Title: [2015-08-07] Challenge #226 [Hard] Kakuro Solver
Text: # Description

Kakuro is a popular Japanese logic puzzle sometimes called a mathematical crossword. The objective of the puzzle is to insert a digit from 1 to 9 inclusive into each white cell such that the sum of the numbers in each entry matches the clue associated with it and that no digit is duplicated in any contiguous row or column. It is that lack of duplication that makes creating Kakuro puzzles with unique solutions possible. Numbers in cells elsewhere in the grid may be reused.

More background on Kakuro can be found on [Wikipedia](https://en.wikipedia.org/wiki/Kakuro). There's an [online version](http://www.kakuroconquest.com/) you can play as well.

# Input Description

You'll be given a pair of integers showing you the number of columns and rows (respectively) for the game puzzle. Then you'll be given *col* + *row* lines with the sum and the cell identifiers as *col id* and *row number*. Example:

```
1 2
3 A1 A2
```

This example means that the sum of two values in A1 and A2 should equal 3.

# Challenge Output

Your program should emit the puzzle as a 2D grid of numbers, with columns as letters (e.g. A, B, C) and rows as numbers (1, 2, 3). Example:

```
  A
1 1
2 2
```

# Challenge Input

This puzzle is a 2x3 matrix. Note that it has non-unique solutions.

```
2 3
13 A1 A2 A3
8 B1 B2 B3
6 A1 B1
6 A2 B2
9 A3 B3
```

# Challenge Output

One possible solution for the above puzzle is

```
  A  B
1 5  1
2 2  4
3 6  3
```

Title: [2015-07-24] Challenge #224 [Hard] Langford strings
Text: #Description

A "Langford string of order N" is defined as follows:

* The length of the string is equal to 2*N
* The string contains the the first N letters of the uppercase English alphabet, with each letter appearing twice

* Each pair of letters contain X letters between them, with X being that letter's position in the alphabet (that is, there is one letter between the two A's, two letters between the two B's, three letters between the two C's, etc)

An example will make this clearer. These are the only two possible Langford strings of order 3:

```
    BCABAC
    CABACB
```

Notice that for both strings, the A's have 1 letter between them, the B's have two letters between them, and the C's have three letters between them. As another example, this is a Langford string of order 7:

```
    DFAGADCEFBCGBE
```

It can be shown that Langford strings only exist when the order is a multiple of 4, or one less than a multiple of 4.

Your challenge today is to calculate all Langford strings of a given order.

#Formal inputs &amp; outputs

##Inputs

You will be given a single number, which is the order of the Langford strings you're going to calculate.

##Outputs

The output will be all the Langford strings of the given order, one per line. The ordering of the strings does not matter.

Note that for the second challenge input, the output will be somewhat lengthy. If you wish to show your output off, I suggest using a service like gist.github.com or [hastebin](http://hastebin.com) and provide a link instead of pasting them directly in your comments.

#Sample input &amp; output

##Input

```
    3
```

##Output

```
    BCABAC
    CABACB
```

#Challenge inputs

##Input 1

```
    4
```

##Input 2

```
    8
```

#Bonus

For a bit of a stiffer challenge, consider this: there are more than 5 trillion different Langford strings of order 20. If you put all those strings into a big list and sorted it, what would the first 10 strings be?

Title: [2015-07-17] Challenge #223 [Hard] The Heighway dragon fractal
Text: # Description

Write a program to print out the (x, y) coordinates of each point in the nth iteration of the [Heighway dragon fractal](http://www-user.uni-bremen.de/schmuhl/fractals/dragon_curve_o12.png). Start at the origin (0, 0) and take steps of length 1, starting in the positive x direction (1, 0), then turning to the positive y direction (1, 1). Your program should generate 2^n + 1 lines of output.

You can use any resources you want for help coming up with the algorithm, but if you want to start from the very beginning, use only the fact that the nth iteration can be made by [folding a strip of paper in half n times, then unfolding it so that each crease is at a right angle](http://www.cutoutfoldup.com/images/0216-s03b.jpg).

# Example

For n = 3, your output should be:

```
0 0
1 0
1 1
0 1
0 2
-1 2
-1 1
-2 1
-2 2
```

[Plotted image of these points](http://i.imgur.com/3sCzNyG.png), made using LibreOffice.

The sum of the x's here is -4, and the sum of the y's is 10. For n = 12, the sums are -104896 and 52416. To verify that your program is correct, post the sum of x's and y's for n = 16 along with your code.

# Optional challenges

Today's basic challenge is not too hard, relatively speaking, so if you want more, try some of these optional add-ons, or take it in your own direction.

1. Show us a plot of your output. There are many options for this. You can use a plotting library for your language of choice, or use a spreadsheet like I did. gnuplot is another free option. Feel free to get creative with colors, effects, animations, etc.
1. Optimize your code for memory usage. Aim for O(n) space.
1. Optimize your code for speed. What's the largest n you can generate all the data for in less than 1 minute? (You can skip printing output for this one, as long as you actually do all the calculations.)
1. Golf: minimize your code length. What's the shortest program you can write in your language that works?
1. There are [other ways of generating the Heighway dragon](http://i.imgur.com/n30yp.gif) than the paper folding one I suggested. Try implementing a different one than you used first.
1. There are many variations of the Heighway dragon [(see Variations at the bottom)](http://ecademy.agnesscott.edu/~lriddle/ifs/heighway/heighway.htm). Try creating a terdragon, golden dragon, or anything else you can find.
1. Find a way to efficiently calculate s(n), the sum of the x's and y's for the nth iteration. For example, s(3) = (-4, 10) and s(12) = (-104896, 52416). Post s(100) along with your code. (This is possible without any advanced math, but it's tricky.)
1. Find a way to efficiently calculate p(k), the (x, y) position after k steps (i.e. the (k+1)th line of output when n is sufficiently large), starting from from p(0) = (0, 0), p(1) = (1, 0). For example, p(345) = (13, 6). Post p(3^(45)) along with your code. (This one is also quite tricky.)

