYck
    YCKM
    myYm
    kYyY
    CMKM
    yYCM
    YKyk
```

Graphical version:

* (http://i.imgur.com/Msa9F6G.png)

Title: [10/15/2014] Challenge #184 [Intermediate] Radioactive Decay
Text: # [](#IntermediateIcon) **(Intermediate)**: Radioactive Decay

Radioactive decay occurs when an unstable atomic nucleus tries to make itself become more stable. It does this by spitting bits of itself out - like [taking bits off your car to make it lighter](https://www.youtube.com/watch?v=OETj9aTYO2Q). While radioactive decay is an entirely random process, the probability of one type of nucleus decaying per second is well-defined.

There are two ways of describing this probability. The first is using a constant called λ (lambda). λ describes the probability of a specific type of atomic nucleus decaying per second. If λ=0, the probability is 0 - meaning the nucleus never decays, and is thus stable. If λ=0.5, every second there is a 50% chance the nucleus will decay. You get the point.

The second is using a value called t½ (the half-life). This describes how long it takes for exactly half of a sample to decay. For example, if the half-life of a nucleus is 10 seconds, and you start with a sample of 2000 nuclei:

* At the start, 2000 atoms remain - nothing has happened yet.
* After 10 seconds, half of the 2000 atoms will have decayed. 1000 remain.
* After 20 seconds, half of those 1000 will have decayed. 500 remain.
* After 30 seconds, half of those 500 will have decayed. 250 remain.

And so on, and so forth. This is all very simple - until you introduce the concept of a *decay chain*. This describes how a starting nucleus will decay into a 'daughter' nucleus, which in turn decays again into another type of nucleus - this happens all the way down the chain until the nucleus becomes stable.

Your challenge is: given a decay chain and λ values for each type of nucleus in the chain, calculate the percentage of each nucleus in the sample over time. This can be done by random sampling (the simpler way), or calculation of the exponential decays (for mathematicians). You can choose which method to do it by.

# Trouble understanding the concept?

This challenge introduces a physics concept which may be new to you - don't worry if you have some issues understand what's going on. Imagine you have a bag of 400 marbles. These marbles can be different colours - at the start of the experiment the marbles are all red.

Let's say a red marble has a 10% chance of turning into a green one per second. A green marble has a 50% chance of turning into blue marble every second. This means that (roughly) 40 red marbles will have turned into green marbles after 1 second. We now have 360 red and 40 green.

Now, you have 360 red marbles and 40 green ones. Remember the green marbles have a 50% chance of turning into blue marbles. After another second (2 seconds in total), 50% of the green marbles turn into blue marbles, that is 20 of them. 10% of the remaining red marbles will turn green, that is 36 of them.

We now have 324 (400*90%*90%) red, 56 (the 20 that remain, and then the other 36 that just decayed) green and 20 blue. This carries on happening. Of course these are approximations, as we don't count any marbles that happen to change twice in one second. If the percentages of changing are low enough, however, this is negligible for our simulation.

Now, replace marble colour with 'type of nucleus' (aka. isotope), marble with nucleus (ie. an instance of a type of nucleus), and bag with sample. These percentages are the λ values - so the λ of the red marble, 10%, is 0.1.

# Formal Inputs and Outputs

## Input Description

You will be first given a value **t**. This is the number of seconds to run your calculations for - ie. calculate the percentages of the nuclei at this time.

You will be then given a line in the format:

```
    a->b->c->...->z
```

Where `a`...`z` are types of nucleus. `a` is the starting nucleus type of your sample, and `z` is the end of the chain; the stable nucleus.

You will then be given lines in the format:

```
    a: 0.0002
```

Where `a` is an unstable type of nucleus, and `0.0002` is the λ constant for `a`. The last one in the decay chain must have a λ of zero (stable).

## Output Description

You will print a line for all nuclei in the decay chain in the format:

```
    a: 3.4%
```

Where `a` is a type of nucleus in the decay chain, and `3.4%` is the percentage of the sample that is nucleus type `a` after **t** seconds.

# Sample Inputs and Outputs

## Sample Input

```
    5000
    a->b->c->d->s
    a: 0.00007
    b: 0.0005
    c: 0.00013
    d: 0.00022
    s: 0
```

## Sample Output 1

```
    a: 70.37%
    b: 10.25%
    c: 15.00%
    d: 3.31%
    s: 1.07%
```

## Sample Output 2

```
    a: 70.76%
    b: 11.00%
    c: 14.48%
    d: 2.80%
    s: 0.96%
```

I'm using a random method so my results differ slightly (but are relatively consistent) each time. With a larger sample size your results will be more accurate. YMMV!

# Extension

Plot the data as percentages over time! For the given input above, my test program gives [this output](http://i.imgur.com/evZO02O.png) for t=5000 s and [this output](http://i.imgur.com/FtmQzsb.png) for t=50000 s, where cooling colours are going down the decay chain. Make yours look really cool for extra points!

Title: [10/23/2014] Challenge #185 [Intermediate] Syntax Highlighting
Text: # [](#IntermediateIcon) **(Intermediate)**: Syntax Highlighting

(sorry for the delay, an unexpected situation arose yesterday which meant the challenge could not be written.)

Nearly every developer has came into contact with syntax highlighting before. Most modern IDEs support it to some degree, and even some text editors such as Notepad++ and gedit support it too. Syntax highlighting is what turns this:

```
using System;

public static class Program
{
   public static void Main(params string[] args)
   {
      Console.WriteLine("hello, world!");
   }
}
```

into [something like this](http://i.imgur.com/DhfeU8D.png). It's very useful and can be applied to almost every programming language, and even some markup languages such as HTML. Your challenge today is to pick any programming language you like and write a converter for it, which will convert source code of the language of your choice to a highlighted format. You have some freedom in that regard.

# Formal Inputs and Outputs

## Input Description

The program is to accept a source code file in the language of choice.

## Output Description

You are to output some format which allows formatted text display. Here are some examples for you to choose.

* You could choose to make your program output HTML/CSS to highlight the syntax. For example, a highlighted keyword `static` could be output as `<span class="syntax-keyword">static</span>` where the CSS `.syntax-keyword` selector makes the keyword bold or in a distinctive colour.
* You could output an image with the text in it, coloured and styled however you like.
* You could use a library such as `ncurses` (or another way, such as `Console.ForegroundColor` for .NET developers) to output coloured text to the terminal directly, siimlar to the style of complex editors such as vim and Emacs.

# Sample Inputs and Outputs

The exact input is up to you. If you're feeling meta, you could test your solution using... your solution. If the program can highlight its own source code, that's brilliant! Of course, this assumes that you write your solution to highlight the language it was written in. If you don't, don't worry - you can write a highlighter for Python in C# if you wish, or for C in Ruby, for example.

# Extension (Easy)

Write an extension to your solution which allows you to toggle on and off the printing of comments, so that when it is disabled, comments are omitted from the output of the solution.

# Extension (Hard)

If your method of output supports it, allow the collapsing of code blocks. [Here is an example in Visual Studio](http://gfycat.com/DefensiveLimpDore). You could achieve this using JavaScript if you output to HTML.

Title: [10/24/2014] Challenge #185 [Intermediate to Hard] Roots of a Polynomial
Text: # [](#HardIcon) **(Intermediate to Hard)**: Roots of a Polynomial

In mathematics, a polynomial is a form of expression. The type of polynomials we're dealing with today are called *univariate* polynomials, which means they only have one variable. For this challenge, this variable will be called `x`. You'll need to dig out your algebra textbooks if you're a bit rusty, though this challenge doesn't require you to use anything more than high school (A-level) mathematics.

The simplest type of polynomial is this:

```
    4
```

Fairly simple, right? Right. A constant value such as `4`, `0` or `-0.2` are polynomials of degree zero. The next simplest type looks like this:

```
    4x+3
```

The equation for a straight-line graph is a polynomial of degree one. Again, fairly simple to work with. The good thing about polynomials is that we can visualise them using graphs. [Here is the graph for `y=4x+3`, the polynomial above](https://www.desmos.com/calculator/llczd44a8i). The next simplest is the quadratic equation, otherwise known as a polynomial of degree two (notice the pattern yet?). These are similar to linear equations, but they feature a multiple of x squared bolted onto the front. [Here is the graph of `y=x^2-6x+3`](https://www.desmos.com/calculator/nmzbjtiqmf), [and here is the graph of `y=(-1/3)x^2-x+8`](https://www.desmos.com/calculator/2vjpjxxgwp).

The cool thing about quadratics is that you can create them by multiplying together two linear polynomials. For example, `(3x-1)(x+7)` is the same as `3x^2+20x-7`, [as you can see here](https://www.desmos.com/calculator/si0svfjmcj). If we take a look at the graph of `y=3x-1`, `y=x+7` and `y=3x^2+20x-7` we notice something interesting. [Here you can see](https://www.desmos.com/calculator/maw5tkik1p) the quadratic graph crosses the x-axis at the same point as where the linear graphs do. The point where a polynomial crosses the x=axis are called its *roots* - which is what we will be finding in today's challenge.

You can also do the reverse operation - given an equation, find its roots. For a linear equation, this is simple. A bit of algebraic jiggery-pokery gives us these steps. Remember, the graph will cross the x-axis where the height (y) is at zero, so we need to set `y=0`.

```
    y=ax+b and y=0
    0=ax+b (replace the y in the first equation with 0, as y=0)
    -b=ax (subtract b from both sides)
    -b/a=x (divide both sides by a)
```

Therefore, we can see that if we have a linear equation `y=ax+b`, it crosses the x=axis at the point where its x value is `-b/a`. The same can be done for quadratic polynomials via a few methods, including using the [quadratic formula](https://en.wikipedia.org/wiki/Quadratic_formula) or [completing the square](https://en.wikipedia.org/wiki/Completing_the_square). If all else fails you can just draw the graph of the expression to approximate its roots.

What happens when the plotted graph never crosses the x-axis? Simply, it has *no* roots (or no real roots). If you attempt to use the quadratic formula on an equation such as `x^2+x+4` you will end up square-rooting a negative number, which we ignore for today's challenge.

Things get a little awkward when you have 3rd-degree polynomials and above. They act the same and are treated the same as other polynomials but there is no simple formula to find the roots. The Babylonians could find the roots of quadratic polynomials, but it took mathematicians until the Renaissance to find a one-step formula to get the roots of a cubic polynomial.

Rather than bothering with the convoluted cubic formula you can instead use what are known as numerical methods. These methods are approximation methods, and rather than giving you an exact answer immediately they 'home in' on the roots like a heat-seeking missile. The benefits of these are that they can be used to find roots of almost any mathematical function, not only polynomils. They can also be used to find roots of very complex polynomials, where a one-step equation would be huge and ugly. The downsides are that they can often be slow to find the answer, they can only give you one root at a time and, sometimes, they never even find the root at all! There are several numerical methods to find polynomial roots, the most commonly used are these:

* [Interval Bisection method](https://en.wikipedia.org/wiki/Bisection_method). This is a simple to understand, divide-and-conquer algorithm.
* [Newton-Raphson method](https://en.wikipedia.org/wiki/Newton%27s_method). More complex to understand but quicker at finding a root.
* [Other root finding algorithms](https://en.wikipedia.org/wiki/Root-finding_algorithm#Interpolation_2) such as linear interpolation are also easy to understand.

Your challenge is, given a polynomial expression of degree no higher than 7, find a root (if it exists) of the expression where it crosses the x-axis (equal to zero.)

# Formal Inputs and Outputs

## Input Description

You will accept a polynomial in the form used in this challenge description. That is:

* `x` denotes the variable.
* `^...` denotes the exponent of a term.
* A constant denotes the coefficient of a term.

A valid input would be `x^3-5x^2+10x-44` or `-4x^5-7`, but not `2^x+3` (not a polynomial), `x^2+2xy+y^2` (more than one variable) or `x^11-6x^2-1` (no higher than 7th degree allowed; that is 11th degree).

## Output Description

You are to output a root of the polynomial as a number (or an algebraic expression.. if you're crazy!)

# Sample Inputs and Outputs

Here are some examples to get you going. You can create your own by [typing them in on Wolfram|Alpha](http://www.wolframalpha.com/input/?i=x^2-7x%2B6), which also plots it and tells you the roots, if any.

## Sample Inputs

1. `4x^2-11x-3`
2. `4x-8`
3. `x^4-2x^3+7x^2-16x+4`
4. `x^2-7x+6`

## Sample Outputs

1. `-0.25` or `3`
2. `2`
3. `2` or `0.2825..`
4. `1` or `6`

# Extension (Hard)

You've found one root of the polynomial - now modify your solution to find *all* of the roots. This will require a divide-and-conquer algorithm of some sort.

Title: [10/29/2014] Challenge #186 [Intermediate] Syzygyfication
Text: # [](#IntermediateIcon) **(Intermediate)**: Syzygyfication

In astronomical terms, a *syzygy* is when 3 or more objects line up in a straight line. The classic example of this is an *eclipse* (not the IDE, thankfully.) If the Sun, the Moon and the Earth (in that order) line up in a straight line, then the Moon is directly in-between the Sun and the Earth, meaning the view of the Sun is occluded - a solar eclipse. Another example of a syzygy is a *transit*. This is like an eclipse, but when a planet goes in front of the sun instead; for example, [in this image](http://upload.wikimedia.org/wikipedia/commons/6/64/Mercury_transit_2.jpg), the big yellow disc is (predictably) the Sun and the circular black spot in the middle is Mercury. It's like a mini-eclipse. Besides these two examples, syzygy can occur without the Sun. The dots in [this image here](https://upload.wikimedia.org/wikipedia/commons/c/c2/Three_Planets_Dance_Over_La_Silla.jpg) are the planets Mercury, Venus and Jupiter. They do not form a perfect syzygy - the chance of that occurring is next to nothing - but they line up close enough that they're within a few degrees of each other in the sky.

The Wikipedia page for syzygy is here:
[en.wikipedia.org/wiki/Syzygy_(astronomy)](https://en.wikipedia.org/wiki/Syzygy_%28astronomy%29)

Today, you'll have two challenges. The first one is to pronounce **syzygyfication**. The second one will be to determine if a syzygy is occurring at a given time, for a given solar system.

## Simplification

This challenge as stated would require a load of mathematics to solve. For this programming challenge, we will assume that the planets orbit the Sun in perfect circles on the same plane, that the Sun does not move at all, and the planets all start off with zero degrees rotation (ie. all in syzygy with each other.)

# Formal Inputs and Outputs

## Required Data

You will need this data of the Solar system. An AU (astronomical unit) is the distance from the Earth to the Sun. The orbital period is the time it takes for the planet to complete its orbit; a value of eg. `2` means the planet completes an orbit around the Sun every 2 years.

Object	Orbit Radius (AU)	Orbital Period (Earth year)
Sun	0.000	*n/a*
Mercury	0.387	0.241
Venus	0.723	0.615
Earth	1.000	1.000
Mars	1.524	1.881
Jupiter	5.204	11.862
Saturn	9.582	29.457
Uranus	19.189	84.017
Neptune	30.071	164.795

## Input Description

You are to accept a number, which is a number of years after the starting time.

## Output Description

You are to output which of the planets, **or the Sun**, are in syzygy at the given time (in no particular order). For example:

```
Venus-Sun-Earth syzygy occurring.
```

A syzygy should be when the objects are within 1 degree of each other in the sky. Remember, syzygy can also occur when the Sun is in-between the two objects. In this case, this is called 'opposition'.

# Sample Inputs and Outputs

An example 4-syzygy occurs at 3.30085 years, where Mercury, Earth, Mars and Jupiter line up. [A visual example of this is here](http://i.imgur.com/gkD863d.png). Some more syzygy occurrences are:

Time (Earth year)	Syzygy
3.30085	Mercury-Earth-Mars-Jupiter
9.12162	Sun-Mercury-Mars, Mercury-Venus-Saturn
18.0852	Sun-Mars-Saturn, Mercury-Earth-Saturn-Neptune

31.0531	Sun-Earth-Saturn, Venus-Earth-Mars
40.2048	Sun-Venus-Mars, Mercury-Mars-Saturn, Earth-Mars-Uranus
66.2900	Sun-Venus-Earth-Uranus

# Extension

If your programming language supports it, draw a view of the Solar system at the given time, to show the objects in syzygy (like the image above.)

Title: [11/05/2014] Challenge #187 [Intermediate] Finding Time to Reddit
Text: #Description:

I cover the border of my monitor with post it notes with tasks I have to do during the week. I am very unorganized. Each day I want to find the biggest block of free time to go on to Reddit. But I am not sure when that time is. I am also curious how I spend my days.

This challenge you will help me get organized and find that time for me to be on Reddit.

#Input:

I will give you a listing of the post it notes around my monitor. Each line represents a single post it note. Sorry but they are not in any order but I was at least smart enough to date them and put the times of my daily events.

#Output:

Get me organized. I need to see my schedule for the week. For each day you must find the 1 block of time that is the most time between events on the post its that I can Reddit. Please help maximize my time on Reddit. Assume my start time at work is the beginning of the first event and my end time at work is the end time of the last event for that day.

Then show me my final schedule. And while you are at it show me across the week how many minutes I dedicate to each task with a percentage of time it takes up my time. Hopefully I don't spend most of my time on Reddit.

#Challenge Input:

```
    11-6-2014: 05:18 AM to 06:00 AM -- code review
    11-9-2014: 08:52 AM to 09:15 AM -- food
    11-8-2014: 07:00 PM to 08:05 PM -- meeting
    11-8-2014: 05:30 PM to 06:36 PM -- personal appointment
    11-6-2014: 02:47 PM to 03:23 PM -- work
    11-11-2014: 07:14 AM to 08:32 AM -- meeting
    11-11-2014: 11:22 AM to 12:10 PM -- code review
    11-8-2014: 01:39 PM to 02:06 PM -- food
    11-9-2014: 07:12 AM to 08:06 AM -- meeting
    11-9-2014: 02:14 PM to 03:15 PM -- code review
    11-8-2014: 05:13 AM to 06:05 AM -- food
    11-6-2014: 05:54 PM to 06:17 PM -- personal appointment
    11-7-2014: 08:24 AM to 09:23 AM -- personal appointment
    11-8-2014: 11:28 AM to 12:44 PM -- meeting
    11-7-2014: 09:35 AM to 10:35 AM -- workout
    11-9-2014: 10:05 AM to 11:15 AM -- code review
    11-11-2014: 05:02 PM to 06:09 PM -- work
    11-6-2014: 06:16 AM to 07:32 AM -- food
    11-10-2014: 10:08 AM to 11:14 AM -- workout
    11-8-2014: 04:33 PM to 05:12 PM -- meeting
    11-10-2014: 01:38 PM to 02:10 PM -- workout
    11-11-2014: 03:03 PM to 03:40 PM -- food
    11-11-2014: 05:03 AM to 06:12 AM -- food
    11-9-2014: 09:49 AM to 10:09 AM -- meeting
    11-8-2014: 06:49 AM to 07:34 AM -- work
```