Title: [2015-07-10] Challenge #222 [Hard] Customer Unit Delivery Scheduling
Text: # Description

You run a business where you sell doohickies, and business is booming. You're customers are all local, but you're just getting off the ground and you don't have a large fleet of trucks, just one driver. Your truck has a finite capacity, and you have to keep costs down as you make deliveries - minimize milage, maximize deliveries, etc. That's where today's challenge program comes in.

As you make delivery runs, your truck will run out of enough doohickies and so you have to return to the depot and restock it. Assume that you refill the truck to its full capacity on visiting the depot. You may visit them in any order but must visit them all and satisfy all orders. Finally, assume the truck has an infinite energy source, so don't worry about refueling.

# Input Description

You'll be given a line with an integer *N*, which tells you how many doohickies your truck can hold, and a two-tuple of coordinates (x & y) where the doohickie depot is. Then you'll be given a line with another single integer *M*, which tells you how many customers to read. Each customer line (of which there are *M*) will be how many units they want and then a two-tuple telling you the x,y coordinated where the customer is located.

# Output Description

Your program should emit the sequence of stops you need to make, including depot stops, that *minimizes* the distance driven. You must deliver enough units for every customer when you stop! No customer will ask for more than *N* doohickies (your truck's capacity), and you *should* expect to travel from one customer to the next without stopping at the depot if you can deliver enough units at once.

# Challenge Input

```
        40 (20,20)
        12
        10 (20,8)
        15 (31,20)
        18 (13,21)
        17 (30,20)
        3 (20,10)
        5 (11,29)
        9 (28,12)
        4 (14,14)
        6 (32,8)
  12 (1,1)
  18 (3,32)
  23 (5,5)
```

Title: [2015-07-03] Challenge #221 [Hard] Poetry in a haystack
Text: #Description

Today we're going to try something a little bit different.

You're are going to be given a file with 50,000 lines of text in it. 49,997 of those lines are going to be gibberish, but 3 lines are going to be part of a famous poem. Your task today is to find those three lines.

A few notes:

* All text in the file is lower-case
* All lines contain nothing but alphabetic characters, spaces, and a few pieces of punctuation
* The lines of poetry are written in English
* The three lines of the poem is in the file in the right order, but split up with lines of gibberish.

#Formal inputs &amp; outputs

##Input

The input for this challenge is
[this](https://gist.githubusercontent.com/anonymous/c8fb349e9ae4fcb40cb5/raw/05a1ef03626057e1b57b5bbdddc4c2373ce4b46
5/challenge.txt) aforementioned file. Download it and use it as input for your problems.

##Output

The three lines of the poem, in the right order.

Note that it might be the case that you reduce the number of possible lines to some very low number (say, 10-20 lines), after which you can easily use visual inspection to find the right lines. This is an acceptable way to solve the problem, but I *highly encourage* you to try and find a way to print only the correct lines.

Oh, and by the way: if you happen to figure out what the right lines are exactly, either from visual inspection, reading it in a comment here (if you do solve the problem and wish to post the output, please indent the output with four space so as to hide the text as a spoiler), or any other way, you are not allowed to just put in a search function in your code for the correct words. That's cheating :). You have to figure out a way to do it "legitimately", and write the code pretending you have no idea what the lines are supposed to be.

#Notes

If you have a suggestion for a problem, please head to /r/dailyprogrammer_ideas and suggest it!

Much thanks today to /u/adrian17 for some comments on the design of the problem on IRC. By the way, did you know we have an IRC channel where you can go to chat with other dailyprogrammers and get help on problems you are struggling with? It's on irc.freenode.net in the channel #reddit-dailyprogrammer. Why don't you stop by if you have a chance?

On another note: I was unsure how to classify this problem, whether it is hard enough for the [Hard] difficulty. I would much appreciate feedback on whether you guys think this is an appropriate challenge for [Hard] and whether it was a good challenge in general. Be honest but gentle :)

Title: [2015-06-26] Challenge #220 [Hard] Substitution Cryptanalysis
Text: # [](#HardIcon) _(Hard)_: Substitution Cryptanalysis

A [substitution cipher](https://en.wikipedia.org/?title=Substitution_cipher) is one where each letter in the alphabet is substituted for another letter. It's like a Caesar shift cipher, but where every letter is ciphered independently. For example, look at the two rows below.

```
    abcdefghijklmnopqrstuvwxyz
    YOJHZKNEALPBRMCQDVGUSITFXW
```

To encode something, find the letter on the top row, and swap it with the letter on the bottom row - and vice versa. For example, the plaintext:

```
    hello world
```

Becomes:

```
    EZBBC TCVBH
```

Now, how would you go about decrypting something like this? Let's take another example, with a different key.

```
    IAL FTNHPL PDDI DR RDNP WF IUD
```

You're also given the following hints: `A` is ciphered to `H` and `O` is ciphered to `D`. You know the text was in English, so you could plausibly use a word list to rule out impossible decrypted texts - for example, in the third words `PDDI`, there is a double-O in the middle, so the first letter rules out P being the letter Q, as Q is always followed by a U.

Your challenge is to decrypt a cipher-text into a list of possible original texts using a few letters of the substitution key, and whichever means you have at your disposal.

# Formal Inputs and Outputs

## Input Description

On the first line of input you will be given the ciphertext. Then, you're given a number **N**. Finally, on the next **N** lines, you're given pairs of letters, which are pieces of the key. For example, to represent our situation above:

```
IAL FTNHPL PDDI DR RDNP WF IUD
2
aH
oD
```

Nothing is case-sensitive. You may assume all plain-texts are in English. Punctuation is preserved, including spaces.

## Output Description

Output a list of possible plain-texts. Sometimes this may only be one, if your input is specific enough. In this case:

```
the square root of four is two
```

You don't need to output the entire substitution key. In fact, it may not even be possible to do so, if the original text isn't a pangram.

# Sample Inputs and Outputs

## Sample 1

### Input

```
LBH'ER ABG PBBXVAT CBEX PUBC FNAQJVPURF
2
rE
wJ
```

### Output

```
you're not cooking pork chop sandwiches
you're nob cooking pork chop sandwiches
```

Obviously we can guess which output is valid.

## Sample 2

### Input

This case will check your word list validator.

```
ABCDEF
2
aC
zF
```

### Output

```
quartz
```

## Sample 3

### Input

```
WRKZ DG ZRDG D AOX'Z VQVX
2
wW
sG
```

### Output

```
what is this i don't even
whet is this i can't ulun
```

(what's a ulun? I need a better word list!)

## Sample 4

### Input

```
JNOH MALAJJGJ SLNOGQ JSOGX
1
sX
```

### Output

```
long parallel ironed lines
```

# Notes

There's a handy word-list [here](https://gist.githubusercontent.com/Quackmatic/512736d51d84277594f2/raw/words) or you could check out [this thread](/r/dailyprogrammer/comments/2nluof/) talking about word lists.

You could also *in*validate words, rather than just validating them - check out [this list of impossible two-letter combinations](http://linguistics.stackexchange.com/questions/4082/impossible-bigrams-in-the-english-language). If you're using multiple systems, perhaps you could use a weighted scoring system to find the correct decrypted text.

There's an [example solver](http://quipqiup.com/) for this type of challenge, which will try to solve it, but it has a really weird word-list and ignores punctuation so it may not be awfully useful.

Title: [2015-06-17] Challenge #219 [Hard] The Cave of Prosperity
Text: #Description

Mandy is out in the woods one day hiking with her backpack Steven (yes, she named her backpack Steven) when she suddently sees a cave in the side of a hill. Curious, she walks into it and after a while she comes to a big room. In this room is a small chest filled with gold nuggets. Suddenly, she hears a voice: "Welcome, Mandy, to The Cave of Prosperity!", it says. "You may take as as much gold as you can carry from this room, but once you leave it, you may never return!"

Mandy knows that Steven can carry up to 10 kilograms of gold (Steven's not a very good backpack, as it turns out), and luckily, there's a scale next to the chest. She sees that there are five gold nuggets in the chest and she weighs them. Four of them weigh 2 kilograms each and the last one weighs 5 kilograms. She puts the four gold nuggets weighing 2 kilograms (for a total of 8 kilograms) into to Steven and exists the cave, happy at her good fortune.

However, as you might have realized, Mandy made a mistake! If she had taken the 5 kilogram nugget and two of the 2 kilogram nuggets, she would have gotten out with 9 kilograms of gold instead of 8.

Today, you are going to visit The Cave of Prosperity, and we are going to see if you can do better than Mandy

#Formal inputs &amp; outputs
##Input
On the first line of the input, you will get the capacity of your backpack, rounded to 7 digits after the decimal point. After that, there will be one line specifying how many gold nuggets there are in the cave. After that, there will be one line per gold nugget specifying how much each of them weighs.

The weights of the gold nuggets will be a floating point number between 0.0 and 1.0, with seven digits after the decimal point.

##Output
On the first line of the output, you will specify how much gold you are able to escape with by putting gold nuggets into your backpack. This number will be as large as possible without exceeding the capacity of your backpack.

After that, you will print out the weights of the gold nuggets you have collected. In other words, the first line should be the sum of the rest of the lines.

#Sample inputs &amp; outputs
##Input 1

```
2.0000000
5
0.3958356
0.4109163
0.5924923
0.6688261
0.8720640
```

##Output 1

```
1.9518064
0.4109163
0.6688261
0.8720640
```

##Input 2

```
4.0000000
10
0.0359785
0.9185395
0.2461690
0.7862738
0.9237070
0.2655587
0.3373235
0.8795087
0.7802254
0.8158674
```

##Output 2

```
3.9970232
0.9185395
0.2655587
```

```
    0.3373235
    0.8795087
    0.7802254
    0.8158674
```

#Challenge inputs
##Input 1
[This 15-nugget challenge](https://gist.githubusercontent.com/anonymous/d18c4b31a9e4aa2941c4/raw/c51cd7fdaf925a6137f8728a5b30741615ba923d/gistfile1.txt)

##Input 2
[This 30-nugget challenge](https://gist.githubusercontent.com/anonymous/2451fef8cbbd0fa30705/raw/7b877d5b5330106aa1af935ea52ec5616541c8db/gistfile1.txt)

#Bonus

[This 46-nugget challenge](https://gist.githubusercontent.com/anonymous/39899cb2d250a7fd02fa/raw/6f8be09206dfbcbe32915678260dd42c5a75a435/gistfile1.txt)

Title: [2015-06-12] Challenge #218 [Hard] Elevator Scheduling
Text: While this one is a hard challenge, it's also open ended - be creative in how you solve the problem. I encourage you to compare notes, think about the challenge before you write code, and explore your algorithms. See the bonus questions below for some additional questions.