```
11-7-2014: 07:29 AM to 08:22 AM -- food
11-10-2014: 03:08 PM to 03:29 PM -- code review
11-9-2014: 03:27 PM to 04:39 PM -- food
11-7-2014: 05:38 AM to 06:49 AM -- meeting
11-7-2014: 03:28 PM to 04:06 PM -- code review
11-8-2014: 02:44 PM to 03:35 PM -- meeting
11-6-2014: 08:53 AM to 09:55 AM -- workout
11-11-2014: 02:05 PM to 02:49 PM -- meeting
11-10-2014: 08:29 AM to 09:23 AM -- code review
11-10-2014: 11:09 AM to 11:35 AM -- sales call
11-6-2014: 11:29 AM to 12:18 PM -- code review
11-11-2014: 08:04 AM to 08:45 AM -- work
11-9-2014: 12:27 PM to 01:29 PM -- sales call
11-7-2014: 11:04 AM to 12:07 PM -- code review
11-11-2014: 09:21 AM to 10:37 AM -- food
11-8-2014: 09:34 AM to 10:53 AM -- meeting
11-11-2014: 12:36 PM to 01:30 PM -- meeting
11-10-2014: 05:44 AM to 06:30 AM -- personal appointment
11-6-2014: 04:22 PM to 05:05 PM -- code review
11-6-2014: 01:30 PM to 01:59 PM -- sales call
11-10-2014: 06:54 AM to 07:41 AM -- code review
11-9-2014: 11:56 AM to 12:17 PM -- work
11-10-2014: 12:20 PM to 01:17 PM -- personal appointment
11-8-2014: 07:57 AM to 09:08 AM -- meeting
11-7-2014: 02:34 PM to 03:06 PM -- work
11-9-2014: 05:13 AM to 06:25 AM -- workout
11-11-2014: 04:04 PM to 04:40 PM -- food
11-9-2014: 06:03 AM to 06:26 AM -- code review
11-6-2014: 10:32 AM to 11:22 AM -- sales call
11-6-2014: 07:51 AM to 08:25 AM -- personal appointment
11-7-2014: 01:07 PM to 02:14 PM -- meeting
```

#FAQ:

Dates are mm-dd-yyyy

Title: [2014-11-12] Challenge #188 [Intermediate] Box Plot Generator
Text: # [](#IntermediateIcon) **(Intermediate)**: Box Plot Generator

A [box plot](http://en.wikipedia.org/wiki/Box_plot) is a convenient way of representing a set of univariate (one-variable) numerical data, while showing some useful statistical info about it at the same time. To understand what a box plot represents you need to learn about quartiles.

## Quartiles

Quartiles show us some info on the distribution of data in a data set. For example, here's a made-up data set representing the number of lines of code in 30 files of a software project, arranged into order.

```
7 12 21 28 28 29 30 32 34 35 35 36 38 39 40 40 42 44 45 46 47 49 50 53 55 56 59 63 77 191
```

The three quartiles can be found at the quarter intervals of a data set. For this example, the number of data items is 30, so the lower quartile (**Q1**) is item number (`30/4=8` - round up) which the value is `32`. The median quartile (**Q2**) is item number (`2*30/4=15`) which the value is `40`. The upper quartile (**Q3**) is item number (`3*30/4=23` - round up) which the value is `50`. The bit between **Q1** and **Q3** is called the **inter quartile range** or **IQR**. To demonstrate the fact that this splits the data set into 'quarters' the quartiles here are displayed.

```
7 12 21 28 28 29 30 32 34 35 35 36 38 39 40 40 42 44 45 46 47 49 50 53 55 56 59 63 80 191
```

```
                    ||                ||                  ||
  --- 1st quarter ----Q1--- 2nd quarter ---Q2---- 3rd quarter -----Q3--- 4th quarter -----
                    \          inter quartile range          /
```

The value of the **IQR** here is `50-32=18` (ie. **Q3**-**Q1**.) This forms the 'box' part of the box plot, with the line in the moddle of it representing the median **Q2** point. The 'whiskers' of the box plot are also fairly easy to work out. They represent the rest of the data set that isn't an **outlier** (anomalous). For example, here the 191-line-long file is an anomaly among the rest, and the 7-ling-long file might be too. How do we say for sure what is an anomaly and what isn't? If the data point is at the lower end of the data set, you work out if the value is less than 1.5 times the inter-quartile range from **Q1** - ie. if `x < Q1 - 1.5 * IQR`. If the data point is at the higher end of the data set, you work out of the value is more than 1.5 times the inter-quartile range from **Q3** - ie. if `x > Q3 + 1.5 * IQR`. Here, for 7, `Q1 - 1.5 * IQR` is `32 - 27 = 5`, and `7 > 5`, so 7 is not an outlier. But for 191, `Q3 + 1.5 * IQR` is `50 + 27 = 77`, and both 90 and 191 are greater than 77, so they are outliers. The end of the 'whiskers' on the box plot (the endmost bits) are the first and last values that *aren't* outliers - any outlying points are represented as crosses `x` outside of the plot.

Note: in reality, a better method than rounding up the quartile indices is usually used.

# Formal Inputs and Outputs

## Input Description

The program is to accept any number of numerical values, separated by whitespace.

## Output Description

You are to output the box plot for the input data set. You have some freedom as to how you draw the box plot - you could dynamically generate an image, for example, or draw it ASCII style.

# Sample Inputs and Outputs

## Sample Input

The example above: `7 12 21 28 28 29 30 32 34 35 35 36 38 39 40 40 42 44 45 46 47 49 50 53 55 56 59 63 80 191`

Unique traffic data for this sub:

```
2095 2180 1049 1224 1350 1567 1477 1598 1462  972 1198 1847
2318 1460 1847 1600  932 1021 1441 1533 1344 1943 1617  978
1251 1157 1454 1446 2182 1707 1105 1129 1222 1869 1430 1529
1497 1041 1118 1340 1448 1300 1483 1488 1177 1262 1404 1514
1495 2121 1619 1081  962 2319 1891 1169
```

# Sample Output

Sample output from my solution here: http://i.imgur.com/RIfoQ54.png (fixed now, sorry.)

# Extension (intermediate)

What about if you wish to compare two data sets? Allow your program to accept two or more data-sets, plotting the box plots such that they can be compared visually.

Title: [2014-11-19] Challenge #189 [Intermediate] Roman Numeral Conversion
Text: Your friend is an anthropology major who is studying roman history. They have never been able to quite get a handle for roman numerals and how to read them, so they've asked you to come up with a simple program that will let them input some numbers and return roman numerals, as well as the opposite, to input roman numerals and return base-10 numbers. They are bribing you with Indiana Jones memorabilia, so you are totally up for the challenge!

#Description

Most people learn about roman numerals at a young age. If you look at many analog clocks, you will find that many of them actually use roman numerals for the numbers. Roman numerals do not just stop at 12 though, they actually can represent numbers as high as 4999 using their most basic form.
The challenge, is to create a program that will allow you to convert decimal (base-10) numbers to roman numerals as well as roman numerals to decimal numbers. The history of roman numerals is a bit debated because of their varied use throughout history and a seeming lack of a standard definition. Some rules are well accepted and some less-so. Here are the guidelines for your implementation:

I	V	X	L	C	D	M
1	5	10	50	100	500	1000

#Rules

You cannot repeat the same roman numeral more than three times in a row, except for M, which can be added up to four times. (Note: Some descriptions of roman numerals allows for IIII to represent 4 instead of IV. For the purposes of this exercise, that is not allowed.)
When read from left to right, if successive roman numerals decrease or stay the same in value, you add them to the total sum.
When read from left to right, if successive roman numerals increase in value, you subtract the smaller value from the larger one and add the result to the total sum.

#Restrictions

I can only be subtracted from V or X

X can only be subtracted from L or C

C can only be subtracted from D or M

Only one smaller value can be subtracted from a following larger value. (e.g. 'IIX' would be an invalid way to represent the number 8)

#Examples

XII = 10 + 1 + 1 = 12

MDCCLXXVI = 1000 + 500 + 100 + 100 + 50 + 10 + 10 + 5 + 1 = 1776

IX = "1 from 10" = 10 - 1 = 9

XCIV = "10 from 100" + "1 from 5" = (100 - 10) + (5 - 1) = 90 + 4 = 94

#Inputs & Outputs

Your program should be able to accept numbers in either integer or roman numeral format to return the other. You may want to add validation checks on the input.
When converting to a roman numeral, the maximum number is 4999.
When converting from a roman numeral, I,V,X,L,C,D,M are the only valid characters.
You should be able to accept one or many numbers or numerals and convert to the other direction.

#Challenge

Some historical accounts state that roman numerals could actually go much higher than 4999. There are incredibly varied explanations and syntactical requirements for them. Some state that an over-line (vinculum) would be used over a number to multiply it by 1000, some say that you would put a curved line on either side of a number to multiply it by 1000.
For the challenge, see if you can add support to your code to allow parenthesis to encapsulate parts of a number that can be multiplied by one thousand. You can nest parenthesis as well to allow for numbers that are incredibly large.

#Restriction

The last roman numeral digit inside a set of parenthesis can not be an "I". There are two reasons for this (1) because historical accounts claimed that confusion would happen with the curved lines that encapsulate a number to be multiplied by one thousand and (2) because the easiest way to validate your numbers is with Wolfram Alpha and they do not allow it either.

#Examples

(V)M = 5*1000 + 1000 = 6000

(X)MMCCCXLV = 10*1000 + 1000 + 1000 + 100 + 100 + 100 + (50 - 10) + 5 = 10000 + 2000 + 300 + 40 + 5 = 12345

((XV)M)DCC = ((10 + 5) * 1000 + 1000) * 1000 + 500 + 100 + 100 = (15000 + 1000) * 1000 + 1700 = 16000000 + 1700 = 16001700

#Hints

You can visit Wolfram Alpha to validate some of your numbers if you are having any trouble.
http://www.wolframalpha.com/input/?i=314+in+roman+numerals

#Sample Data
##Basic
IV = 4

XXXIV = 34

CCLXVII = 267

DCCLXIV = 764

CMLXXXVII = 987

MCMLXXXIII = 1983

MMXIV = 2014

MMMM = 4000

MMMMCMXCIX = 4999

##Challenge
(V) = 5000

(V)CDLXXVIII = 5478

(V)M = 6000

(IX) = 9000

(X)M = 11000

(X)MM = 12000

(X)MMCCCXLV = 12345

(CCCX)MMMMCLIX = 314159

(DLXXV)MMMCCLXVII = 578267

(MMMCCXV)CDLXVIII = 3215468

(MMMMCCX)MMMMCDLXVIII = 4214468

(MMMMCCXV)CDLXVIII = 4215468

(MMMMCCXV)MMMCDLXVIII = 4218468

(MMMMCCXIX)CDLXVIII = 4219468

((XV)MDCCLXXV)MMCCXVI = 16777216

((CCCX)MMMMCLIX)CCLXV = 314159265

((MLXX)MMMDCCXL)MDCCCXXIV = 1073741824

Title: [2014-11-26] Challenge #190 [Intermediate] Words inside of words
Text: #Description

This weeks challenge is a short yet interesting one that should hopefully help you exercise elegant solutions to a problem rather than bruteforcing a challenge.

#Challenge

Given the wordlist [enable1.txt](http://www.joereynoldsaudio.com/enable1.txt), you must find the word in that file which also contains the greatest number of words within that word.

For example, the word 'grayson' has the following words in it

Grayson

Gray

Grays

Ray

Rays

Son

On

Here's another example, the word 'reports' has the following

reports

report

port

ports

rep

You're tasked with finding the word in that file that contains the most words.

NOTE : If you have a different wordlist you would like to use, you're free to do so.

#Restrictions

* To keep output slightly shorter, a word will only be considered a word if it is 2 or more letters in length

* The word you are using may not be permuted to get a different set of words (You can't change 'report' to 'repotr' so that you can add more words to your list)

Title: [2014-12-3] Challenge #191 [Intermediate] Space Probe. Alright Alright Alright.
Text: #Description:

NASA has contracted you to program the AI of a new probe. This new probe must navigate space from a starting location to an end location. The probe will have to deal with Asteroids and Gravity Wells. Hopefully it can find the shortest path.

#Map and Path:

This challenge requires you to establish a random map for the challenge. Then you must navigate a probe from a starting location to an end location.

#Map:

You are given N -- you generate a NxN 2-D map (yes space is 3-D but for this challenge we are working in 2-D space)

* 30% of the spots are "A" asteroids
* 10% of the spots are "G" gravity wells (explained below)
* 60% of the spots are "." empty space.

When you generate the map you must figure out how many of each spaces is needed to fill the map. The map must then be randomly populated to hold the amount of Gravity Wells and Asteroids based on N and the above percentages.

## N and Obstacles

As n changes so does the design of your random space map. Truncate the amount of obstacles and its always a min size of 1. (So say N is 11 so 121 spaces. At 10% for wells you need 12.1 or just 12 spots) N can be between 2 and 1000. To keep it simple you will assume every space is empty then populate the random Asteroids and Gravity wells (no need to compute the number of empty spaces - they will just be the ones not holding a gravity well or asteroid)

## Asteroids

Probes cannot enter the space of an Asteroid. It will just be destroyed.

## Empty Spaces

Probes can safely cross space by the empty spaces of space. Beware of gravity wells as described below.

## Gravity Wells

Gravity wells are interesting. The Space itself is so dense it cannot be travelled in. The adjacent spaces of a Gravity well are too strong and cannot be travelled in. Therefore you might see this.

. = empty space, G = gravity well

```
    .....
    .....
    ..G..
    .....
    .....
```

But due to the gravity you cannot pass (X = unsafe)

```
    .....
    .XXX.
    .XGX.
    .XXX.
    .....
```

You might get Gravity wells next to each other. They do not effect each other but keep in mind the area around them will not be safe to travel in.

```
    ......
    .XXXX.
    .XGGX.
    .XXXX.
    ......
```

#Probe Movement:

Probes can move 8 directions. Up, down, left, right or any of the 4 adjacent corners. However there is no map wrapping. Say you are at the top of the map you cannot move up to appear on the bottom of the map. Probes cannot fold space. And for whatever reason we are contained to only the spots on the map even thou space is infinite in any direction.

#Output:

Must show the final Map and shortest safe route on the map.

* . = empty space
* S = start location
* E = end location
* G = gravity well
* A = Asteroid
* O = Path.

If you fail to get to the end because of no valid path you must travel as far as you can and show the path. Note that the probe path was terminated early due to "No Complete Path" error.

#Challenge Input:

using (row, col) for coordinates in space.

Find solutions for:

* N = 10, start = (0,0) end = (9,9)

* N = 10, start = (9, 0) end = (0, 9)
* N= 50, start = (0,0) end = (49, 49)

#Map Obstacle %

I generated a bunch of maps and due to randomness you will get easy ones or hard ones. I suggest running your solutions many times to see your outcomes. If you find the solution is always very straight then I would increase your asteroid and gravity well percentages. Or if you never get a good route then decrease the obstacle percentages.

Title: [2014-12-10] Challenge #192 [Intermediate] Markov Chain Error Detection
Text: # [](#IntermediateIcon) **(Intermediate)**: Markov Chain Error Detection

A Markov process describes a system where the probability of changing to a certain state is dependent on the current state. A Markov Chain is a system where there is a discrete set of states. One application of this is in some predictive-texting systems. For example, a Markov chain can describe how, in writing, the word 'car' has a higher probability of being followed by the word 'key' than the word 'banana' or 'the'. This system is handy as it allows the predictive-texting system to adapt (in a limited way) to the specific user. For example, for the word 'source', an academic would have a likely following word as 'reference', whereas a programmer would have a likely following word as 'code' - as the text 'source reference' might be used a lot by an academic whereas the text 'source code' would be used a lot by a developer. This is of course a crude example but it illustrates the point nicely.

The Markov chain could be represent in memory via a matrix. For example, for a small sample of 4 words in a paragraph, the matrix may look like:

	The	Thing	Did	Do
The	0	12	0	0
Thing	0	0	3	5
Did	6	0	0	11
Do	8	0	0	0

At a glance you can see the number of times the word 'thing' was followed by 'do' more than 'did', and the word 'do' was preceded more by 'did' than 'thing'. There are other ways to store this data, of course - the implementation of this part is up to you.

This can be used to detect errors in input. For example, you could use the above table to predict that a sentence containing 'the do' is likely to be erroneous. Your challenge today will involve letters in words (rather than words in sentences) to predict if a word is likely to be misspelled or not.

# Formal Inputs and Outputs

## Input Description

The program is to utilise [a word list of your choice](https://www.reddit.com/r/dailyprogrammer/comments/2nluof/request_the_ultimate_wordlist/) to construct Markov chain data for the occurrence of certain letters following other letters. For example, the word 'occurrence' would have a matrix that looks like:

	O	C	U	R	E	N
O	0	1	0	0	0	0
C	0	1	1	0	1	0
U	0	0	0	1	0	0
R	0	0	0	1	1	0
E	0	0	0	0	0	1
N	0	1	0	0	0	0

Of course with more data used to populate the table the numbers would be larger and more meaningful.

The program is also to accept a word to compare against the Markov chain - your program will predict whether the word is likely to be misspelled or not. You may ask 'why not just check against a word-list?' In most cases that would be fine. However, is a word is amalgamated like errorcorrection then this system should still find that the word is likely to be valid (if not malformed.)

## Output Description

You have some freedom in this section. The specific way of determining the likelihood of a word being invalid is up to you. A naive one would check if the word contains any consecutive letters that have a 0 for the Markov chain count - for example, the word 'examqle' is likely to misspelled as Q probably never follows M in the word-list. You will need to do some of the testing of this yourself, and hence different people's solutions may differ.

# Sample Inputs and Outputs

## Word List Data

You can use some of the word lists linked to in [our currently-stickied post](https://www.reddit.com/r/dailyprogrammer/comments/2nluof/request_the_ultimate_wordlist/) (at the time of writing.)

## Sample Input

I assume you can come up with some testing data yourself - just pick some actual words to test for validity, and fake words to test your program with, like horqqar or axumilog.

# Further Reading

Wikipedia page on Markov chains is [here](https://en.wikipedia.org/wiki/Markov_chain). An interesting use of Markov chains is automatic text generation based on previous input to train the program, [like this cool article](http://vedantmisra.com/markov-chain-paper-title-generator/).

Title: [14-12-17] Challenge #193 [Intermediate] 50,000 Subscriber Meta-challenge
Text: # [](#IntermediateIcon) **(Intermediate)**: 50,000 Subscriber Meta-challenge

Congratulations to everyone for getting the subreddit to 50K subscribers! As a reward I'll do a nice relaxed meta challenge. Effective communication is an important skill to have, but it certainly isn't easy; hence, it is a challenge unto itself. This also gives less experienced members of the subreddit a chance to see into the minds of the more veteran submitters.

# Challenge

Pick your favourite solution (that you have written) to a past challenge, or one that you are particularly proud of. It can be from any challenge, but preferably one with some complexity. Now, describe how it works (via in-code comments or otherwise) as you would to a person. Then, describe how you might improve it or do it differently in hindsight. Also, link to the challenge post itself.

Title: [2014-12-31] Challenge #195 [Intermediate] Math Dice
Text: #Description:

Math Dice is a game where you use dice and number combinations to score. It's a neat way for kids to get mathematical dexterity. In the game, you first roll the 12-sided Target Die to get your target number, then roll the five 6-sided Scoring Dice. Using addition and/or subtraction, combine the Scoring Dice to match the target number. The number of dice you used to achieve the target number is your score for that round. For more information, see the product page for the game: (http://www.thinkfun.com/mathdice)

#Input:

You'll be given the dimensions of the dice as NdX where N is the number of dice to roll and X is the size of the dice. In standard Math Dice Jr you have 1d12 and 5d6.

#Output:

You should emit the dice you rolled and then the equation with the dice combined. E.g.

```
9, 1 3 1 3 5

3 + 3 + 5 - 1 - 1 = 9
```

#Challenge Inputs:

```
1d12 5d6
1d20 10d6
1d100 50d6
```

Title: [2015-01-07] Challenge #196 [Intermediate] Rail Fence Cipher
Text: # [](#IntermediateIcon) **(Intermediate)**: Rail Fence Cipher

Before the days of computerised encryption, cryptography was done manually by hand. This means the methods of encryption were usually much simpler as they had to be done reliably by a person, possibly in wartime scenarios.

One such method was the [rail-fence cipher](http://en.wikipedia.org/wiki/Rail_fence_cipher). This involved choosing a number (we'll choose 3) and writing our message as a zig-zag with that height (in this case, 3 lines high.) Let's say our message is `REDDITCOMRDAILYPROGRAMMER`. We would write our message like this:

```
R   I   M   I   R   A   R
 E D T O R A L P O R M E
  D   C   D   Y   G   M
```

See how it goes up and down? Now, to get the ciphertext, instead of reading with the zigzag, just read along the lines instead. The top line has `RIMIRAR`, the second line has `EDTORALPORME` and the last line has `DCDYGM`. Putting those together gives you `RIMIRAREDTORALPORMEDCDYGM`, which is the ciphertext.

You can also decrypt (it would be pretty useless if you couldn't!). This involves putting the zig-zag shape in beforehand and filling it in along the lines. So, start with the zig-zag shape:

```
?   ?   ?   ?   ?   ?   ?
 ? ? ? ? ? ? ? ? ? ? ? ?
  ?   ?   ?   ?   ?   ?
```

The first line has 7 spaces, so take the first 7 characters (`RIMIRAR`) and fill them in.

```
R   I   M   I   R   A   R
 ? ? ? ? ? ? ? ? ? ? ? ?
  ?   ?   ?   ?   ?   ?
```

The next line has 12 spaces, so take 12 more characters (`EDTORALPORME`) and fill them in.

```
R   I   M   I   R   A   R
 E D T O R A L P O R M E
  ?   ?   ?   ?   ?   ?
```

Lastly the final line has 6 spaces so take the remaining 6 characters (`DCDYGM`) and fill them in.

```
R   I   M   I   R   A   R
 E D T O R A L P O R M E
```

```
    D   C   D   Y   G   M
```

Then, read along the fence-line (zig-zag) and you're done!

## Input Description

You will accept lines in the format:

```
    enc # PLAINTEXT
```

or

```
    dec # CIPHERTEXT
```

where `enc #` encodes `PLAINTEXT` with a rail-fence cipher using `#` lines, and `dec #` decodes `CIPHERTEXT` using `#` lines.

For example:

```
    enc 3 REDDITCOMRDAILYPROGRAMMER
```

## Output Description

Encrypt or decrypt depending on the command given. So the example above gives:

```
    RIMIRAREDTORALPORMEDCDYGM
```

# Sample Inputs and Outputs

```
    enc 2 LOLOLOLOLOLOLOLOLO
    Result: LLLLLLLLLOOOOOOOOO

    enc 4 THEQUICKBROWNFOXJUMPSOVERTHELAZYDOG
    Result: TCNMRZHIKWFUPETAYEUBOOJSVHLDGQRXOEO

    dec 4 TCNMRZHIKWFUPETAYEUBOOJSVHLDGQRXOEO
    Result: THEQUICKBROWNFOXJUMPSOVERTHELAZYDOG

    dec 7 3934546187438171450245968893099481332327954266552620198731963475632908289907
    Result: 3141592653589793238462643383279502884197169399375105820974944592307816406286 (pi)

    dec 6 AAPLGMESAPAMAITHTATLEAEDLOZBEN
    Result: ?
```

Title: [2015-01-14] Challenge #197 [Intermediate] Food Delivery Problem
Text: #Description:

You are owner of a new restaurant that is open 24 hours a day 7 days a week. To be helpful to your customers you deliver. To make sure you are the best in business you offer a guarantee of the fastest delivery of food during your hours of operation (which is all the time)

Our challenge this week is to build a program our delivery people can use to help pick the fastest route in time to get from a source to a destination in the town of our restaurant.

#City Routes

The city has many streets connected to many intersections. For the sake of naming we will label intersections with letters. Streets between intersections will use their street name.

#Time Intervals

The data for each street has 4 values of time in minutes. They represent the time it takes one to travel that street based on a fixed interval of time of day to travel on that street. The varied time is due to different traffic loads on that street.

* T1 = 0600-1000 (6 am to 10 am)
* T2 = 1000 - 1500 (10 am to 3 pm)
* T3 = 1500 - 1900 (3 pm to 7 pm)
* T4 = 1900 - 0600 (7 pm to 6 am)

#Data Format

(Start Intersection) (Stop Intersection) (Name of street) (T1) (T2) (T3) (T4)

```
    (Start Intersection) - The letter of that unique intersection
    (Stop Intersection) - The letter of that unique intersection
    (Name of Street) - Name of the street with this time data
    (T1 to T4) are the minutes it takes to travel based on fixed time intervals (described above)
```

#Data

The data:

```
    A B "South Acorn Drive" 5 10 5 10
    B C "Acorn Drive" 15 5 15 5
    C D "North Acorn Drive" 7 10 15 7
    H G "South Almond Way" 10 10 10 10
    G F "Almond Way" 15 20 15 20
    F E "North Almond Way" 5 6 5 6
    I J "South Peanut Lane" 8 9 10 11
    J K "Peanut Lane" 11 10 9 8
    K L "North Peanut Lane" 7 5 7 5
    P O "South Walnut" 6 5 6 5
    O N "Walnut" 10 8 10 8
    N M "North Walnut" 9 6 9 6
    D E "West Elm Street" 10 8 12 7
    E L "Elm Street" 12 11 12 8
    L M "East Elm Street" 5 4 5 4
    C F "West Central Avenue" 9 8 9 8
    F K "Central Avenue" 5 4 5 4
    K N "East Central Avenue" 9 9 9 9
    B G "West Pine Road" 7 6 7 6
    G J "Pine Road" 9 8 9 8
    J O "East Pine Road" 6 5 6 5
    A H "West Oak Expressway" 9 8 7 7
    H I "Oak Expressway" 10 10 10 10
    I P "East Oak Expressway" 8 7 8 7
```

#Time Changes and Routes

It is possible that a route might take you long enough that it might cross you over a time change such that the route times get change. To make this easier just please consider the time between intersections based on the start time of the drive. So say I pick 5:50am - and if the route would take us into 6am hour you don't have to compute the route times for 6am to 10am but just keep the route computed based on 7pm to 6am since our starting time was 5:50am.

#Challenge Input:

You will be given start and end intersections and time of day to compute a route.

#Challenge Output:

List the route direction street by street and time. This must be the "Fastest" route from start to end at that time of day. Also list the time it took you in minutes.

#Challenge Routes to solve:

```
    A M 0800
    A M 1200
    A M 1800
    A M 2200


    P D 0800
    P D 1200
    P D 1800
    P D 2200
```

Title: [2015-01-21] Challenge #198 [Intermediate] Base-Negative Numbers
Text: # [](#IntermediateIcon) _(Intermediate)_: Base-Negative Numbers

"Don't be stupid, Elite6809!", I hear you say. *"You can't have a negative base."* Well, why not? Let's analyse what we mean by base. Given a base-**r** system, the column **p** places from the right (starting from zero), which contains the digit **n**, has the value **n**×**r**^(**p**). The binary columns 1, 2, 4, 8, 16, ... is the same as 2^(0), 2^(1), 2^(2), 2^(3), 2^(4). Nothing stops you from using a negative base with this system, except perhaps the understanding of the concept and practicality of its usage.

Let's imagine base -10 (negadecimal). Here, the place values for each column are now 1, -10, 100, -1000 and so on. Therefore, the negadecimal number 7211:

```
    -Thousands    Hundreds    -Tens    Units
        7            2          1       1
     (-7000)   +   (200)   +   (-10) +  (1) = -6809
```

Is equivalent to -6809 in standard decimal.

Your challenge is, given a negative base and a value, convert it to the representation in the corresponding positive base, and vice versa.

# Input and Output Description

## Challenge Input

You will accept two numbers: **r** and **n**. **n** is a number in base **r**. For example:

```
    -4 1302201
```

This input means 1302201 in base -4.

## Challenge Output

Print the value of the input number in the corresponding opposite-signed base, for example, for the input above:

    32201

As 1302201 in base -4 equals 32201 in base 4.

# Sample Inputs and Outputs

Input: `-10 12345678` (convert from base -10 to base 10)
Output: `-8264462`

Input:`-7 4021553`
Output: `4016423`

Similarly, if the given base is *positive*, convert back to the corresponding negative base.

Input: `7 4016423` (convert from base 7 to base -7)
Output: `4021553`

Input: `6 -3014515`
Output: `13155121`

## Extension (Hard)

Extend your program to support imaginary bases. Imaginary bases can represent any complex number. The principle is the same; for example, base 4*i* can be used to represent complex numbers much the same way as a cartesian representation like a+b*i*.
If you have forgotten the principles of imaginary numbers, re-read the challenge description for [The Complex Number](/r/dailyprogrammer/comments/2nr6c4/) - you might want to re-use some code from that challenge anyway.

# Notes

Try and do both the main challenge and extension without looking for the conversion algorithms on the internet. This is part of the challenge!

Title: [2015-1-26] Challenge #199 Bank Number Banners Pt 2
Text: #Description

To do this challenge, first you must complete this weeks [Easy](http://www.reddit.com/r/dailyprogrammer/comments/2tr6yn/2015126_challenge_199_bank_number_banners_pt_1/) challenge.

Now, when we purchased these fax machines and wrote the programme to enable us to send numbers to our machine, we realised something... We couldn't translate it back!
This meant that sending a fax of this number format was useless as no one could interpret it.

Your job is to parse **back** the fax numbers into normal digits.

#Inputs & Outputs

##Input
As input, you should take the output of the easy challenge

##Output
Output will consists of integers that translate to what the fax read out.

These numbers :

```
 _ _ _ _ _ _ _ _ _
| || || || || || || || || |
|_||_||_||_||_||_||_||_||_|
```

```
  |  |  |  |  |  |  |  |  |
  |  |  |  |  |  |  |  |  |
```

```
    _  _  _  _  _  _     _
|_||_|| || ||_   |  |  ||_
  | _||_||_||_|  |  |  | _|
```

Would translate back to :

000000000

111111111

490067715

Title: [2015-02-04] Challenge #200 [Intermediate] Metro Tile Meltdown
Text: # [](#IntermediateIcon) _(Intermediate)_: Metro Tile Meltdown

In the continued name of backward-compatibility, Microsoft has released a version of their flagship operating system for VGA text-mode terminals. In this version of their operating system, rectangular tiles consisting of a single character are displayed on the screen, like so:

```
    .........................................................................
    .aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.bbbbbbbbbb.kkkkkkkkkkkkkkkk.jjjjjjjjjjjj.
    .aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.bbbbbbbbbb.kkkkkkkkkkkkkkkk.jjjjjjjjjjjj.
    .aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.bbbbbbbbbb.kkkkkkkkkkkkkkkk.jjjjjjjjjjjj.
    .aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.bbbbbbbbbb.kkkkkkkkkkkkkkkk.jjjjjjjjjjjj.
    .aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.bbbbbbbbbb.kkkkkkkkkkkkkkkk.jjjjjjjjjjjj.
    .aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.bbbbbbbbbb.kkkkkkkkkkkkkkkk.jjjjjjjjjjjj.
    .aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.bbbbbbbbbb.kkkkkkkkkkkkkkkk.jjjjjjjjjjjj.
    .aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.bbbbbbbbbb.kkkkkkkkkkkkkkkk.jjjjjjjjjjjj.
    ................................bbbbbbbbbb.kkkkkkkkkkkkkkkk.jjjjjjjjjjjj.
    .ddddddddddddddddd.cccccccccccc.bbbbbbbbbb.kkkkkkkkkkkkkkkk.jjjjjjjjjjjj.
    .ddddddddddddddddd.cccccccccccc.bbbbbbbbbb.kkkkkkkkkkkkkkkk.jjjjjjjjjjjj.
    .ddddddddddddddddd.cccccccccccc.bbbbbbbbbb.kkkkkkkkkkkkkkkk.jjjjjjjjjjjj.
    ...................cccccccccccc.bbbbbbbbbb.kkkkkkkkkkkkkkkk.jjjjjjjjjjjj.
    .eeeeeeeeee.ffffff.cccccccccccc.bbbbbbbbbb.kkkkkkkkkkkkkkkk.jjjjjjjjjjjj.
    .eeeeeeeeee.ffffff.cccccccccccc.bbbbbbbbbb.kkkkkkkkkkkkkkkk.jjjjjjjjjjjj.
    .eeeeeeeeee.ffffff.cccccccccccc.bbbbbbbbbb.kkkkkkkkkkkkkkkk.jjjjjjjjjjjj.
    .eeeeeeeeee.ffffff.cccccccccccc.............................jjjjjjjjjjjj.
    .eeeeeeeeee.ffffff.cccccccccccc.hhhhhhhhhhhhhhhhhhhhhhhhhhh.jjjjjjjjjjjj.
    .eeeeeeeeee.ffffff.cccccccccccc.hhhhhhhhhhhhhhhhhhhhhhhhhhh.jjjjjjjjjjjj.
    .eeeeeeeeee.ffffff.cccccccccccc.hhhhhhhhhhhhhhhhhhhhhhhhhhh.jjjjjjjjjjjj.
    .eeeeeeeeee..............................................................
    .eeeeeeeeee.gggggggggggggggggggggggggggggggggggggg.iiiiiiiiiiiiiiiiiiiii.
    .eeeeeeeeee.gggggggggggggggggggggggggggggggggggggg.iiiiiiiiiiiiiiiiiiiii.
    .eeeeeeeeee.gggggggggggggggggggggggggggggggggggggg.iiiiiiiiiiiiiiiiiiiii.
    .eeeeeeeeee.gggggggggggggggggggggggggggggggggggggg.iiiiiiiiiiiiiiiiiiiii.
    .eeeeeeeeee.gggggggggggggggggggggggggggggggggggggg.iiiiiiiiiiiiiiiiiiiii.
```

```
.eeeeeeeeee.ggggggggggggggggggggggggggggggggggggggggggg.iiiiiiiiiiiiiiiiiiii.
.eeeeeeeeee.ggggggggggggggggggggggggggggggggggggggggggg.iiiiiiiiiiiiiiiiiiii.
..........................................................................
```

Screen space with no tile is denoted by a period (`.`). Tiles can be made of any character other than periods (due to the reason given) and spaces.

However, the dev team forgot to add support for screen-readers! Now visually impaired users cannot determine the location of the tiles on their display. Your task is, given a tile display such as the one above, write a program to find the location and size of each rectangular tile on the screen, along with the character in it, and output it in a way that can be read by a screen reader. For example, one such tile in the above example is located at position `(1, 1)` on the screen (from the top-left), consists of the character `a` and is 30 characters wide and 8 characters tall.

## Tiles

A tile will always be perfectly rectangular:

```
aaaaaaaaaa
aaaaaaaaaa
aaaaaaaaaa
```

There will never be a non-rectangular tile on the screen, or one that is not completely filled in. These are not single tiles:

```
..................................
.bbbbbbbbbb.........ccccccccccccc.
.bbbbbbbbb..........c...........c.
.bbbbbbbb...........c...........c.
.bbbbbbb............ccccccccccccc.
..................................
```

A tile is something completely bordered by empty space (`.`), so this is two separate tiles:

```
.....................................
.aaaaaaaaaaaaaaaaaaa.aaaaaaaaaaaaaaa.
.aaaaaaaaaaaaaaaaaaa.aaaaaaaaaaaaaaa.
.aaaaaaaaaaaaaaaaaaa.aaaaaaaaaaaaaaa.
.aaaaaaaaaaaaaaaaaaa.aaaaaaaaaaaaaaa.
.....................................
```

Lastly, if a tile is made of two regions of separate colours, then the input as invalid:

```
.....................................
.aaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbb.
.aaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbb.
.aaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbb.
.aaaaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbb.
.....................................
```

The above 'tile' is two separate tiles: one made of `a`, one made of `b`.

Handling of invalid input is undefined and thus mostly up to you; your program can try and make sense of the input if you want, but for the purpose of the challenge, assume all tiles will be rectangular, separated by empty space (`.`) and consisting of a single character.

# Input and Output Description

## Sample Input

You will first be given two numbers, like so:

```
74 30
```

These denote the width and height of the tile display in characters. You will then be given the tile display of that size via standard input, for example:

```
..........................................................................
.aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.bbbbbbbb.ddddddddddddddddddddd.
.aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.bbbbbbbb.ddddddddddddddddddddd.
.aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.bbbbbbbb.ddddddddddddddddddddd.
.aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.bbbbbbbb.ddddddddddddddddddddd.
.aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.bbbbbbbb.ddddddddddddddddddddd.
.aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.bbbbbbbb.ddddddddddddddddddddd.
...........................................bbbbbbbb.ddddddddddddddddddddd.
.jjjjjjjjjjjjjjjjjjjjjjjjj.eeeeeeeeeeeeeee.bbbbbbbb.ddddddddddddddddddddd.
.jjjjjjjjjjjjjjjjjjjjjjjjj.eeeeeeeeeeeeeee.bbbbbbbb.ddddddddddddddddddddd.
.jjjjjjjjjjjjjjjjjjjjjjjjj.eeeeeeeeeeeeeee.bbbbbbbb.ddddddddddddddddddddd.
.jjjjjjjjjjjjjjjjjjjjjjjjj.eeeeeeeeeeeeeee.bbbbbbbb.ddddddddddddddddddddd.
.jjjjjjjjjjjjjjjjjjjjjjjjj.eeeeeeeeeeeeeee.bbbbbbbb.......................
.jjjjjjjjjjjjjjjjjjjjjjjjj.eeeeeeeeeeeeeee.bbbbbbbb.ccccccccccccccccccccc.
.jjjjjjjjjjjjjjjjjjjjjjjjj.eeeeeeeeeeeeeee.bbbbbbbb.ccccccccccccccccccccc.
...........................eeeeeeeeeeeeeee.bbbbbbbb.ccccccccccccccccccccc.
.iiiiiiiiii.kkkkkkkkkkkkkk.eeeeeeeeeeeeeee.bbbbbbbb.ccccccccccccccccccccc.
.iiiiiiiiii.kkkkkkkkkkkkkk.eeeeeeeeeeeeeee................................
.iiiiiiiiii.kkkkkkkkkkkkkk.eeeeeeeeeeeeeee.fffffffffffffff.gggggggggggggg.
.iiiiiiiiii.kkkkkkkkkkkkkk.eeeeeeeeeeeeeee.fffffffffffffff.gggggggggggggg.
.iiiiiiiiii.kkkkkkkkkkkkkk.eeeeeeeeeeeeeee.fffffffffffffff.gggggggggggggg.
.iiiiiiiiii.kkkkkkkkkkkkkk.eeeeeeeeeeeeeee.fffffffffffffff.gggggggggggggg.
.iiiiiiiiii................................fffffffffffffff.gggggggggggggg.
.iiiiiiiiii.hhhhhhhhhhhhhhhhhhhhhhhhhhhhhh.fffffffffffffff.gggggggggggggg.
.iiiiiiiiii.hhhhhhhhhhhhhhhhhhhhhhhhhhhhhh.fffffffffffffff.gggggggggggggg.
.iiiiiiiiii.hhhhhhhhhhhhhhhhhhhhhhhhhhhhhh.fffffffffffffff.gggggggggggggg.
.iiiiiiiiii.hhhhhhhhhhhhhhhhhhhhhhhhhhhhhh.fffffffffffffff.gggggggggggggg.
.iiiiiiiiii.hhhhhhhhhhhhhhhhhhhhhhhhhhhhhh.fffffffffffffff.gggggggggggggg.
.iiiiiiiiii.hhhhhhhhhhhhhhhhhhhhhhhhhhhhhh.fffffffffffffff.gggggggggggggg.
..........................................................................
```

## Sample Output

You are to print the location (with `(0, 0)` being the top-left), width, height and filled character of each tile on the screen, like this:

```
41×6 tile of character 'a' located at (1,1)
8×16 tile of character 'b' located at (43,1)
21×4 tile of character 'c' located at (52,13)
21×11 tile of character 'd' located at (52,1)
15×14 tile of character 'e' located at (27,8)
15×11 tile of character 'f' located at (43,18)
14×11 tile of character 'g' located at (59,18)
30×6 tile of character 'h' located at (12,23)
10×13 tile of character 'i' located at (1,16)
25×7 tile of character 'j' located at (1,8)
14×6 tile of character 'k' located at (12,16)
```

# Sample Inputs and Outputs

## Input

```
4 4
xx.z
xx..
..yy
z.yy
```

## Output

```
2×2 tile of character 'x' located at (0,0)
1×1 tile of character 'z' located at (0,3)
2×2 tile of character 'y' located at (2,2)
1×1 tile of character 'z' located at (3,0)
```

## Input

```
10 10
..........
.@@@@@.ss.
.@@@@@.ss.
.......ss.
.\\\\\.ss.
.\\\\\....
.\\\\\.\\.
.......\\.
./////.\\.
..........
```

## Output

```
5×2 tile of character '@' located at (1,1)
5×3 tile of character '\' located at (1,4)
5×1 tile of character '/' located at (1,8)
2×4 tile of character 's' located at (7,1)
2×3 tile of character '\' located at (7,6)
```

## Input

```
74 30
..........................................................................
.aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.bbbbbbbbbb.kkkkkkkkkkkkkkkk.jjjjjjjjjjjjj.
.aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.bbbbbbbbbb.kkkkkkkkkkkkkkkk.jjjjjjjjjjjjj.
.aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.bbbbbbbbbb.kkkkkkkkkkkkkkkk.jjjjjjjjjjjjj.
.aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.bbbbbbbbbb.kkkkkkkkkkkkkkkk.jjjjjjjjjjjjj.
.aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.bbbbbbbbbb.kkkkkkkkkkkkkkkk.jjjjjjjjjjjjj.
.aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.bbbbbbbbbb.kkkkkkkkkkkkkkkk.jjjjjjjjjjjjj.
.aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.bbbbbbbbbb.kkkkkkkkkkkkkkkk.jjjjjjjjjjjjj.
.aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.bbbbbbbbbb.kkkkkkkkkkkkkkkk.jjjjjjjjjjjjj.
................................bbbbbbbbbb.kkkkkkkkkkkkkkkk.jjjjjjjjjjjjj.
.ddddddddddddddddd.cccccccccccc.bbbbbbbbbb.kkkkkkkkkkkkkkkk.jjjjjjjjjjjjj.
.ddddddddddddddddd.cccccccccccc.bbbbbbbbbb.kkkkkkkkkkkkkkkk.jjjjjjjjjjjjj.
.ddddddddddddddddd.cccccccccccc.bbbbbbbbbb.kkkkkkkkkkkkkkkk.jjjjjjjjjjjjj.
...................cccccccccccc.bbbbbbbbbb.kkkkkkkkkkkkkkkk.jjjjjjjjjjjjj.
.eeeeeeeeee.ffffff.cccccccccccc.bbbbbbbbbb.kkkkkkkkkkkkkkkk.jjjjjjjjjjjjj.
.eeeeeeeeee.ffffff.cccccccccccc.bbbbbbbbbb.kkkkkkkkkkkkkkkk.jjjjjjjjjjjjj.
.eeeeeeeeee.ffffff.cccccccccccc.bbbbbbbbbb.kkkkkkkkkkkkkkkk.jjjjjjjjjjjjj.
.eeeeeeeeee.ffffff.cccccccccccc.............................jjjjjjjjjjjjj.
.eeeeeeeeee.ffffff.cccccccccccc.hhhhhhhhhhhhhhhhhhhhhhhhhhh.jjjjjjjjjjjjj.
.eeeeeeeeee.ffffff.cccccccccccc.hhhhhhhhhhhhhhhhhhhhhhhhhhh.jjjjjjjjjjjjj.
```

```
    .eeeeeeeeee.ffffff.cccccccccccc.hhhhhhhhhhhhhhhhhhhhhhhhhhh.jjjjjjjjjjjjj.
    .eeeeeeeeee...............................................................
    .eeeeeeeeee.gggggggggggggggggggggggggggggggggggggg.iiiiiiiiiiiiiiiiiiiiii.
    .eeeeeeeeee.gggggggggggggggggggggggggggggggggggggg.iiiiiiiiiiiiiiiiiiiiii.
    .eeeeeeeeee.gggggggggggggggggggggggggggggggggggggg.iiiiiiiiiiiiiiiiiiiiii.
    .eeeeeeeeee.gggggggggggggggggggggggggggggggggggggg.iiiiiiiiiiiiiiiiiiiiii.
    .eeeeeeeeee.gggggggggggggggggggggggggggggggggggggg.iiiiiiiiiiiiiiiiiiiiii.
    .eeeeeeeeee.gggggggggggggggggggggggggggggggggggggg.iiiiiiiiiiiiiiiiiiiiii.
    .eeeeeeeeee.gggggggggggggggggggggggggggggggggggggg.iiiiiiiiiiiiiiiiiiiiii.
    ..........................................................................
```

## Output

```
    30×8 tile of character 'a' located at (1,1)
    17×3 tile of character 'd' located at (1,10)
    10×15 tile of character 'e' located at (1,14)
    6×7 tile of character 'f' located at (12,14)
    38×7 tile of character 'g' located at (12,22)
    12×11 tile of character 'c' located at (19,10)
    10×16 tile of character 'b' located at (32,1)
    27×3 tile of character 'h' located at (32,18)
    16×16 tile of character 'k' located at (43,1)
    22×7 tile of character 'i' located at (51,22)
    13×20 tile of character 'j' located at (60,1)
```

Title: [2015-02-11] Challenge #201 [Practical Exercise] Get Your Priorities Right!
Text: # [](#PEIcon) **(Practical Exercise)**: Get Your Priorities Right!

A [priority queue](http://en.wikipedia.org/wiki/Priority_queue) is a data structure similar to a standard queue, except that entries in the queue have a *priority* associated with them - such that, when removing items from the queue, the entry with the highest priority will be removed before ones with lower priority. This is similar to a hospital triage system: patients with more severe wounds get treated quicker, even if they arrived later than a patient with a smaller injury. Let's say I have a priority queue containing strings, where the priority value is a real number. I add these 3 objects to my priority queue, in no particular order:

Patient	Priority
`"David Whatsit"`	3.06
`"Joan Smith"`	4.33
`"Bob Testing"`	0.39
`"Samuel Sample"`	1.63

Here, if I was to *dequeue* four strings from the priority queue, the strings `"Joan Smith"`, `"David Whatsit"`, `"Samuel Sample"` and `"Bob Testing"` would come out, in that order.

But what if we could assign *two* priorities to each object? Imagine a hospital (to keep with the theme), that needs to keep a list of equipment supplies and their costs. It also needs to keep track of how long it will take to receive that item.

Item	Cost	Shipping Time
Hyperion Hypodermic Needle	£1.99	3 days
SuperSaver Syringe	£0.89	7 days
InjectXpress Platinum Plated Needle	£2.49	2 days

Here, when the hospital is at normal running conditions with good supply stock, it would want to buy the cheapest product possible - shipping time is of little concern. Thus, dequeueing by the *Lowest Cost* priority would give us the SuperSaver syringe. However, in a crisis (where supply may be strained) we want supplies as fast as possible, and thus dequeueing an item with the *Lowest

Shipping Time* priority would give us the InjectXpress needle. This example isn't the best, but it gives an example of a priority queue that utilizes *two priority values* for each entry.

Your task today for the (non-extension) challenge will be to implement a two-priority priority queue for strings, where the priority is represented by a real number (eg. a floating-point value). The priority queue must be able to hold an unbounded number of strings (ie. no software limit). **If your language of choice already supports priority queues with 1 priority, it might not be applicable to this challenge - read the specification carefully**.

# Specification

## Core

Your priority queue must implement *at least* these methods:

* `Enqueue`. This method accepts three parameters - a **string**, **priority value A**, and **priority value B**, where the priority values are *real numbers* (see above). The **string** is inserted into the priority queue with the given priority values **A** and **B** (how you store the queue in memory is up to you!)

* `DequeueA`. This method removes and returns the **string** from the priority queue with the highest **priority A** value. If two entries have the same A priority, return whichever was enqueued first.

* `DequeueB`. This method removes and returns the **string** from the priority queue with the highest **priority B** value. If two entries have the same B priority, return whichever was enqueued first.

* `Count`. This method returns the number of strings in the queue.

* `Clear`. This removes *all* entries from the priority queue.

## Additional

If you can, implement this method, too:

* `DequeueFirst`. This removes and returns the **string** from the priority queue that was **enqueued first**, ignoring priority.

Depending on how you implemented your priority queue, this may not be feasible, so don't get too hung up on this one.

## Extension

Rather than making your priority queue only accept strings, make a **generic** priority queue, instead. A generic object is compatible with many types. In C++, this will involve the use of [templates](//en.wikipedia.org/wiki/Template_\(C%2B%2B\)). More reading resources [here](//en.wikipedia.org/wiki/Generic_programming). For example, in C#, your class name might look like `DualPriorityQueue<TEntry>`. Some dynamic languages such as Ruby or Python do not have static typing, so this will not be necessary.

# Notes

## Making Use of your Language

The main challenge of this exercise is knowing your language and its features, and adapting your solution to them. For example, in .NET-based languages, `Count` would be a property rather than a method, as that is more idiomatic in those languages. Similarly, in some languages such as Ruby, F# or other functional language, overloading operators may be more idiomatic than the usage of verbose `Enqueue` and `Dequeue` functions. How you do the specifics is up to you.

You should also be writing clean, legible code. Follow the style guide for your language, and use the correct naming/capitalisation conventions, which differ from language to language. Consider writing [unit tests](http://en.wikipedia.org/wiki/Unit_testing) for your code, as an exercise in good practice!

## Tips and Reading Material

If you are planning on using something like a heap for the priority queue, consider interleaving each item into two heaps to store both priorities. How you will handle the dequeueing is part of the fun! If the concept of a priority queue is new to you, here is some reading material.

* [University of Wisconsin-Madison: priority queue](http://pages.cs.wisc.edu/~vernon/cs367/notes/11.PRIORITY-Q.html)
* [Queues and Priority Queues](http://www.oopweb.com/Java/Documents/ThinkCSJav/Volume/chap16.htm)

Here's some more stuff on unit testing.

* [Unit testing on WikiBooks](http://en.wikibooks.org/wiki/Introduction_to_Software_Engineering/Testing/Unit_Tests)
* [MSDN: .NET unit tests](https://msdn.microsoft.com/en-us/library/hh694602.aspx)
* [Writing good unit tests](https://developer.salesforce.com/page/How_to_Write_Good_Unit_Tests)

Title: [2015-02-18] Challenge #202 [Intermediate] Easter Challenge
Text: #Description:

Given the year - Write a program to figure out the exact date of Easter for that year.

#Input:

A year.

#Output:

The date of easter for that year.

#Challenge:

Figure out easter for 2015 to 2025.

Title: [2015-03-02] Challenge #204 [Intermediate] It's like regular binary, only way more hype!
Text: #Description

We all know and love the binary number system, but today we're going to do something a little bit different with it. We're going to break it by adding another number.

The regular binary number system uses two digits, 0 and 1, and the positions they are put in represents different powers of 2, increasing from right to left. So, for example, if you have the binary number 110101, that is equal to

1*2^5 + 1*2^4 + 0*2^3 + 1*2^2 + 0*2^1 + 1*2^0

= 2^5 + 2^4 + 2^2 + 2^0

= 32 + 16 + 4 + 1

= 53

Easy enough, but now lets have some fun with it.

Imagine that instead of having just the two digits 0 and 1, the binary number system had three digits, 0, 1 and 2 with everything else working exactly the same. This system is known as the "hyperbinary number system".

Lets see an example how the hyperbinary number system works. Lets take the hyperbinary number "1021", and try and figure out what number it represents. Just as before, each position represents a power of 2, but now you can have 0, 1 or 2 of each of them, so the calculation goes like this:

1*2^3 + 0*2^2 + 2*2^1 + 1*2^0

= 8 + 2*2 + 1

= 8 + 4 + 1

= 13

Interestingly, this is not the only way you can represent the number 13 in hyperbinary, you could also write 13 as "221" and "1101".

In fact, this is a common issue with this number system: most numbers can be written in multiple ways in hyperbinary. Your challenge today is to find every single hyperbinary representation of a given number.

#Formal Inputs and Outputs

##Input description

The input will be a single line containing a single number (written in regular decimal).

##Output description

Your program should print out all possible representations of the input number in hyperbinary, one per line. Every representation should be printed out once and only once. The order of the outputs doesn't matter, and you can use leading zeroes if you want to.

#Examples

##Input 1

```
18
```

##Output 1

```
1122
1202
1210
2002
2010
10002
10010
```

##Input 2

```
73
```

##Output 2

```
112121
112201
120121
120201
121001
200121
200201
201001
1000121
1000201
1001001
```

##Challenge inputs

##Input 1

```
128
```

##Input 2

```
239
```

#Bonus

If you're looking for a stiffer challenge, try this input:

```
12345678910
```

I wouldn't recommend printing all the representations of that number out, though, becuse there are quite a few of them.

Have your program generate all the hyperbinary representations of that number, and then count them. Exactly how many are there?

Title: [2015-03-11] Challenge #205 [Intermediate] RPN
Text: #Description:

My father owned a very old HP calculator. It was in reverse polish notation (RPN). He would hand me his calculator and tell me "Go ahead and use it". Of course I did not know RPN so everytime I tried I failed.

So for this challenge we will help out young coder_d00d. We will take a normal math equation and convert it into RPN. Next week we will work on the time machine to be able to send back the RPN of the math equation to past me so I can use the calculator correctly.

#Input:

A string that represents a math equation to be solved. We will allow the 4 functions, use of () for ordering and thats it. Note white space between characters could be inconsistent.

* Number is a number
* "+" add
* "-" subtract
* "/" divide
* "x" or "*" for multiply
* "(" with a matching ")" for ordering our operations

#Output:

The RPN (reverse polish notation) of the math equation.

#Challenge inputs:

Note: "" marks the limit of string and not meant to be parsed.