# Description

Most of us have seen and ridden elevators - you crazy folks in the UK and commonwealth countries often call them "lifts" - but I'm sure I'm not the only one who has puzzled about the scheduling algorithms. Which riders do you pick up and when? Do you service requests in the order of arrival or do you work on maximal overlap?

For this challenge, you'll have to anwer those questions. You're designing an elevator scheduling algorithm for a building and you have plenty of riders to keep happy. You can have any algorithm you want as long as you stick to the constraints - the cars have a fixed capacity and speed.

Make sure you see the bonus questions after the challenge input.

# Input Description

You'll be given a single integer *N* on a line. The first *N* lines will identify elevator cars and with these fields: Car identifier, capacity, vertical speed in floors per second, and starting floor. Assume instantaneous getting on or off the elevator for the riders once you arrive on the floor. Assume that the elevator *is able to* leave with the rider as soon as it is able, but it *may* linger waiting for more people to arrive - the choice is yours.

Example:

```
        C1 12 .1 1
```

This translates to Car 1, capacity of 12 people, moves at .1 floors per second (ten seconds to traverse a floor up or down), and starting at floor 1.

Then you'll get another integer on a line, *M*. The next *M* lines will show riders, with fields: Rider identification, elevator request time in seconds, source floor and destination floor. Rider identification numbers will be *stable*, meaning the rider will have the same identifier the entire exercise. Examples:

```
R1 0 1 4
```

This translates to Rider 1 who at time point 0 wants to go from floor 1 to floor 4. Riders will not transit floors without an elevator.

# Output Description

The main thing to show in the output is the time point at which all requests have been satisfied. (Yes, this is trying to get you guys to compete for the most efficient algorithm). Optionally show all intermediate steps and journeys, and wait times for riders.

# Challenge Input

This was randomly generated, and so it has a few "oddities" in it, like riders who get on and off on the same floor, and riders who change their destination in the next second (e.g. in the middle of a ride). You still have to satisfy *every* request.

```
    2
    C1 12 .1 1
    C2 12 .2 1
    359
R3 0 1 9
R4 1 1 11
R0 11 1 7
R2 11 1 9
R15 13 1 9
R5 26 1 4
R16 27 1 2
R1 28 1 2
R13 28 1 9
R10 32 1 3
R14 35 1 4
R8 36 1 10
R17 38 1 12
R3 49 9 9
R18 50 1 10
R7 51 1 3
R10 53 3 10
R12 54 1 6
R0 60 7 1
R1 62 2 1
R9 66 1 8
R19 66 1 6
R15 71 9 2
R11 72 1 8
R16 78 2 4
R6 82 1 12
R8 85 10 11
R10 89 10 12
R3 90 9 6
R5 94 4 7
R2 94 9 10
R6 95 12 1
R3 111 6 9
R14 114 4 5
R13 115 9 5
R19 117 6 2
R12 122 6 12
R4 123 11 7
R9 123 8 12
```