```
 "0+1"
 "20-18"
 " 3        x        1   "
 " 100    /        25"
```

```
" 5000        /  ((1+1) / 2) * 1000"
" 10 * 6 x 10 / 100"
" (1 + 7 x 7) / 5 - 3  "
"10000 / ( 9 x 9 + 20 -1)-92"
"4+5 * (333x3 /      9-110                              )"
" 0 x (2000 / 4 * 5 / 1 * (1 x 10))"
```

#Additional Challenge:

Since you already got RPN - solve the equations.

Title: [2015-03-18] Challenge #206 [Intermediate] Maximizing Crop Irrigation
Text: # Description

You run a farm which isn't doing so well. Your crops that you planted aren't coming up, and your bills are bigger than your expected proceeds. So, you have to conserve water and focus instead on the plants that are growing. You have a center pivot watering system which has a rotating sprinkler around a central pivot, creating a circular watered area. For this challenge, you just have to decide where to locate it based on this year's crops.

Some notes:

* Because this is a simple grid, we're only dealing with integers in this challenge.
* For partially covered squares, round down: the sprinkler covers the crop if the distance from the sprinkler is less than or equal to the sprinklers radius.
* If you place the sprinkler on a square with a crop, you destroy the crop so handle accordingly (e.g. deduct 1 from the calculation).
* If in the event you find two or more placements that yield identical scores, pick any one of them (or even emit them all if you so choose), this is entirely possible.

# Input Description

You'll be given three integers (h w r) which correspond to the number of rows (h) and columns (w) for the ASCII map (respectively) and then the radius (r) of the watering sprinkler. The ASCII map will have a "." for no crop planted and an "x" for a growing crop.

# Output Description

You should emit the coordinates (0-indexed) of the row and column showing where to place the center of the sprinkler. Your coordinates should be integers.

# Challenge Input

```
51 91 9
......x...x....x............x............x.................................x...............
.........x...........x...................x.....x...........xx.............x................
...........x.................x.x............x..........................x................x..
......x...x.....................x.....x....x.........x......x.......x...x..................
.x...x.....x................xx...........................x.....xx.....x............x.......
.....xx.......x..x........x.............xx........x.......x.....................x.......x..
...x..x.x..x......x..............................................................x...x.....
........x..x......x......x...x.x....x.......x........x..x...........x.x...x..........xx....
...............x.x....x...........x......x.............x..........................x........
...................x........x..............................................................
..x.x.....................................x..x.x......x......x.............................
......x.............................................................................x..x...
......x....x...............x...............................................................
............x.............x.............................x...............x................x.
..xx........xx............x...x......................x.....................................
........x........xx..............x.....................x.x.......x........................x
```

```
.......x....................xx..............................................................
............x...x.........x...xx...............x............................................
..............................x...............xx..x...........x....x........x...x.......x.x.
..........x.......................x......................................x..................
...xx..x.x..................x.........................x.....................x..x.......x....
.............xx..........x...............x.......................x.........x.........x....x.
...............................x......................x.x...................................
...................x....x.............................x...x.......x.............x....x.x....
.x.xx.........................x....................................x.....x.......xx..........
.......x....................................................................................
.........x.....x.................x..................x...x.......x..................x........
.......x................x.x....................................x...xx....x.....x...x........
...............................................x..................x.........................
............................x........x.......x.............................................x
..x.............x.....x...............x............x...x....x...x...........................
........................xx.................x...................x...................x.......x
.x.x.............x....x..................................x...........x..x..........x.....x..
...x..x..x.......................x...........x..........x.............xxx....x..........x...
............................................................x...............................
x......x.....x................x...............x.....................................x.......
..x...........................x............x..........x....x...............................x
........................x.......xx...............x...x.x.................x..x............x..
x................x.......x........x..............................x.x.x...................x.x
.......................x...x........................................................x.......
.x..................x.....x...........................................x...........x.........
.x...................x........x..................x..........xx..................x..x........
.x..........x...x............................x.x....................x..x.......x............
.............x...x..................x.................x..x.x.....xxx..x...xx..x.............
.x...................x.x....x...x.................x..............................x.....x....
......................x.x........x...........x....................................x......x..
................x.....................................x....x....x......x..............x..x..
......x..........................................x..x......x.x.......x......................
.x...............................x..........x.x....x.................x......................
x..x...........x..x.x...x...........................................x..............xx.......
..xx......x.......x.x.................x.......................................x.............
```

# Bonus

Emit the map with the circle your program calculated drawn.

Title: [2015-03-25] Challenge #207 [Intermediate] Bioinformatics 2: DNA Restriction Enzymes
Text: Continuing with our bioinformatics theme today. If you like these sorts of problems, I encourage you to check out Project Rosalind (their site seems back up): http://rosalind.info/

# Description

Restriction enzymes are DNA-cutting enzymes found in bacteria (and harvested from them for use). Because they cut within the molecule, they are often called restriction endonucleases. In order to be able to sequence DNA, it is first necessary to cut it into smaller fragments. For precise molecular biology work, what is needed is a way to cleave the DNA molecule at a few specifically-located sites so that a small set of homogeneous fragments are produced. The tools for this are the restriction endonucleases. The rarer the site it recognizes, the smaller the number of pieces produced by a given restriction endonuclease.

For more information on how these enzymes work, including a great visualization of how they cut, have a look here: http://users.rcn.com/jkimball.ma.ultranet/BiologyPages/R/RestrictionEnzymes.html

These enzymes can cleave the DNA at a site that leaves both strands the same length. This is called a "blunt" end because of this and can be visualized like this:

```
5'-GG  +CC-3'
3'-CC   GG-5'
```

Other DNA restriction enzymes cleave the ends at different lengths, although it's symmetrical about the central axis. These are called "sticky" ends, and here's a simple visualization of one of those cuts:

```
5'-ATCTGACT      + GATGCGTATGCT-3'
3'-TAGACTGACTACG       CATACGA-5'
```

In both cases the two strands are cut at a point of symmetry (the upper and lower strands are symmetrical if rotated).

Today your challenge is to write a program that can recognize the locations where various enzymes will cut DNA.

# Input

You'll be given a list of DNA restriction enzymes and their recognition site and where the cut occurs. The input will be structured as enzyme name, if the enzyme makes a "sticky" or "blunt" end cut, the DNA recognition sequence and the position of the cut marked with a caret ("\^"). For the sticky ends, you should assume the mirror image of the complementary strand gets the same cut, leaving one of the strands to overhang (hence it's "sticky").

```
BamHI sticky G^GATCC
HaeIII blunt GG^CC
HindIII sticky A^AGCTT
```

Then you'll be given a DNA sequence and be asked to cut it with the listed enzymes. For sake of convenience, the DNA sequence is broken into blocks of 10 bases at a time and in lengths of 6 blocks per row. You should merge these together and drop the first column of digits.

This sequence was taken from the genome of *Enterobacteria phage phiX174 sensu lato* http://www.genome.jp/dbget-bin/www_bget?refseq+NC_001422 and modified for this challenge.

```
  1 gagttttatc gcttccatga cgcagaagtt aacactttcg gatatttctg atgagtcgaa
 61 aaattatctt gataaagcag gaattactac tgcttgttta cgaattaaat cgaagtggac
121 tgctggcgga aaatgagaaa attcgaccta tccttgcgca gctcgagaag ctcttacttt
181 gcgacctttc gccatcaact aacgattctg tcaaaaactg acgcgttgga tgaggagaag
241 tggcttaata tgcttggcac gttcgtcaag gactggttta gatatgagtc acattttgtt
301 catggtagag attctcttgt tgacatttta aaagagcgtg gattactatc tgagtccgat
361 gctgttcaac cactaatagg taagaaatca tgagtcaagt tactgaacaa tccgtacgtt
421 tccagaccgc tttggcctct attaagctta ttcaggcttc tgccgttttg gatttaaccg
481 aagatgattt cgattttctg acgagtaaca aagtttggat ccctactgac cgctctcgtg
541 ctcgtcgctg cgttgaggct tgcgtttatg gtacgctgga ctttgtggga taccctcgct
601 ttcctgctcc tgttgagttt attgctgccg tcaaagctta ttatgttcat cccgtcaaca
661 ttcaaacggc ctgtctcatc atggaaggcg ctgaatttac ggaaaacatt attaatggcg
721 tcgagcgtcc ggttaaagcc gctgaattgt tcgcgtttac cttgcgtgta cgcgcaggaa
781 acactgacgt tcttactgac gcagaagaaa acgtgcgtca aaaattacgt gcggaaggag
841 tgatgtaatg tctaaaggta aaaaacgttc tggcgctcgc cctggtcgtc cgcagccgtt
```

# Output

Your program should emit the name of the enzyme, the cut positions for that enzyme, and the contextualized cut. For the above the solution would be:

```
BamHI 517 agttt[g gatcc]ctactg
HaeIII 435 gcttt[gg cc]tctattaa
HaeIII 669 caaac[gg cc]tgtctcat
HindIII 444 ctatt[a agctt]attcag
HindIII 634 cgtca[a agctt]attatg
```

# Bonus

Write some code that identifies any and all symmetrical points along the DNA sequence where an enzyme (not just the three listed) could cut. These should be even-length palindromes between 4 and 10 bases long.

Title: [2015-04-01] Challenge #208 [Intermediate] ASCII Gradient Generator
Text: # [](#IntermediateIcon) _(Intermediate)_: ASCII Gradient Generator

A linear colour gradient is where an image transitions through a range of colours, [like this](http://i.imgur.com/IPwnI8X.png). A gradient doesn't need to be directly horizontal or vertical - it can be [diagonal](http://i.imgur.com/D4trkEk.png) too, or only be [longer or shorter](http://i.imgur.com/8CHx95i.png) than usual. It can also cycle through [as many colours as you like](http://i.imgur.com/Br3xwXM.png).

A radial colour gradient is a similar concept, except the colours move [radially outwards like this](http://i.imgur.com/C6SE6m3.png), rather than linearly across. Radial gradients can also be in [different positions or with different colours](http://i.imgur.com/S19EOu3.png).

To describe a gradient, you need two things - the colours in it, and its location. Describing the location of a radial gradient is easy: for a radial gradient [like this](http://i.imgur.com/dTvfj7f.png), you only need to know the center of the gradient (the red dot), and the radius from the center at which the gradient finishes (`r`). To locate a linear gradient [like this](http://i.imgur.com/kyZTQnK.png), you need to know two points - the start (red) and end (green) location. The gradient colours run perpendicular to the line joining the start and end points.

Today, we won't be dealing with colours. Instead, we'll be dealing with characters on the screen. You'll accept the parameters of a gradient, and you'll output the displayed gradient.

# Formal Inputs and Outputs

## Input Description

You will first accept the size of the output display, as a width and height in characters, like this:

```
40 30
```

This corresponds to a grid 40 across and 30 down, like this:

```
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
```

```
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
........................................
```

The grid follows **screen space**, so the **top-left** corner is position **(0, 0)**.

Next, you will accept the characters that make up the gradient 'colours', from start to finish (or from inside to outside, for a radial gradient), like this: (note the space at the start)

```
 .,:;xX&@
```

Any points outside the gradient will have the first/last character, depending on which side of the gradient they're on.

After this, you will accept the parameters of the gradient. This may take one of two forms:

* For a **radial** gradient, the next line will look like this:
 `radial x y r`
 Where **(`x`, `y`)** is the center of the gradient, and **`r`** is the radius of the gradient, both in pixels.

* For a **linear** gradient, the next line will look like this:
 `linear x1 y1 x2 y2`
 Where **(`x1`, `y1`)** is the start point of the gradient, and **(`x2`, `y2`)** is the end point of the gradient, both in pixel measure.

## Output Description

You are to display the given gradient on a grid with the given size, like this:

```
@@@@@@@@@@@&&&&&XXXXXXXXX&&&&&@@@@@@@@@@
@@@@@@@@@@&&&&XXXXXXXXXXXXX&&&&@@@@@@@@@
@@@@@@@@&&&&XXXXXXxxxxxXXXXXX&&&&@@@@@@@
@@@@@@@&&&&XXXXxxxxxxxxxxxXXXX&&&&@@@@@@
@@@@@@@&&&XXXxxxxxx;;;xxxxxxXXX&&&@@@@@@
@@@@@@&&&XXXxxxx;;;;;;;;;xxxxXXX&&&@@@@@
@@@@@&&&XXXxxx;;;;;;;;;;;;;xxxXXX&&&@@@@
@@@@@&&XXXxxx;;;;:::::::;;;;xxxXXX&&@@@@
@@@@&&&XXxxx;;;:::::::::::;;;xxxXX&&&@@@
@@@@&&XXXxx;;;::::,,,,,::::;;;xxXXX&&@@@
@@@&&&XXxxx;;:::,,,,,,,,,:::;;xxxXX&&&@@
@@@&&XXXxx;;;::,,,,...,,,,::;;;xxXXX&&@@
@@@&&XXXxx;;:::,,.......,,:::;;xxXXX&&@@
@@@&&XXxxx;;::,,,... ...,,,::;;xxxXX&&@@
@@@&&XXxx;;;::,,...   ...,,::;;;xxXX&&@@
@@@&&XXxx;;;::,,..     ..,,::;;;xxXX&&@@
@@@&&XXxx;;;::,,...   ...,,::;;;xxXX&&@@
@@@&&XXxxx;;::,,,... ...,,,::;;xxxXX&&@@
@@@&&XXXxx;;:::,,.......,,:::;;xxXXX&&@@
@@@&&XXXxx;;;::,,,,...,,,,::;;;xxXXX&&@@
@@@&&&XXxxx;;:::,,,,,,,,,:::;;xxxXX&&&@@
@@@@&&XXXxx;;;::::,,,,,::::;;;xxXXX&&@@@
@@@@&&&XXxxx;;;:::::::::::;;;xxxXX&&&@@@
```

```
@@@@@&&XXXxxx;;;;:::::::;;;;xxxXXX&&@@@@
@@@@@&&&XXXxxx;;;;;;;;;;;;;xxxXXX&&&@@@@
@@@@@@&&&XXXxxxx;;;;;;;;;xxxxXXX&&&@@@@@
@@@@@@@&&&XXXxxxxxx;;;xxxxxxXXX&&&@@@@@@
@@@@@@@&&&&XXXXxxxxxxxxxxxXXXX&&&&@@@@@@
@@@@@@@@&&&&XXXXXXxxxxxXXXXXX&&&&@@@@@@@
@@@@@@@@@@&&&&XXXXXXXXXXXXX&&&&@@@@@@@@@
```

# Sample Inputs and Outputs

## Gradient 1

### Input

```
40 30
 .,:;xX&@
radial 20 15 20
```

### Output

(shown above, in **Output Description**)

## Gradient 2

Notice how the colours appear in the reverse order, as the end point is to the *left* of the start point.

### Input

```
60 30
 '"^+$
linear 30 30 0 0
```

### Output

```
$$$$$$$$$$$++++++++++^^^^^^^^^^""""""""""''''''''''
$$$$$$$$$$++++++++++^^^^^^^^^^""""""""""''''''''''
$$$$$$$$$++++++++++^^^^^^^^^^""""""""""''''''''''
$$$$$$$$++++++++++^^^^^^^^^^""""""""""''''''''''
$$$$$$$++++++++++^^^^^^^^^^""""""""""''''''''''
$$$$$$++++++++++^^^^^^^^^^""""""""""''''''''''
$$$$$++++++++++^^^^^^^^^^""""""""""''''''''''
$$$$++++++++++^^^^^^^^^^""""""""""''''''''''
$$$++++++++++^^^^^^^^^^""""""""""''''''''''
$$++++++++++^^^^^^^^^^""""""""""''''''''''
$++++++++++^^^^^^^^^^""""""""""''''''''''
++++++++++^^^^^^^^^^""""""""""''''''''''
+++++++++^^^^^^^^^^""""""""""''''''''''
++++++++^^^^^^^^^^""""""""""''''''''''
+++++++^^^^^^^^^^""""""""""''''''''''
++++++^^^^^^^^^^""""""""""''''''''''
+++++^^^^^^^^^^""""""""""''''''''''
++++^^^^^^^^^^""""""""""''''''''''
+++^^^^^^^^^^""""""""""''''''''''
++^^^^^^^^^^""""""""""''''''''''
+^^^^^^^^^^""""""""""''''''''''
^^^^^^^^^^""""""""""''''''''''
^^^^^^^^^""""""""""''''''''''
^^^^^^^^""""""""""''''''''''
```

```
^^^^^^^""""""""""'''''''''
^^^^^^""""""""""'''''''''
^^^^^""""""""""'''''''''
^^^^""""""""""'''''''''
^^^""""""""""'''''''''
^^""""""""""'''''''''
```

## Gradient 3
The gradient start/end/centre points don't have to be inside the grid!

### Input

```
40 40
aaabcccdeeefggg
radial -10 20 60
```

### Output

```
ccccccccccdddddeeeeeeeeeeeeeeeffffgggggg
cccccccccccdddddeeeeeeeeeeeeeefffffggggg
ccccccccccccdddddeeeeeeeeeeeeeeffffggggg
cccccccccccccdddddeeeeeeeeeeeeeffffggggg
cccccccccccccdddddeeeeeeeeeeeeefffffgggg
ccccccccccccccdddddeeeeeeeeeeeeeffffgggg
ccccccccccccccccddddeeeeeeeeeeeeeffffgggg
cccccccccccccccdddddeeeeeeeeeeeeeffffggg
bcccccccccccccccddddeeeeeeeeeeeeeffffggg
bbccccccccccccccdddddeeeeeeeeeeeeffffggg
bbbccccccccccccccddddeeeeeeeeeeeeffffggg
bbbbcccccccccccccddddeeeeeeeeeeeeeffffgg
bbbbcccccccccccccddddeeeeeeeeeeeeeffffgg
bbbbbcccccccccccccddddeeeeeeeeeeeeffffgg
abbbbcccccccccccccddddeeeeeeeeeeeeffffgg
abbbbbccccccccccccddddeeeeeeeeeeeeffffgg
aabbbbccccccccccccddddeeeeeeeeeeeeffffgg
aabbbbccccccccccccddddeeeeeeeeeeeeffffgg
aabbbbccccccccccccddddeeeeeeeeeeeeffffgg
aabbbbccccccccccccddddeeeeeeeeeeeeffffgg
aabbbbccccccccccccddddeeeeeeeeeeeeffffgg
aabbbbccccccccccccddddeeeeeeeeeeeeffffgg
aabbbbccccccccccccddddeeeeeeeeeeeeffffgg
aabbbbccccccccccccddddeeeeeeeeeeeeffffgg
aabbbbccccccccccccddddeeeeeeeeeeeeffffgg
abbbbbccccccccccccddddeeeeeeeeeeeeffffgg
abbbbcccccccccccccddddeeeeeeeeeeeeffffgg
bbbbbcccccccccccccddddeeeeeeeeeeeeffffgg
bbbbcccccccccccccddddeeeeeeeeeeeeeffffgg
bbbbcccccccccccccddddeeeeeeeeeeeeeffffgg
bbbccccccccccccccddddeeeeeeeeeeeeffffggg
bbccccccccccccccdddddeeeeeeeeeeeeffffggg
bcccccccccccccccddddeeeeeeeeeeeeeffffggg
cccccccccccccccdddddeeeeeeeeeeeeeffffggg
ccccccccccccccccddddeeeeeeeeeeeeeffffgggg
ccccccccccccccdddddeeeeeeeeeeeeeffffgggg
cccccccccccccdddddeeeeeeeeeeeeefffffgggg
cccccccccccccdddddeeeeeeeeeeeeeffffggggg
ccccccccccccdddddeeeeeeeeeeeeeeffffggggg
cccccccccccdddddeeeeeeeeeeeeeefffffggggg
```

Title: [2015-04-08] Challenge #209 [Intermediate] Packing a Sentence in a Box
Text: # Description

You're moving, and you have a bunch of sentences to pack up. To accomplish this, you'll be using a small program you should write to pack these sentences efficiently into a box for shipping. Leave no unused space, you have a lot of sentences to pack and you don't want to waste precious shipping space.

For this challenge you're free to choose any legal dimensions of a rectangle, and you're free to start in any position you wish. Your program (and thus your output) should walk the grid to adjacent squares using only left, right, up, down (no diagonal moves allowed).

# Input

You'll be given a sentence to pack into a box