R6 124 1 5
R0 124 1 6
R7 127 3 3
R11 139 8 9
R7 141 3 4
R17 143 12 2
R14 143 5 5
R16 151 4 9
R5 155 7 12
R1 155 1 11
R18 159 10 10
R15 160 2 4
R19 162 2 3
R2 164 10 3
R11 164 9 9
R3 165 9 4
R12 167 12 1
R10 169 12 1
R0 174 6 9
R11 181 9 2
R18 182 10 12
R9 184 12 4
R5 185 12 11
R4 197 7 5
R2 198 3 3
R3 198 4 8
R6 199 5 5
R8 199 11 6
R13 201 5 5
R14 203 5 4
R1 205 11 12
R16 211 9 1
R6 212 5 11
R7 214 4 8
R15 216 4 6
R19 226 3 11
R1 230 12 12
R7 232 8 5
R0 234 9 12
R3 237 8 2
R17 238 2 6
R2 240 3 11
R12 240 1 3
R15 246 6 6
R13 247 5 10
R5 248 11 5
R10 249 1 6
R18 252 12 4
R9 253 4 8
R1 256 12 12
R4 257 5 12
R16 258 1 2
R13 258 10 5
R6 262 11 2
R11 263 2 7
R9 269 8 5
R3 271 2 6
R14 274 4 9

```
R5 282 5 12
R11 285 7 6
R16 287 2 8
R14 290 9 5
R2 297 11 4
R18 299 4 6
R13 300 5 5
R8 301 6 5
R0 303 12 3
R19 305 11 1
R7 310 5 8
R2 311 4 4
R1 315 12 8
R16 318 8 11
R8 320 5 8
R1 324 8 2
R10 325 6 9
R17 325 6 2
R2 330 4 11
R19 330 1 9
R9 332 5 5
R5 335 12 11
R18 338 6 9
R11 340 6 8
R12 342 3 9
R9 344 5 11
R12 346 9 12
R13 346 5 12
R6 351 2 2
R0 354 3 10
R10 358 9 9
R4 369 12 12
R15 370 6 8
R3 372 6 5
R17 374 2 9
R14 383 5 4
R7 389 8 1
R18 396 9 6
R12 396 12 7
R8 411 8 1
R16 419 11 3
R2 420 11 1
R10 420 9 11
R6 423 2 12
R1 423 2 8
R7 425 1 1
R11 426 8 4
R13 429 12 11
R19 430 9 7
R5 432 11 9
R15 435 8 3
R0 438 10 6
R6 444 12 9
R17 449 9 9
R14 452 4 4
R9 456 11 2
R18 460 6 7
R5 463 9 2
```

```
R12 464 7 2
R4 468 12 5
R13 468 11 6
R2 475 1 4
R19 478 7 4
R12 491 2 10
R10 496 11 7
R0 501 6 3
R2 501 4 2
R7 502 1 3
R3 502 5 7
R14 505 4 11
R6 507 9 2
R1 508 8 12
R15 510 3 1
R16 512 3 12
R11 515 4 10
R18 515 7 2
R19 517 4 3
R15 519 1 5
R9 521 2 2
R2 524 2 5
R14 525 11 2
R18 526 2 11
R4 530 5 5
R6 531 2 5
R8 536 1 5
R12 536 10 3
R16 536 12 7
R15 538 5 7
R17 538 9 5
R13 544 6 7
R10 546 7 11
R11 547 10 5
R7 548 3 1
R4 554 5 1
R3 558 7 11
R10 568 11 7
R6 570 5 5
R12 572 3 7
R7 573 1 4
R19 574 3 6
R16 576 7 3
R0 577 3 8
R4 586 1 9
R11 587 5 9
R14 587 2 4
R2 590 5 5
R5 599 2 2
R10 599 7 7
R9 601 2 4
R1 603 12 6
R3 606 11 1
R18 606 11 9
R13 610 7 11
R10 614 7 4
R17 615 5 4
R16 616 3 3
```

```
R12 617 7 10
R7 621 4 2
R6 622 5 4
R19 626 6 12
R2 628 5 11
R15 629 7 7
R14 630 4 4
R11 632 9 6
R8 632 5 3
R0 639 8 6
R6 649 4 10
R10 651 4 11
R9 653 4 6
R14 653 4 12
R4 655 9 10
R0 656 6 4
R2 660 11 5
R13 660 11 6
R3 663 1 6
R18 664 9 5
R1 667 6 7
R5 668 2 11
R12 668 10 9
R16 672 3 9
R15 675 7 4
R17 680 4 3
R7 681 2 10
R9 681 6 9
R10 686 11 10
R14 689 12 9
R4 690 10 3
R1 698 7 9
R18 698 5 8
R0 699 4 12
R19 705 12 7
R2 708 5 1
R8 712 3 8
R13 718 6 2
R0 721 12 7
R14 721 9 5
R18 722 8 7
R15 723 4 8
R14 730 5 11
R4 733 3 12
R13 738 2 4
R6 741 10 1
R10 741 10 1
R15 741 8 9
R19 743 7 2
R13 751 4 7
R3 752 6 1
R14 755 11 9
R4 758 12 2
R11 759 6 9
R5 762 11 9
R15 765 9 2
R19 770 2 6
R9 775 9 9
```

```
R12 777 9 12
R17 778 3 7
R0 780 7 3
R0 781 3 11
R18 785 7 1
R8 787 8 11
R6 788 1 11
R7 790 10 4
R19 791 6 7
R13 791 7 6
R2 792 1 1
R9 794 9 5
R10 800 1 10
R15 804 2 5
R12 807 12 1
R11 808 9 4
R5 809 9 5
R14 813 9 2
R1 819 9 11
R19 819 7 5
R16 822 9 4
R0 823 11 8
R17 828 7 2
R11 834 4 4
R8 834 11 11
R3 837 1 6
R5 839 5 4
R4 842 2 4
R2 844 1 11
R18 851 1 1
R15 854 5 8
R0 855 8 5
R6 857 11 11
R12 857 1 3
R9 858 5 11
R8 859 11 3
R10 863 10 5
R7 867 4 6
R5 869 4 6
R0 878 5 8
R6 879 11 12
R7 882 6 12
R17 883 2 10
R13 883 6 5
R8 885 3 11
R13 887 5 7
R15 888 8 6
R3 891 6 6
R6 898 12 10
R17 898 10 3
R3 899 6 5
R5 900 6 11
R18 901 1 9
R15 906 6 10
R19 907 5 12
R13 908 7 9
R11 914 4 5
R16 917 4 5
```

```
    R8 924 11 11
    R14 924 2 2
    R0 926 8 9
    R9 926 11 2
    R2 935 11 7
    R1 937 11 5
    R10 940 5 8
    R18 946 9 11
    R19 946 12 4
    R3 947 5 8
    R8 947 11 4
    R13 947 9 4
    R12 948 3 4
    R4 950 4 2
    R9 951 2 9
    R0 963 9 11
    R17 973 3 3
    R16 975 5 12
    R18 977 11 12
    R9 980 9 6
    R13 980 4 9
    R5 983 11 1
    R3 983 8 11
    R7 985 12 7
    R14 985 2 8
    R10 991 8 12
    R19 991 4 6
    R17 992 3 5
    R0 993 11 6
    R1 997 5 3
```
# Bonus

Which improves delivery efficiency most?