```
EVERYWHERE IS WITHIN WALKING DISTANCE IF YOU HAVE THE TIME
```

# Output

Your program should emit the starting position (column and row, 1-indexed) for the sentence, and then the box with the sentence packed into it. The sentence must be packed in the original word order with only spaces removed. You can chose your own box dimensions. The above example is a 49 character sentence (minus spaces), so that's a 7x7 box. Here's one possible solution:

```
4 4
E    T    I    M    E    D    I
H    W    S    I    E    G    S
T    I    E    V    R    N    T
E    T    R    E    E    I    A
V    H    Y    W    H    K    N
A    I    N    W    A    L    C
H    U    O    Y    F    I    E
```

# Challenge Input

```
IT IS RAINING CATS AND DOGS OUT THERE
```

# Challenge Output

Here's one possible solution

```
1 1
I    T    I    N    I
E    I    A    G    N
R    S    R    C    A
E    G    O    D    T
H    S    O    D    S
T    T    U    N    A
```

Title: [2015-04-15] Challenge #210 [Intermediate] Drawing a gradient
Text: #Description

One of the most basic tools in graphic design toolbox is the "gradient": a smooth transtion from one color to another. They are remarkably useful and are available in virtually every graphic design program and graphic programming library, and even natively in design languages like CSS and SVG.

Your task today is to make a program that can generate these wonderful gradients, and then either draw it to the screen or save it as an image file. You will get as inputs pixel dimensions for the size of the gradient, and the two colors that the gradient should transition between.

NOTE: As I said, there are many imaging libraries that provide this functionality for you, usually in some function called `drawGradient()` or something similar. You are *strongly encouraged* not to use functions like this, the spirit of this challenge is that you should figure out how to calculate the gradient (and thus the individual pixel colors) yourself.

This isn't an ironclad rule, and if you really can't think of any way to do this yourself, then it's fine to submit your solution using one of these functions. I encourage you to try, though.

It is, however, perfectly acceptable to use a library to save your pixels in whatever format you like.

#Formal Inputs &amp; Outputs

##Input description

Your input will consist of three lines. The first line contains two numbers which is the width and the height of the resulting gradient. The other two lines consist of three numbers between 0 and 255 representing the colors that
the gradient should transition between. The first color should be on the left edge of the image, the second color should be on the right edge of the image.

So, for instance, the input

```
500 100
255 255 0
0 0 255
```

means that you should draw a 500x100 gradient that transitions from yellow on the left side to blue on the right side.

##Output description

You can either choose to draw your gradient to the screen or save it as an image file. You can choose whatever image format you want, though it should preferably a lossless format like PNG.

If you don't wish to tangle with libraries that output PNG images, I recommend checking out the [Netpbm](http://en.wikipedia.org/wiki/Netpbm) format, which is a very easy format to output images in. There's even a [dailyprogrammer challenge](https://www.reddit.com/r/dailyprogrammer/comments/2ba3g3/7212014_challenge_172_easy/) that can help you out.

Regardless of your chosen method of output, I highly encourage you to upload your resulting images to [imgur](http://imgur.com) so that the rest of us can see the product of your hard work! If you chose to output your image to the screen, you can take a screenshot and crop the gradient out.

#Example inputs & outputs

#Input

```
500 100
255 255 0
0 0 255
```

#Output

[This image](http://i.imgur.com/LNBRYhr.png)

#Challenge inputs

```
1000 100
```

```
204 119 34
1 66 37
```

Those two colors are Ochre and British Racing Green, my two favorite colors. Use those as a challenge input, or pick your own two favorite colors!

#Bonus

We often see people solving these problems in weird languages here at /r/dailyprogrammer, and this bonus is for all you crazy people:

Solve this problem in [brainfuck](http://en.wikipedia.org/wiki/Brainfuck). You don't have to read the values from input, you can "hard-code" the colors and dimensions in your program. You can pick whatever colors and dimensions you like, as long as both the width and the height is larger than 100 pixels. You can also output the image in whatever format you want (I imagine that one of the binary Netpbm formats will be the easiest). Good luck!

Title: [2015-04-22] Challenge #211 [Intermediate] Ogre Maze
Text: #Description:

Today we are going to solve a maze. What? Again? Come on, Simpsons did it. Yah okay so we always pick a hero to walk a maze. This time our hero is an Ogre.

An ogre is large. Your run of the mill hero "@" takes up a 1x1 spot. Easy. But our beloved hero today is an ogre.

```
@@
@@
```

Ogres take up a 2x2 space instead of a 1x1. This makes navigating a maze tougher as you have to handle the bigger ogre.

So I will give you a layout of a swamp. (Ogres navigate swamps while puny heroes navigate caves. That's the unwritten rules of maze challenges) You will find the path (if possible) for
the ogre to walk to his gold.

#Input:

You will read in a swamp. The swamp is laid out in 10x10 spaces. Each space can be the following:

* . - empty spot
* @ - 1/4th of the 2x2 ogre
* $ - the ogre's gold
* O - sink hole - the ogre cannot touch these. All 2x2 of the Ogre manages to fall down one of these (even if it is a 1x1 spot too. Don't be bothered by this - think of it as a "wall" but in a swamp we call them sink holes)

#Output:

You will navigate the swamp. If you find a path you will display the solution of all the spaces the ogre will occupy to get to his gold. Use a "&" symbol to show the muddy path created by the ogre to reach his gold. If there is no path at all then you will output "No Path"

#Example Input 1:

```
@@........
@@O.......
.....O.O..
..........
..O.O.....
```

```
    ..O....O.O
    .O........
    ..........
    .....OO...
    .........$
```

#Example Output 1:

```
            &&.&&&&&&&
            &&O&&&&&&&
            &&&&&O.O&&
            &&&&&&&&&&
            ..O.O&&&&&
            ..O..&&O.O
            .O...&&&&.
            .....&&&&.
            .....OO&&&
            .......&&&
```

#Example Input 2:

```
            @@........
            @@O.......
            .....O.O..
            ..........
            ..O.O.....
            ..O....O.O
            .O........
            ..........
            .....OO.O.
            .........$
```

#Example Output 2:

No Path

#FAQ (Will update with answers here)

* Q: Does path have to be shortest Path.
* A: No.

-

* Q: There could be a few different paths. Which one do I output?
* A: The first one that works. Answers will vary based on how people solve it.

-

* Q: My output should show all the spots the Ogre moves too or just the optimal path?
* A: The ogre will hit dead ends. But only show the optimal path and not all his dead ends. Think of this as a GPS Tom-Tom guide for the Ogre so he uses the program to find his gold. TIL Ogres subscribe to /r/dailyprogrammer. (And use the internet....)

#Challenge Input 1:

```
   $.O...O...
            ...O......
            ..........
```

```
O..O..O...
..........
O..O..O...
..........
......OO..
O..O....@@
........@@
```

#Challenge Input 2:

```
.@@.....O.
.@@.......
..O..O....
.......O..
...O......
..........
.......O.O
...O.O....
.......O..
.........$
```

#Bonus:

For those seeking more challenge. Instead of using input swamps you will generate a swamp. Place the Ogre randomly. Place his gold randomly. Generate sinkholes based on the size of the swamp.

For example you are given N for a NxN swamp to generate. Generate a random swamp and apply your solution to it. The exact design/algorithm for random generation I leave it for you to tinker with. I suggest start with like 15% of the swamp spots are sinkholes and go up or down based on your results. (So you get paths and not always No Path)

Title: [2015-04-29] Challenge #212 [Intermediate] Animal Guess Game
Text: #Description:

There exists a classic game which I knew by the name of "Animal". The computer would ask you to think of an animal. If would then ask a bunch of questions that could be answered with a Yes or No. It would then make a guess of what animal you are thinking of. If the computer was right the program ended with smug satisfaction. If the program was wrong it would ask you type in a specific Yes/No question about your Animal. It would then update its library of animals to include it. As it already had a bunch of yes/no questions it would just add the final one to that animal.

As you played this game it would learn. The more you played the more animals it learned. it got better. You taught this program.

For today's challenge we will implement this game.

#Input:

The program will display an intro message and then just ask a series of yes/no questions. How you implement this interface I leave the design to you. It must prompt you with questions and you must be able to answer yes/no.

#Output:

The program will have an intro message welcoming you to the game and then ask you to think of an animal and then proceed to start asking questions once you prompt you are ready.

For this challenge the exact design and text I leave for you to develop as part of the challenge.

The computer will continue to output questions for yes/no responses. Eventually the computer will take a guess. You can then tell the computer by a yes/no if it was a correct guess. If the computer is correct you may output a success message for the computer and exit. if the computer was wrong in the guess picked you will be asked to enter your animal and a yes/no question string that would answer a "yes". The computer program will prompt for this data and you must supply it. You are teaching the program.

Again exact design and words I leave to you to design. I give a rough example below in examples.

#AI:

The requirements for this game is a learning game. Every time you play it must load contain all previous game learning. You therefore must find a way to design and implement this.

The tricky part is what questions to ask. I leave it to you and your design to develop those initial or base questions and then using the learned questions.

#Example of Play 1:

```
Welcome to Animal Guess. Please think of an Animal and type "y" to proceed --> y

Is your animal a mammal? --> y
Is your animal a swimmer? --> y
Is your animal grey? --> y

I think your animal is a grey whale. Am I correct? --> n

Oh well. please help me learn.
What is the name of your animal-> dolphin
What is a unique question that answers yes for dolphin -> Does your animal have high intelligence

Thank  you for teaching me.
```

#Example of Play 2:

```
Welcome to Animal Guess. Please think of an Animal and type "y" to proceed --> y

Is your animal a mammal? --> y
Is your animal a swimmer? --> n
Is your animal grey? --> y

I think your animal is an elephant. Am I correct? --> y

It is okay to feel bad you did not stump me. I am a computer. :)
Thank you for playing!
```

Title: [2015-05-06] Challenge #213 [Intermediate] The Lazy Typist
Text: # [](#IntermediateIcon) _(Intermediate)_: The Lazy Typist

We've all had a night where we're so lazy that we actively avoid moving our hands around on the keyboard. In today's challenge, we'll be given a sentence to type, and we'll work out a minimal-effort way of typing that string (ie. minimize how much the hand moves), using a basic QWERTY keyboard layout - the keys supported are the letters A to Z, shift, and space - in this arrangement:

```
qwertyuiop
asdfghjkl
^zxcvbnm ^
  #####
```

The only letters that can be typed are upper-case and lower-case letters, and space. Our typist is quite inefficient: they move their fingers around the keyboard, [hunting for keys one by one](http://en.wikipedia.org/wiki/Typing#Hunt_and_peck), so the user only uses one finger of each hand.

The user may start with both hands on any key, and may move either hand to the next key. The main important things to remember are:

* The user may move to any of the five `#` (space) positions to type a space.
* Two hands are required to type a capital letter - one must go to a shift key. Which hand goes to which key is up to your program to decide, but the same hand can't press both the shift key and the letter.

As a score of laziness, you'll also need to work out the total Manhattan distance (x + y) moved by the hands. We'll call this total distance the *effort*.

# Formal Inputs and Outputs

## Input Description

Enter a sentence, consisting of only upper-case, lower-case and spaces, like so:

```
The quick brown Fox
```

## Output Description

Display all the key presses, along with the hand that presses the key, and the distance that the hand moves, for example:

```
Shift: Use left hand
T: Use right hand
H: Move right hand from T (effort: 2)
E: Move left hand from Shift (effort: 4)
Space: Move right hand from H (effort: 2)
Q: Move left hand from E (effort: 2)
U: Move right hand from Space (effort: 4)
I: Move right hand from U (effort: 1)
C: Move left hand from Q (effort: 5)
K: Move right hand from I (effort: 1)
Space: Move left hand from C (effort: 1)
B: Move left hand from Space (effort: 3)
R: Move left hand from B (effort: 4)
O: Move right hand from K (effort: 2)
W: Move left hand from R (effort: 2)
N: Move right hand from O (effort: 4)
Space: Move right hand from N (effort: 1)
Shift: Move left hand from W (effort: 3)
F: Move right hand from Space (effort: 5)
O: Move right hand from F (effort: 6)
X: Move left hand from Shift (effort: 2)
```

Finally, display the total effort:

```
Total effort: 54
```

You may be able to find a more efficient way of doing this - you only need to find a heuristic solution. If a hand is already over the key which it needs to press, the distance and effort is (obviously) zero.   Shift: Use left hand

```
Q: Use right hand
Shift: Use left hand again
P: Move right hand from Q (effort: 9)
G: Move left hand from Shift (effort: 5)
```

```
I: Move right hand from P (effort: 2)
Z: Move left hand from G (effort: 4)
M: Move right hand from I (effort: 2)
Space: Move right hand from M (effort: 1)
Shift: Move left hand from Z (effort: 1)
Q: Move right hand from Space (effort: 10)
Shift: Use left hand again
F: Move right hand from Q (effort: 4)
P: Move right hand from F (effort: 7)
Shift: Use left hand again
R: Move right hand from P (effort: 6)
Shift: Use left hand again
K: Move right hand from R (effort: 5)
B: Move right hand from K (effort: 3)
I: Move right hand from B (effort: 4)
Space: Move right hand from I (effort: 3)
Shift: Use left hand again
Q: Move right hand from Space (effort: 10)
Y: Move right hand from Q (effort: 5)
C: Move left hand from Shift (effort: 3)
N: Move left hand from C (effort: 3)
Total effort: 87
```

# Sample Inputs and Outputs

(All of these sample outputs are calculated with a nearest-neighbour approach. Your solution might be better!)

## Sample 1

### Input

```
hello world
```

### Output

```
H: Use left hand
E: Use right hand
L: Move left hand from H (effort: 3)
L: Use left hand again
O: Move left hand from L (effort: 1)
Space: Move left hand from O (effort: 4)
W: Move right hand from E (effort: 1)
O: Move left hand from Space (effort: 4)
R: Move right hand from W (effort: 2)
L: Move left hand from O (effort: 1)
D: Move right hand from R (effort: 2)
Total effort: 18
```

## Sample 2

### Input

```
qpalzm woskxn
```

### Output

```
Q: Use left hand
P: Use right hand
```

```
A: Move left hand from Q (effort: 1)
L: Move right hand from P (effort: 2)
Z: Move left hand from A (effort: 2)
M: Move right hand from L (effort: 2)
Space: Move right hand from M (effort: 1)
W: Move left hand from Z (effort: 2)
O: Move right hand from Space (effort: 4)
S: Move left hand from W (effort: 1)
K: Move right hand from O (effort: 2)
X: Move left hand from S (effort: 2)
N: Move right hand from K (effort: 2)
Total effort: 21
```

## Sample 3

### Input

```
Hello there DailyProgrammers
```

### Output

```
Shift: Use left hand
H: Use right hand
E: Move left hand from Shift (effort: 4)
L: Move right hand from H (effort: 3)
L: Use right hand again
O: Move right hand from L (effort: 1)
Space: Move left hand from E (effort: 4)
T: Move left hand from Space (effort: 4)
H: Move left hand from T (effort: 2)
E: Move left hand from H (effort: 4)
R: Move left hand from E (effort: 1)
E: Move left hand from R (effort: 1)
Space: Move left hand from E (effort: 4)
Shift: Move right hand from O (effort: 3)
D: Move left hand from Space (effort: 3)
A: Move left hand from D (effort: 2)
I: Move right hand from Shift (effort: 4)
L: Move right hand from I (effort: 2)
Y: Move right hand from L (effort: 4)
Shift: Move left hand from A (effort: 1)
P: Move right hand from Y (effort: 4)
R: Move left hand from Shift (effort: 5)
O: Move right hand from P (effort: 1)
G: Move left hand from R (effort: 2)
R: Move left hand from G (effort: 2)
A: Move left hand from R (effort: 4)
M: Move right hand from O (effort: 3)
M: Use right hand again
E: Move left hand from A (effort: 3)
R: Move left hand from E (effort: 1)
S: Move left hand from R (effort: 3)
Total effort: 75
```

## Sample 4

### Input

```
QPgizm QFpRKbi Qycn
```

### Output

```
Shift: Use left hand
Q: Use right hand
Shift: Use left hand again
P: Move right hand from Q (effort: 9)
G: Move left hand from Shift (effort: 5)
I: Move right hand from P (effort: 2)
Z: Move left hand from G (effort: 4)
M: Move right hand from I (effort: 2)
Space: Move right hand from M (effort: 1)
Shift: Move left hand from Z (effort: 1)
Q: Move right hand from Space (effort: 10)
Shift: Use left hand again
F: Move right hand from Q (effort: 4)
P: Move right hand from F (effort: 7)
Shift: Use left hand again
R: Move right hand from P (effort: 6)
Shift: Use left hand again
K: Move right hand from R (effort: 5)
B: Move right hand from K (effort: 3)
I: Move right hand from B (effort: 4)
Space: Move right hand from I (effort: 3)
Shift: Use left hand again
Q: Move right hand from Space (effort: 10)
Y: Move right hand from Q (effort: 5)
C: Move left hand from Shift (effort: 3)
N: Move left hand from C (effort: 3)
Total effort: 87
```

Title: [2015-05-13] Challenge #214 [Intermediate] Pile of Paper
Text: # Description

Have you ever layered colored sticky notes in interesting patterns in order to make pictures? You can create surprisingly complex pictures you can make out of square/rectangular pieces of paper. An interesting question about these pictures, though, is: what area of each color is actually showing? We will simulate this situation and answer that question.

Start with a sheet of the base color 0 (colors are represented by single integers) of some specified size. Let's suppose we have a sheet of size 20x10, of color 0. This will serve as our "canvas", and first input:

```
20 10
```

We then place other colored sheets on top of it by specifying their color (as an integer), the (x, y) coordinates of their top left corner, and their width/height measurements. For simplicity's sake, all sheets are oriented in the same orthogonal manner (none of them are tilted). Some example input:

```
1 5 5 10 3
2 0 0 7 7
```

This is interpreted as:

- Sheet of color `1` with top left corner at `(5, 5)`, with a width of `10` and height of `3`.
- Sheet of color `2` with top left corner at `(0,0)`, with a width of `7` and height of `7`.

Note that multiple sheets *may* have the same color. Color is not unique per sheet.

Placing the first sheet would result in a canvas that looks like this:

```
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000000000000
00000111111111100000
00000111111111100000
00000111111111100000
00000000000000000000
00000000000000000000
```

Layering the second one on top would look like this:

```
22222220000000000000
22222220000000000000
22222220000000000000
22222220000000000000
22222220000000000000
22222221111111100000
22222221111111100000
00000111111111100000
00000000000000000000
00000000000000000000
```

This is the end of the input. The output should answer a single question: *What area of each color is visible after all the sheets have been layered, in order?* It should be formatted as an one-per-line list of colors mapped to their visible areas. In our example, this would be:

```
0 125
1 26
2 49
```

# Sample Input:

```
20 10
1 5 5 10 3
2 0 0 7 7
```

# Sample Output:

```
0 125
1 26
```

    2 49

# Challenge Input

Redditor /u/Blackshell has a bunch of inputs of varying sizes from 100 up to 10000 rectangles up here, with solutions: https://github.com/fsufitch/dailyprogrammer/tree/master/ideas/pile_of_paper

Title: [2015-05-20] Challenge #215 [Intermediate] Validating sorting networks
Text: #Description

When we computer programmers learn all about how computers sort lists of numbers, we are usually taught about sorting algorithms like Quicksort and Heapsort. There is, however, an entirely different model for how computers can sort numbers called [sorting networks](http://en.wikipedia.org/wiki/Sorting_network). Sorting networks are very useful for implementing sorting in hardware, and they have found a use for designing sorting algorithms in GPUs. Today, we are going to explore these strange and fascinating beasts.

In a sorting network, a list of numbers travel down idealized "wires" that are connected with so-called "comparators". Each comparator connects two wires and swaps the values between them if they're out of order (the smaller value going to the top wire, and the larger to the bottom wire). [This image shows the principle clearly](http://upload.wikimedia.org/wikipedia/commons/thumb/e/e8/Sorting-network-comparator-demonstration.svg/467px-Sorting-network-comparator-demonstration.svg.png), and [this image](http://upload.wikimedia.org/wikipedia/commons/thumb/9/9b/SimpleSortingNetworkFullOperation.svg/1024px-SimpleSortingNetworkFullOperation.svg.png) demonstrates a full run of a 4-wire sorting network wtih 5 comparators (both images courtesy of wikipedia, which has an excellent article on [sorting networks](http://en.wikipedia.org/wiki/Sorting_network) if you are interested in learning more). Notice that the list on the right is correctly sorted top to bottom.

It is easy to see why that particular network correctly sorts a list of numbers: the first four comparators "float" the smallest value to the top and "sinks" the largest value to the bottom, and the final comparator sorts out the middle two values.

In general, however, there's often no easy way to tell whether or not a sorting network will actually correctly sort a list of numbers, and the only way to tell is to actually test it. This seems like a daunting task, since for a sorting network with N wires, there's N! (i.e. ["N factorial"](http://en.wikipedia.org/wiki/Factorial)) possible input permutations. That function grows extremely quickly, and become prohibitively large for even N of 14 or 15.

##The zero-one principle
Thankfully, there's a better way, thanks to the so-called "zero-one principle", which is as follows:

> If an N-wire sorting network can correctly sort all 2^N possible sequences of 0's and 1's, it will correctly sort all possible input sequences.

So, instead of having to try and check all N! possible permutations of the input sequence, we can just check all 2^N sequences consisting of 0's and 1's. While this is still exponential, it is far smaller than N!.

Today, you will be given a sorting network and your challenge will be to check whether or not that network will correctly sort all inputs.

#Formal inputs &amp; outputs

##Inputs

The input will first consist of a single line with two numbers on it separated by a space. The first number is the number of wires in the sorting network, and the second number is the total number of comparators on the network.

After that, there will follow a number of lines, each of which describes one comparator. The lines will consist of two numbers separated by a space describing which two wires the comparator connects. The first number will always be smaller than the second number

The "top" wire of the sorting network is wire 0, and the bottom wire is wire N-1. So, for a 16-wire sorting network, the top wire (which will hold the smallest number at the end, hopefully) is wire 0, and the bottom wire is wire 15 (which will hold the highest number at the end, hopefully).

Note that in most sorting networks, several comparators compare numbers in parallel. For the purposes of this problem, you can ignore that and assume that all comparators work in sequence, following the order provided in the input.

##Output

The output will consist of a single line which will either be "Valid network" if the network will indeed sort all sequences correctly, or "Invalid network" if it won't.

#Sample inputs and outputs

##Input 1
This is the example 4-wire, 5-comparator sorting network from the description:

```
4 5
0 2
1 3
0 1
2 3
1 2
```

##Output 1

```
Valid network
```

##Input 2

```
8 19
0 2
1 3
0 1
2 3
1 2
4 6
5 7
4 5
6 7
5 6
0 4
1 5
2 6
3 7
2 4
3 5
1 2
3 4
6 7
```

##Output 2

```
Invalid network
```

#Challenge inputs

##Input 1

[This 16-wire 60-comparator network](https://gist.githubusercontent.com/anonymous/274991a6297f8291716f/raw/ea60f5bcfa2577bdce5f18ffbbaa7d09058f1c7a/challenge1.txt)

##Input 2

[This (slightly different) 16-wire 60-comparator network](https://gist.githubusercontent.com/anonymous/1d74c14d00dff6369db6/raw/58cad39fa8c7980620c24aca681248084e0738c4/challenge2.txt)

Title: [2015-05-27] Challenge #216 [Intermediate] Texas Hold 'Em 2 of 3 - Winning Hand & Know when to fold 'em
Text: #Description :
This continues the week long challenge of Texas Hold 'Em. Today we are going to be challenged with 2 added features.

* AI logic for the computer hands to pick a "fold" option
* Determining the Winning Hand

#AI Logic - When to fold:

I feel this is related to the winning hand which is the 2nd of the two challenges for today. Knowing what a winning hand is helps determine if you should fold. If the CPU determines it doesn't look good it should fold.

The exact logic/method for when to fold it is not so easy. I think there exists many different ways (be it programmed logic or math with probability) to determine it is time to fold.

You will have the freedom and challenge of coming up with code for the AI controlled hands to look at their hands after the flop and the turn cards. Based on your solution for this you will have the AI determine if their hand is not worth pursuing any long and do a "fold". Solutions will vary here. There is no wrong or right way to do this.

Furthermore we will have the ability to see all the cards and check if the logic was a good move or perhaps by also detemining the best hand (regardless if a fold was picked or not)

#Winning Hand and Best hand

Following general rules for Poker we can determine who wins the hand. [List of winning hands in poker](http://en.wikipedia.org/wiki/List_of_poker_hands)

After the river is drawn we will show with our output who wins the hand. During the process of drawing the community cards the AI hands have a chance to enter a "fold" state (see above). The winning hand can never be a CPU who picks the fold option.

We will also pick the "Best Hand" by comparing all hands (even ones that were folded) to check our AI logic. If the Winning hand does not equal the best hand then we see the fold choice was not always optimal.

#Input:

You will use the same input as the Easy part of this challenge. You will ask for how many players 2-8. You will be one of the players and playing against 1-7 random CPU controlled players.

#Output:

We will modify the output to reflect the status of any folds. We will also output who had the winning hand and the method (high card, pair, straight, flush, 3 of a kind, full house, 4 of a kind, etc) We will also note if a folded hand would have won instead if they had not picked the fold option. (See examples below)

#Example 1:

```
    How many players (2-8) ? 3

    Your hand: 2 of Clubs, 5 of Diamonds
    CPU 1 Hand: Ace of Spades, Ace of Hearts
    CPU 2 Hand: King of Clubs, Queen of Clubs

    Flop: 2 of Hearts, 5 of Clubs, Ace of Clubs
    Turn: King of Hearts
    River: Jack of Hearts

    Winner: CPU 1 wins. 3 of a kind.
```

#Example 2:

```
    How many players (2-8) ? 3

    Your hand: 3 of Diamonds, 3 of Spades
    CPU 1: 10 of Diamonds, Jack of Diamonds
    CPU 2: 4 of Clubs, 8 of Hearts

    Flop: Ace of Diamonds, Queen of Diamonds, 9 of Diamonds
    CPU 2 Folds!
    Turn: 7 of Hearts
    River: 4 of Spades

    Winner: CPU 1. Flush.
    Best Hand: CPU 1.
```

#Example 3:

```
    How many players (2-8) ? 3

    Your hand: 2 of Hearts, 8 of Spades
    CPU 1: 4 of Hearts, 6 of Clubs
    CPU 2: Jack of Diamonds, 10 of Hearts

    Flop: 8 of Hearts, Jack of Spades, 10  of Clubs
    CPU 1 Folds!
    Turn: 5 of Hearts
    River: 7 of Hearts

    Winner: CPU 2. Two pair.
    Best Hand: CPU 1. Straight.
```

#Looking ahead

At this point we have Texas Hold Em without money or bets. We can deal cards. We can have the AIs decide to fold and we can determine the winning hand and who had the best hand. The final step on Friday will be to look for trends in running many simulated games to look for trends and patterns. It will really test how good our AI logic is and maybe generate data to help human players see patterns and trends.

Title: [2015-06-03] Challenge #217 [Intermediate] Space Code Breaking
Text: #Description:

The year 2266 we have encountered alien planets who use very simple encryption to send messages. Lucky for us we intercept all these messages and we can break the code.

The problem is the collection of messages are all from the same space probe. So we are not sure which message is from what system.

Our challenge today is to decode the message and have our solutions determine which planet system the message came from.

##Edit note:

Copying my ASCII data over as input is causing problems. I see that some people who were true heroes and tackled the problem early are seeing this. To fix this we will be altering the challenge. Input will be a set of numbers each represent a byte in the message. Hopefully this will fix the issues.

#Input:

Message broken down into numbers representing the ASCII values of the message between " "

#Output:

The name of the system and the message decoded.

#Encryption and Planet Systems:

Omicron V: will take and invert the 5th bit. (0001 0000) That is the bit location in the byte where we invert the bit.

Hoth: Takes the value of the ASCII character and adds 10 to it.

Ryza IV: Takes the value of the ASCII character and subtracts 1 to it.

Htrae: reverses the characters.

#Validation:

It is not enough to just take the message and decode it in all 4 ways and let you decide which one is right or wrong. You need to have your program/solution determine the right decoding. All messages are in english (I know even in the future on alien planets).

#Example:

##input:

" 101 99 97 101 112 32 110 105 32 101 109 111 99 32 101 87 "

##Note:

This would be "ecaeP ni emoc eW" in displayed ascii - some messages don't display well as the
values take them beyond displayable ascii values (thus the decimal values)

##output:

Htrae: We come in Peace

#Challenge Input:

```
" 71 117  48 115 127 125 117  48 121 126  48  96 117 113 115 117 "
" 97 111  42 109 121 119 111  42 115 120  42 122 111 107 109 111 "
" 86 100  31  98 110 108 100  31 104 109  31 111 100  96  98 100 "
" 101  99  97 101 112  32 110 105  32 101 109 111  99  32 101  87 "
" 84 113 121 124 105  48  64  98 127 119  98 113 125 125 117  98  48 121  99  48  99  96 105 121 126 119  48 127 126  48 101  99 "
" 78 107 115 118 131  42  90 124 121 113 124 107 119 119 111 124  42 115 125  42 125 122 131 115 120 113  42 121 120  42 127 125 "
" 67  96 104 107 120  31  79 113 110 102 113  96 108 108 100 113  31 104 114  31 114 111 120 104 109 102  31 110 109  31 116 114 "
" 115 117  32 110 111  32 103 110 105 121 112 115  32 115 105  32 114 101 109 109  97 114 103 111 114  80  32 121 108 105  97 68 "
" 86 121  98 117  48 100 120 117  48  93 121  99  99 124 117  99 "
" 80 115 124 111  42 126 114 111  42  87 115 125 125 118 111 125 "
" 69 104 113 100  31 115 103 100  31  76 104 114 114 107 100 114 "
" 115 101 108 115 115 105  77  32 101 104 116  32 101 114 105  70 "
```

#Challenge Solution:

The 12 messages are 3 messages in each of the 4 encodings. Hopefully you should come up with

```
"We come in Peace"
"Daily Programmer is spying on us"
"Fire the missiles"
```

in all of the 4 encodings each.

Title: [2015-06-10] Challenge #218 [Intermediate] Generating Polyominoes

Text: # Description

A polyomino is a collection of cells (equal-sized squares) which are connected, that is, each cell shares a border with another one. Think about tetris pieces, those would be tetrominoes - they each have four squares, and there are 5 unique combinations of their squares into unique shapes. Polyominoes are considered equivalent if they can be made to look identical if they are rotated or flipped. For additional background on polyominoes see this link: http://home.adelphi.edu/~stemkoski/mathematrix/polys.html

# Input Description

You will be given a single integer, this is the polyomino order to calculate and draw. Example:

```
4
```

# Formal Output Description

Draw the complete set of unique polyominoes in ASCII art. Example output:

```
##
##

##
 ##

#
#
#
```

```
#

#
#
##

#
##
#
```

# Challenge Input

```
6
```

# Challenge Input Solution

```
######

#
#####

 #
#####

  #
#####

##
 ####

##
####

# #
####

#  #
####

 ##
####

#
#
####

 #
 #
####

#
####
#

#
####
 #
```

```
#
####
  #

#
####
   #

 #
####
  #

 #
####
 #

 #
###
#
#

 #
##
#
##

 #
 #
##
#
#

 #
##
##
#

##
##
##

  #
###
 #
 #

###
 ##
 #

  #
 ##
###
 #

  #
###
```

```
#
#

 ##
##
#
#

###
# #
#

# #
###
#

# #
###
 #

 ##
 #
##
#

#
##
###

 #
###
##

  #
###
##

  #
 ##
##
#
```

Title: [2015-06-17] Challenge #219 [Intermediate] To-do list (Part 2)
Text: #Description

Thanks for that list you made me, my thoughts are way more organised!

I've got a few problems though that I thought you might be able to help with?
Sometimes I put the wrong information in a list item. Maybe to prevent this I'd be able to modify/update the list item? That's not the only problem though, when there are 50+ items it gets kind of hard to work my way through. Do you think you could maybe add the ability to categorise my items? Obviously, if I have that, I'd also like to be able to view by category!

Oh and finally, a few of you were really great and did this last time but is there a way you can somehow make my list retain state so that I don't have to re-type it everytime I turn my computer on again?

The newest To-do list should be capable of the following functionality:

* Modifying an existing list item

* Be able to give a list item a category. The list item should be able to take an arbitrary amount of categorys

* View by category - All list items should be able to be sorted and output by category to make it easier to wade through submissions

* Retain state

Thanks!

#Formal Inputs & Outputs

##Output description

Any output that is created should be user-friendly. When I'm viewing my to-do list, I should be able to easily discern one list item from another.

#Examples

(don't take this too literally, do it how you would like to do it)

##Categorisation

Input:

```
addItem('Go to work','Programming'); //Item belongs to the Programming Category
addItem('Create Sine Waves in C', 'Music', 'Programming); //Belongs to 2 categories, 'Programming' and 'Music');
```

##Category Output
Input:

```
viewList('programming');
viewList('music');
viewList('music', 'programming');
```

Output:

```
----PROGRAMMING----
- A pixel is not a pixel is not a pixel
- The Scheme Programming Language
- Memory in C
- Haskell's School of Music
- Algorithmic Symphonies from one line of code

----MUSIC----
- Modes in Folk Music
- The use of the Melodic Minor Scale
- Haskell's School of Music
- Algorithmic Symphonies from one line of code

----MUSIC & PROGRAMMING----
- Haskell's School of Music
- Algorithmic Symphonies from one line of code
```

## Modifying an item

```
updateItem('Create Sine Waves in C', 'Create Sine Waves in Python');
```

//The item has now changed from 'Create Sine Waves in C' to 'Create Sine Waves in Python'. This should be reflected in the viewList function/method you have created.

Title: [2015-06-24] Challenge #220 [Intermediate] It's Go time!
Text: # [](#IntermediateIcon) _(Intermediate)_: It's Go time!

[Go](https://en.wikipedia.org/wiki/Go_\(game\)) is a board game involving placing black and white stones on a grid. Two opponents take turns to place stones; one player places white stones, the other black. Stones of the same colour form a *group*, as long as they're all connected via the cardinal axes. The leftmost pair of stones (represented by `#`) below are valid groups, and the rightmost pair are not.

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

Now, when a player places stones such that a group of the opponent's colour is touching no more open spaces (*liberties*), then that group is removed from play. The edges of the board do **not** count as open spaces. Let the black stones be represented by `b` and white stones by `w`. Here, the player plays as the black stones.