* Longer linger times?
* More cars?
* Faster cars?

Title: [2015-06-05] Challenge #217 [Practical Exercise] TeXSCII
Text: # [](#PEIcon) _(Practical Exercise)_: TeXSCII

LaTeX is a typesetting utility based on the TeX typesetting and macro system which can be used to output mathematical formulae to display or print. For example, the LaTeX code `\frac{-b\pm\sqrt{b^{2}-4ac}}{2a}` will be transformed into [this](http://latex.codecogs.com/gif.latex?%5Cdpi%7B200%7D%20%5Cfrac%7B-b%5Cpm%5Csqrt%7Bb%5E%7B2%7D-4ac%7D%7D%7B2a%7D) when typeset.

The syntax of LaTeX formulae is fairly simple; commands begin with a backslash `\`, followed by the command name, followed by its arguments in curly braces, such as `\sqrt{-1}` (square-root of -1) or `\frac{1}{3}` (1/3 as a fraction). Subscript and superscript are also supported, with the `_` and `^` characters respectively, followed by the script in curly braces - for example, `x^{2}` outputs x^(2). Everything else is output as plain text.

In today's challenge, you'll implement a simplified subset of LaTeX which outputs the resulting formula as ASCII.

# Formal Inputs and Outputs

## Input Specification

You'll be given a LaTeX equation on one line. The commands you need to support are:

* `\frac{top}{bottom}`: A fraction with the given top and bottom pieces
* `\sqrt{content}`: A square-root sign
* `\root{power}{content}`: A root sign with an arbitrary power (eg. cube-root, where the power 3 is at the top-left of the radical symbol)
* `_{sub}`: Subscript
* `^{sup}`: Superscript
* `_{sub}^{sup}`: Subscript and superscript (one on top of the other)
* `\pi`: Output the greek symbol for pi

Feel free to extend your solution to support any additional structures such as integral signs.

## Output Description

Output the formula with ASCII symbols in the appropriate locations. You're free to pick the output style that looks most appropriate to you. One possible way might be something like this:

```
 3_
 √x
y=--
 3
```

# Sample Inputs and Outputs

## Subscripts and Superscripts

### Input

```
log_{e}(e^{x})=x
```

### Output

```
      x
log (e )=x
   e
```

## Stacked Scripts

### Input

```
F_{21}^{3}=2^{5}*7^{3}-30
```

### Output

```
 3   5  3
F  =2 *7 -30
 21
```

## Fractions

### Input

```
sin^{3}(\frac{1}{3}\pi)=\frac{3}{8}\sqrt{3}
```

### Output

```
   3 1   3 _
sin (-π)=-√3
    3   8
```

## Quadratic Formula

### Input

    x=\frac{-b+\sqrt{b^{2}-4ac}}{2a}

### Output

```
       ______
      / 2
  -b+√ b -4ac
x=-----------
      2a
```

## Cubic Formula

(I hope)

### Input

    x=\frac{\root{3}{-2b^{3}+9abc-27a^{2}d+\sqrt{4(-b^{2}+3ac)^{3}+(-2b^{3}+9abc-27a^{2}d)^{2}}}}{3\root{3}{2}a} - \frac{b}{3a} - \frac{\root{3}{2}(-b^{2}+3ac)}{3a\root{3}{-2b^{3}+9abc-27a^{2}d+\sqrt{4(-b^{2}+3ac)^{3}+(-2b^{3}+9abc-27a^{2}d)^{2}}}}

### Output

```
    3_____________________________________________________
   /           ________________________________
  /  3      2  /  2   3   3       2 2                  3_   2
 √  -2b +9abc-27a d+√ 4(-b +3ac) +(-2b +9abc-27a d)   b                  √2(-b +3ac)
x=-------------------------------------------------- - -- - ----------------------------------------------------
               3_                  3a     3_____________________________________________________
              3√2a                         /         ________________________________
                                         /  3      2  /  2   3   3       2 2
                                       3a√  -2b +9abc-27a d+√ 4(-b +3ac) +(-2b +9abc-27a d)
```

# Notes and Further Reading

Solutions have a recommended order of *new* again - feel free to change it back if you prefer *best*. If you want to play around some with LaTeX, try [this online tool](http://www.codecogs.com/latex/eqneditor.php).

Title: [2015-05-29] Challenge #216 [Hard] Texas Hold 'Em 3 of 3 All In
Text: #Description:

For the last part of this week's theme challenge. You have choices.

## Choice 1: Betting

Poker is about money. The betting system is interesting in Texas Hold Em. It involves assigning and moving blinds and inbetween shared cards coming out you have a chance to bet in a cycle until some conditions are meet. For this challenge implement a Texas Hold 'Em Poker betting system with your current progress from the Easy and Intermediate Challenge.

## Choice 2: Simulation

At this point we have a way to run games of different game lengths. We have built a fold AI system based on just cards and not betting. The questions remains, how good is the AI we developed? I think a good way to test it out is run many games and gather some data and see.

For this path of the challenge we want to run many simulations of the game. You will ask for how many players and how many games. At the end you will output data gathered to show some results.

#Betting:

At this point the design/flow of this I would leave to you to develop. Some things to consider in your design:

* Starting Money amount
* CPU betting AI
* Game Ending Conditions. (Player runs out of money or is last player left in game)

I would try to use the fold AI to morph it a bit to help the CPU decide how strong of a hand it thinks it has for the size of the bet. Again the design of how much to bet and if to raise/check/call is up to you. There is no wrong or right choice just the design of how you want it to work.

#Simulation:

Gather the number of cycles to run by asking the user after the amount of players. At the end of all the games we want to see the following data

* Number of Total Rounds/Games played out
* Number of Wins-losses each player had and a percentage of each
* Number of times the best hand equals the highest hand (Remember the best hand includes all hands, including folded AI players vs winning hand was only just best hand of players who did not fold. This potentially could check how good the Fold AI is at deciding to fold)
* Winning hand count - By method (High card, pair, 2 pairs, 3 of a kind, etc) This could be interesting to see what is the most common winning hand

Title: [2015-05-22] Challenge #215 [Hard] Metaprogramming Madness!
Text: #Description

You're working in the devils language. Looser than PHP, more forgiving than Javascript, and more infuriating than LOLCODE.

You've had it up to *here* with this language (and you're a tall guy) so you sit down and think of a solution and then all of a sudden it smacks you straight in the face. Figuratively.

Your comparisons are all over the place since you can't really tell what types evaluate to True and what types evaluate to False. It is in this slightly worrying and dehydrated state that you declare you'll output a truth table for that language in the language!

Armed with a paper cup of saltwater and a lovely straw hat, you set about the task!
Metaprogramming ain't easy but you're not phased, you're a programmer armed with nimble fingers and a spongy brain. You sit down and start typing, *type* ^^*type* ^^^^*type*

...Oh did I mention you're on an island? Yeah there's that too...

#Formal Inputs & Outputs

Given a programming language, output its corresponding truth table. Only the most basic of types need to be included (If you're in a language that doesn't have any of these types, ignore them).

* Int

* Float
* Char
* String
* Array
* Boolean

##Input description

N/A

##Output description

A truth table for the language that you're programming in.

e.g.

Expression	Bool
"Hello World!"	True
''	False
'0'	True
1	True
0	False
0.0	False
[]	False
[1,2,3]	True
True	True
False	False

#Finally

Have a good challenge idea?

Consider submitting it to /r/dailyprogrammer_ideas

Title: [2015-05-15] Challenge #214 [Hard] Chester, the greedy Pomeranian
Text: # Description

Chester is a very spirited young Pomeranian that lives in a pen that exactly covers the unit square. He's sitting in the middle of it, at (0.5, 0.5), minding his own business when out of nowhere, six of the most delicious dog treats you could ever imagine start raining down from the sky.

The treats land at these coordinates:

(0.9, 0.7) (0.7, 0.7) (0.1, 0.1)
(0.4, 0.1) (0.6, 0.6) (0.8, 0.8)

He looks around, startled at his good fortune! He immediately dashes for the closest treat, which is located at (0.6, 0.6). He eats it up, and then runs for the next closest treat, which is at (0.7, 0.7) and eats that up.

He keeps going, always dashing for the nearest treat and eating it up. He's a greedy little puppy, so he keeps going until all the treats have been eaten up. In the end, he's eaten the treats in this order:

(0.6, 0.6), (0.7, 0.7), (0.8, 0.8),

    (0.9, 0.7), (0.4, 0.1), (0.1, 0.1)

Since he started at (0.5, 0.5), he will have travelled a total distance of roughly 1.646710... units.

Your challenge today is to calculate the total length of Chester's journey to eat all of the magically appearing dog-treats.

A small note: distance is calculated using the standard plane distance formula. That is, the distance between a point with coordinates (x1, y1) and a point with coordinates (x2, y2) is equal to:

    sqrt((x1-x2)^2 + (y1-y2)^2)

#Formal inputs &amp; outputs

## Inputs

The first line of the input will be an integer N that will specify how many treats have magically appeared. After that, there will follow N subsequent lines giving the locations of the treats. Each of those lines will have two floating point numbers on them separated by a space giving you the X and Y coordinate for that particular treat.

Each number provided will be between 0 and 1. Except for the first sample, all numbers will be rounded to 8 decimal digits after the period.

Note that most of the inputs I will provide will be in external text files, as they are too long to paste into this description. The bonus input, in particular, is about 2 megabytes large.

## Outputs

You will output a single line with a single number on it, giving the total length of Chester's journey. Remember that he always starts at (0.5, 0.5), before going for the first treat.

#Sample inputs &amp; outputs

##Input 1

```
6
0.9 0.7
0.7 0.7
0.1 0.1
0.4 0.1
0.6 0.6
0.8 0.8
```

##Output 1

```
1.6467103925399036
```

##Input 2

[This file, containing 100 different treats](https://gist.githubusercontent.com/anonymous/4bf5afdc1c85098de9b1/raw/676ca9e4b94668a534854f7c3142f100b4e00f03/sample2.txt)

##Output 2

```
9.127777855837017
```

#Challenge inputs

##Challenge input 1

[This file, containing 1,000 different treats](https://gist.githubusercontent.com/anonymous/5bf6542ebd661804e442/raw/076b6d6dfaf9269f8569b50724efc0ac99013d9b/challenge1.txt)

##Challenge input 2

[This file, containing 10,000 different treats](https://gist.githubusercontent.com/anonymous/c06a78cfc6d2cf7e4acf/raw/559686d0aef082c284e1581b36b4541cb87c7934/challenge2.txt)

#Bonus

[This file, containing 100,000 different treats](https://gist.githubusercontent.com/anonymous/ed9b5f58dc70910e32e9/raw/7c490275414b0c9cea70aabe4a71c907ef435b25/bonus.txt)

I also encourage people to generate their own larger inputs if they find that the bonus is too easy.