```
bbbbb
 wwwb
bwbwb
 bbbb
```

Let `B` be the stone I place in the next turn. If I place the stone here:

```
bbbbb
Bwwwb
bwbwb
 bbbb
```

The white group is entirely enclosed by the black group, and so the white group is removed from play.
If a situation were to arise where both your own and your opponent's stones would be removed, your opponent's stones would be removed first, and then (only if your stones still need to be removed) your own stones would be removed.

Liberties don't need to be outside of the group; they can be inside the group, too. These are called *eyes*. Here, the white group survives, as it has the eye:

```
 bbbbb
bbwwwwb
bww wb
 bwwwwb
  bbbbb
```

Your challenge today is to determine the location on the board which, when a stone of your own colour is placed there, will remove the greatest number of your opponent's stones.

# Formal Inputs and Outputs

## Input Description

You will be given the size of the grid as a width and a height. Next, you will be given the player's colour - either `b` or `w`. Finally, you will be given a grid of the appropriate dimensions, using the format I used in the Description: spaces for empty grid regions, and `b` and `w` for stones of either colour.

## Output Description

Output the co-ordinate of the location which, if you were to place a stone of your own colour there, would result in the greatest number of your opponent's stones being removed. The top-left corner is location `(0, 0)`.

# Sample Inputs and Outputs

### Input

```
7 5
b
 bbbbb
bbwwwwb
bww wb
 bwwwwb
  bbbbb
```

### Output

```
(3, 2)
```

### Input

```
9 11
w
    ww
  wwbbbw
  wbbbbw
 wwbbbbw
 wwwwwww
 wbbbbww
 wbwbbww
 wbwbbww
 wwwbbww
    wbw
    w
```

### Output

```
(5, 10)
```

### Input

```
7 7
w
w w w w
 bbbbb
wbbbbbw
 bbbbb
wbbbbbw
 bbbbb
w w w w
```

### Output

```
No constructive move
```

## Sample 4

### Input

```
4 3
b
 bw
bw w
 bw
```

### Output

```
(2, 1)
```

## Sample 5

(thanks to /u/adrian17)

### Input

```
7 5
b
 bb bb
bww wwb
 bbbbb
bwwwb
 bb
```

### Output

```
(3, 1)
```

Title: [2015-07-01] Challenge #221 [Intermediate] Unravelling a word snake
Text: #Description

As we saw on monday, a "word snake" is a snake made from words.

For instance, take this sequence of words:

`SHENANIGANS SALTY YOUNGSTER ROUND DOUBLET TERABYTE ESSENCE`

Notice that the last letter in each word is the same as the first letter in the next word. In order to make this into a word snake, you can simple snake it across the screen

```
SHENANIGANS       DOUBLET
          A       N     E
          L       U     R
          T       O     A
          YOUNGSTER     B
                        Y
                        T
                  ECNESSE
```

Your task on monday was to take an input word sequence and turn it into a word snake. Your task today is to take an input word snake and turn it into a word sequence. The rules for wod snakes are the same as the previous problem, with one addition:

- The snake starts at the top left corner
- Each word will turn 90 degrees left or right to the previous word
- The snake will not intersect itself
- The snake will be unambiguous

The next letter in the snake's path will always be clear, here's an example of an ambiguous snake:

```
CMYE
HLOG
IADN
LPEA
LALR
INSO
```

In this case it's unclear whether snake's inital direction is right or down solving this kind of ambiguous snake would require a dictionary.

Specifically, "unambiguous" means that every letter will only ever have two neighbors, except for the end-points, which will have only one.

#Formal inputs &amp; outputs

##Input

The input will first be a number specifying how many lines the snake will cover. After that follows the word snake (written in ALL CAPS).

Note that the word-snake will not have any trailing spaces on any line, so you can't assume that every line will be equally long. However, you can assume that no input will be wider than 80 characters.

##Output

The resulting sequence of words from unraveling the word snake! Each word will be in all caps and each word will be separated by a space.

#Sample inputs &amp; outputs

##Input 1

```
6
SNAKE
    A   DUSTY
    T   N   U
    SALSA   M
            M
            YACHTS
```

##Output 1

```
SNAKE EATS SALSA AND DUSTY YUMMY YACHTS
```

##Input 2

```
8
W    DINOSAUR
I    E      E
Z  CAR Y    A
A  I   L    C
```

```
R  D    T  OT
D  R    B  V
R  O    U  A
YAWN    SGEL
```

##Ouput 2

```
WIZARDRY YAWN NORDIC CAR RED DINOSAUR REACT TO OVAL LEGS SUBTLY
```

###Challenge inputs

##Input 1

```
8
NUMEROUS
       Y
LUXURY M
O    E B
B O  A O
M DAOR L
Y      I
SDRATSUC
```

##Input 2

```
10
R       TIGER
E       O   E
S       H   T SO
I  GRAPES   U  N
G  A        R  R
NULL  GNIHTON  E
      R        T
      A        N
      N        A
      DELIGHTFUL
```

Title: [2015-07-08] Challenge #222 [Intermediate] Simple Stream Cipher
Text: # Description

Stream ciphers like [RC4](https://en.wikipedia.org/wiki/RC4) operate very simply: they have a strong psuedo-random number generator that takes a key and produces a sequence of psuedo-random bytes as long as the message to be encoded, which is then XORed against the plaintext to provide the cipher text. The strength of the cipher then depends on the strength of the generated stream of bytes - its randomness (or lack thereof) can lead to the text being recoverable.

# Challenge Inputs and Outputs

Your program should have the following components:

* A psuedo-random number generator which takes a key and produces a consistent stream of psuedo-random bytes. A very simple one to implement is the [linear congruential generator (LCG).](https://en.wikipedia.org/wiki/Linear_congruential_generator)
* An "encrypt" function (or method) that takes a key and a plaintext and returns a ciphertext.
* A "decrypt" function (or method) that takes a key and the ciphertext and returns the plaintext.

An example use of this API might look like this (in Python):

```
    key = 31337
    msg = "Attack at dawn"
    ciphertext = enc(msg, key)
    # send to a recipient

    # this is on a recipient's side
    plaintext = dec(ciphertext, key)
```

At this point, `plaintext` should equal the original `msg` value.

Title: [2015-07-15] Challenge #223 [Intermediate] Eel of Fortune
Text: # Description

You work on the popular game show Eel of Fortune, where contestants take turns fishing live eels out of an aquarium for the opportunity to solve a word puzzle. The word puzzle works like the game Hangman. A secret word is obscured on the board. A player guesses a letter of the alphabet, and if that letter appears anywhere in the secret word, all of the times it appears in the secret word are revealed.

An unfortunate incident occurred on yesterday's show. The secret word was `SYNCHRONIZED`, and at one point the board was showing:

```
    S _ N _ _ _ O N _ _ _ D
```

As you can see, the letters on the board spelled "snond", which is of course an extremely offensive word for telemarketer in the Doldunian language. This incident caused ratings to immediately plummet in East Doldunia. The Eel of Fortune producers want the ability to identify "problem words" for any given offensive word.

Write a function that, given a secret word and an offensive word, returns true if the board could theoretically display the offensive word (with no additional letters) during the course of solving the secret word.

# Examples

```
    problem("synchronized", "snond") -> true
    problem("misfunctioned", "snond") -> true
    problem("mispronounced", "snond") -> false
    problem("shotgunned", "snond") -> false
    problem("snond", "snond") -> true
```

# Optional challenges

1. Define the _problem count_ of an offensive word to be the number of words in the [enable1 word list](https://code.google.com/p/dotnetperls-controls/downloads/detail?name=enable1.txt) that return true when paired with that offensive word as secret words. For instance, the problem count of "snond" is 6. What is the problem count of "rrizi" (Zarthan offensive slang for the air in potato chip bags)?
1. (Edited for clarity) What are the 10 largest problem counts of any sequence of 5 letters ("aaaaa", "aaaab", " aaaac", through "zzzzz")? _A solution to this problem needs to finish in less than a year. Aim for a few minutes, or an hour at most. Post your output along with your code._

Title: [2015-07-22] Challenge #224 [Intermediate] Detecting Four Sided Figures
Text: # Description

I got this idea from the [Mensa quiz](https://www.mensa.org/workout/questions), specifically question 17. It's a basic scanning challenge: can your program detect and count intersecting bounding boxes from an ASCII art input? A four-sided figure is an ASCII art rectangle. Note that it can overlap another one, as long as the four corners are fully connected.

# Formal Inputs &amp; Outputs

Your program will be given an ASCII art chart showing boxes and lines. `-` and `|` characters indicate horizontal and vertical lines, respectively, while "+" characters show intersections.

Your program should emit an integer, *N*, of how many unique four sided figures it found. Rectangles and squares both count.

# Example Input

```
                                +----+
                                |    |
+-------------------------+-----+----+
|                         |     |    |
|     +-------------------+-----+    |
|     |                   |     |    |
|     |                   |     |    |
+-----+-------------------+-----+    |
      |                   |     |    |
      |                   |     |    |
      +-------------------+-----+    |
                          |     |    |
                          |     |    |
                          |     |    |
                          +-----+----+
                                |    |
                                |    |
                                |    |
                                +----+
```

# Example Output

For the above diagram your program should find 25 four sided figures.

# Challenge Input

This one adds a bit to the complexity by throwing in some three sided figures. This should catch more naive implementations.

```
              +-----------+
              |           |
              |           |
              |           |
              |           |
+-------------+-----------+-------------+
|             |           |             |
|             |           |             |
|             |           |             |
|             |           |             |
+-------------+-----------+-------------+
              |           |
              |           |
              |           |
                  |           |
    +-------------+-----------+-------------+
    |             |           |             |
    |             |           |             |
    |             |           |             |
    |             |           |             |
    +-------------+-----------+-------------+
                  |           |
                  |           |
                  |           |
                  |           |
                  +-----------+
```

# Challenge Output

For the challenge diagram your program should find 25 four sided figures.

Title: [2015-07-29] Challenge #225 [Intermediate] Estimating pi from images of circles
Text: #Description

In modern times, if we wish to calculate the value of pi to some precision, there are plenty of mathematical formulas that you can use to get the value. [Leibniz formula for pi](https://en.wikipedia.org/wiki/Leibniz_formula_for_%CF%80) and the solution to the [Basel problem](https://en.wikipedia.org/wiki/Basel_problem) are two of the most famous ones, though both converge very slowly. Modern day computers that attempt to set the world record for digits of pi generally use some variation on [Ramanujan's formula](https://en.wikipedia.org/wiki/Approximations_of_%CF%80#20th_century), which converges very rapidly.

However, back in the good old days, we didn't know of these formulas. They all depend on analysis and infinite sums which mathematicians had no skill at manipulating. Back then, the way you estimated pi was less accurate but more straight-forward: you drew a circle, measured it, and calculated pi from that.

Today, we're going to honor those mathematicians of old. You will be given an image of a black circle on white background, and using the pixel data in the image, you are to come up with an estimate for pi.

For those of you who have forgotten your formulas for circles, the formula for the area of a circle is as follows:

```
A = pi * r^2
```

In other words, to calculate the area of a circle, multiply pi by the square of the radius.

#Formal inputs &amp; outputs
##Inputs

As input, you will be given an image that contains a black circle on white background (those are the only two colors in the image, there's no anti-aliasing or anything). The image provided will be in PNG format, but if you find it difficult to import and analyze PNG images in your language, you're welcome to use a tool like ImageMagick to convert it to a format you prefer (the [Netpbm](https://en.wikipedia.org/wiki/Netpbm) family of formats are famously easy for a computers to parse).

Note that for challenge input 1, the height and width of the image itself is equal to the diameter of the circle, but that *is not* true for challenge input #2. It is highly encouraged (but not required) that you to try and write a program that works for both challenge inputs.

##Outputs

You will output a single line containing your estimate of pi based on the image. It doesn't have to be very exact in all decimal places, just the closest value you can get by looking at the image provided.

#Challenge inputs

##Input 1

[This image](http://i.imgur.com/5GScbUe.png)

##Input 2

[This image](http://i.imgur.com/dRko2KH.png)

#Bonus

If you really want to test your test your skills, extract an estimate of pi from [this image](http://i.imgur.com/Cp0hxTh.png)

#Notes

As always, if you have a challenge suggestion, head on over to /r/dailyprogrammer_ideas and suggest it!

Also, for you historians out there who are going to comment "that's not how Archimedes did it!": [yes, I know that other methods were used](https://en.wikipedia.org/wiki/Pi#Polygon_approximation_era), but lets just forget that for the purposes of this problem :)

Title: [2015-07-31] Challenge #225 [Intermediate] Diagonal Maze
Text: # [](#IntermediateIcon) _(Intermediate)_: Diagonal Maze

A maze can be represented using characters as follows:

```
+-+-+-+-+-+
  |       |
+ +-+-+ + +
| |     | |
+ + + + + +
|   | |   |
+-+-+ +-+-+
|     |   |
+ + +-+ + +
| |       |
+-+-+-+-+-+
```

However, the exact same maze can also be represented diagonally using slashes, like this:

```
    \
   / /\
  / /\ \
 /\   \ \
/  \/    \
\/   / / /
 \ \/\  /
  \   \/
   \/ /
    \
```

Your task today is to convert from the first format (cardinal) to the second (diagonal).

# Formal Inputs and Outputs

## Input Specification

You'll be given a number **N** on one line, followed by **N** further lines of input of a cardinal axis aligned maze, like so:

```
11
+-+-+-+-+-+
  |       |
+ +-+-+ + +
| |     | |
+ + + + + +
|   | |   |
+-+-+ +-+-+
|     |   |
+ + +-+ + +
| |     |
+-+-+-+-+-+
```

The maze cells will *not* necessarily be one-by-one, so watch out!

## Output Description

Output the diagonal-ified maze, like the one shown above (same as in description).

# Sample Inputs and Outputs

## Example 1

```
16
    +--+--+--+--+--+
          |     |  |
          |     |  |
    +  +--+  +  +  +
    |     |  |  |  |
    |     |  |  |  |
    +--+  +  +  +  +
    |     |  |     |
    |     |  |     |
    +  +--+  +  +--+
    |        |     |
    |        |     |
    +--+--+--+--+  +
    |
    |
    +--+--+--+--+--+
```

### Output

```
          \
           \
       /    \
      /      \
     /\   \  /\
    /  \   \/  \
   /       /    \
  /       /      \
 /\   \  /   /   /\
/  \   \/   /   /  \
\   \      /   /   /
 \   \    /   /   /
  \   \  /       /
   \   \/       /
    \   \   \  /
     \   \   \/
      \      /
       \    /
        \
         \
```

## Example 2

### Input

```
17
+---+---+---+---+---+---+---+
                |
                |
                |
+---+---+---+---+---+---+   +
                |
                |
                |
+---+---+---+---+---+---+---+
|
|
|
+   +---+---+---+---+---+---+
|
|
|
+---+---+---+---+---+---+---+
```

### Output

```
                \
                 \
                  \
             \     \
              \     \
               \     \
         /\     \     \
        /  \     \     \
       /    \     \     \
      /      \     \     \
     /        \     \     \
    /          \     \     \
    \     \     \     \     \
     \     \     \     \     \
      \     \     \     \     \
       \     \     \     \     \
        \     \     \     \     \
         \     \     \     \     \
          \     \     \          /
           \     \     \        /
            \     \     \      /
             \     \     \    /
              \     \     \  /
               \     \     \/
                \     \
                 \     \
                  \     \
                   \
                    \
                     \
```

Title: [2015-08-05] Challenge #226 [Intermediate] Connect Four
Text: ** EDITED ** Corrected the challenge output (my bad), verified with solutions from /u/Hells_Bell10 and /u/mdskrzypczyk

# Description

Connect Four is a two-player connection game in which the players first choose a color and then take turns dropping colored discs (like checkers) from the top into a seven-column, six-row vertically suspended grid. The pieces fall straight down, occupying the next available space within the column. The objective of the game is to connect four of one's own discs of the same color next to each other vertically, horizontally, or diagonally before your opponent.

A fun discourse on winning strategies at Connect Four is found here http://www.pomakis.com/c4/expert_play.html .

In this challenge you'll be given a set of game moves and then be asked to figure out who won and when (there are more moves than needed). You should safely assume that all moves should be valid (e.g. no more than 6 per column).

For sake of consistency, this is how we'll organize the board, rows as numbers *1-6* descending and columns as letters *a-g*. This was chosen to make the first moves in row 1.

```
    a b c d e f g
6   . . . . . . .
5   . . . . . . .
4   . . . . . . .
3   . . . . . . .
```

```
2  . . . . . . .
1  . . . . . . .
```

# Input Description

You'll be given a game with a list of moves. Moves will be given by *column only* (gotta make this challenging somehow). We'll call the players *X* and *O*, with *X* going first using columns designated with an uppercase letter and *O* going second and moves designated with the lowercase letter of the column they chose.

```
C  d
D  d
D  b
C  f
C  c
B  a
A  d
G  e
E  g
```

# Output Description

Your program should output the player ID who won, what move they won, and what final position (column and row) won. Optionally list the four pieces they used to win.

```
X won at move 7 (with A2 B2 C2 D2)
```

# Challenge Input

```
D  d
D  c
C  c
C  c
G  f
F  d
F  f
D  f
A  a
E  b
E  e
B  g
G  g
B  a
```

# Challenge Output

```
O won at move 11 (with c1 d2 e3 f4)
```

Title: [2015-08-12] Challenge #227 [Intermediate] Contiguous chains
Text: #Description:

If something is contiguous, it means it is connected or unbroken. For a chain, this would mean that all parts of the chain are reachable without leaving the chain. So, in this little piece of ASCII-art:

```
xxxxxxxx
x      x
```

there is only 1 contiguous chain, while in this

```
xxxx xxxx

x
```
there are 3 contiguous chains. Note that a single x, unconnected to any other, counts as one chain.

For the purposes of this problems, chains can only be contiguous if they connect horizontally of vertically, not diagonally. So this image

```
xx
 xx
  xx
```

contains three chains.

Your challenge today is to write a program that calculates the number of contiguous chains in a given input.

#Formal inputs &amp; outputs
##Input:

The first line in the input will consist of two numbers separated by a space, giving the dimensions of the ASCII-field you're supposed to read. The first number gives the number of lines to read, the second the number of columns (all lines have the same number of columns).

After that follows the field itself, consisting of only x's and spaces.

##Output:

Output a single number giving the number of contiguous chains.

#Sample inputs &amp; outputs

##Input 1

```
2 8
xxxxxxxx
x      x
```

##Output 1

```
1
```

##Input 2

```
3 9
xxxx xxxx
    x
   xx
```

##Output 2

```
    3
```

#Challenge inputs:

##Input 1

```
    4 9
    xxxx xxxx
       xxx
    x   x   x
    xxxxxxxxx
```

##Input 2

```
    8 11
    xx x xx x
    x  x xx x
    xx   xx  x
    xxxxxxxxx x
             xx
    xxxxxxxxxxx
     x x x x x
      x x x x
```

#Bonus
/u/Cephian was nice enough to generete [a much larger 1000x1000 input](https://gist.githubusercontent.com/Cephian/f1c2869bd67d40c88042/raw/63871d115bc7bf887eac5434b5c6c5494f83ba2e/10.txt) which you are welcome to use if you want a little tougher performance test.