Title: [2015-05-08] Challenge #213 [Hard] Stepstring discrepancy
Text: # Description

Define the _discrepancy_ of a string of any two symbols (I'll use `a` and `b`) to be the absolute difference between the counts of each of the two symbols in the string. For example, all of the following strings have a discrepancy of 3:

```
aaa
bbb
abbbb
aababaa
baababbababababababbbaabbaaaabaaabbaa
```

Define a _stepstring_ of a string to be any string that can be formed by starting at any position x in the string, stepping through the string n characters at a time, and ending before any position y. In python, this is any string that can be formed using slice notation `s[x:y:n]`. For example, some stepstrings of the string `"abcdefghij"` are:

```
d
defg
acegi
bdfhj
dfh
beh
ai
abcdefghij
```

Your problem is, given a string of up to 10,000 characters, find the largest discrepancy of any stepstring of the string. For instance, this string:

```
bbaaabababbaabbaaaabbbababbaabbbaabbaaaaabbababaaaaabaabbbaaa
```

has this string as a stepstring (corresponding to the python slice notation `s[4:56:4]`):

```
aaaabaaaaabaa
```

which has a discrepancy of 9. Furthermore, no stepstring has a discrepancy greater than 9. So the correct solution for this string is 9.

# Input Description

A series of strings (one per line) consisting of `a` and `b` characters.

# Output Description

For each string in the input, output the largest discrepancy of any stepstring of the string. (Optionally, also give the slice notation values corresponding to the stepstring with the largest discrepancy.)

# Sample Input

```
bbaaabababbaabbaaaabbbababbaabbbaabbaaaaabbababaaaabaabbbaaa
bbaaaababbbaababbbbabbabababababaaababbbbbbaabbaababaaaabaaa
aaaababbabbaabbaabbbbbbabbbaaabbaabaabaabbbaababababbabbbbaabb
abbabbbbbababaabaaababbbbaababbabbbabbbbaabbabbaaabbaabbbbbb
```

# Sample Output

```
9
12
11
15
```

# Challenge Input:

[Download the challenge input here](http://pastebin.com/raw.php?i=Xt3BV8nK): 8 lines of 10,000 characters each.

# Challenge Output

```
113
117
121
127
136
136
138
224
```

#Note

This problem was inspired by [a recent mathematical discovery](http://www.newscientist.com/article/dn25068-wikipediasize-maths-proof-too-big-for-humans-to-check.html#.Uwa72lK9jAR): the longest string for which your program should output 2 is 1,160 characters. Every string of 1,161 characters will yield a result of 3 or more. The proof of this fact was generated by a computer and is 13 gigabytes long!

Title: [2015-05-01] Challenge #212 [Hard] Reverse Maze Pathfinding
Text: # [](#HardIcon) _(Hard)_: Reverse Maze Pathfinding

We recently saw a maze traversal challenge, where the aim is to find the path through the maze, given the start and end point. Today, however, we're going to do the reverse. You'll be given the maze, and the path from point A to point B as a series of steps and turns, and you'll need to find all the potential candidates for points A and B.

# Formal Inputs and Outputs

## Input Description

You'll first be given a number **N**, which is the number of lines of maze to read. Next, read a further **N** lines of input, containing the maze - a space character describes a place in the maze, and any other non-whitespace character describes a wall. For example:

```
6
xxxxxxxxx
x   x   x
x x x x x
x x x x x
x x   x x
xxxxxxxxx
```

Is exactly equivalent to:

```
6
ERTY*$TW*
f   &   q
@ " @ ` w
' : ; { e
# ^   m r
topkektop
```

(the width of the maze might be anything - you might want to detect this by looking at the width of the first line.)

Finally, you'll be given the path through the maze. The path is contained on a single line, and consists of three possible moves:

* Turn left, represented by the letter `l`.
* Turn right, represented by the letter `r`.
* Move forward *n* spaces, represented by *n*.

An example path might look like `3r11r9l2rr5`, which means to move forward 3 times, turn right, move forward 11 times, turn right, move forward 9 times, turn left, move forward twice, turn right twice and then move forward 5 times. **This path may start pointing in any direction.**

## Output Description

Output the set of possible start and end points, like so: (this example doesn't correspond to the above sample maze.)

```
From (0, 0) to (2, 4)
From (2, 4) to (0, 0)
From (3, 1) to (5, 5)
```

This means that, if you were to travel from any of the given start points to the corresponding end point, the path you take (with the correct initial facing direction) will be the one given in the input.

(Where `(0, 0)` is the top-left corner of the maze.)

# Sample Inputs and Outputs

## Sample 1

### Input

```
5
xxx
x x
x x
x x
```

```
xxx
2rr2ll2
```

### Output

```
From (1, 3) to (1, 1)
From (1, 1) to (1, 3)
```

## Sample 2

### Input

```
9
xxxxxxxxx
x       x
xxx x x x
x   x x x
xxx xxx x
x     x x
x xxx x x
x       x
xxxxxxxxx
2r2r2
```

### Output

```
From (3, 7) to (3, 5)
From (7, 5) to (5, 5)
From (3, 5) to (3, 7)
From (5, 3) to (7, 3)
From (3, 3) to (5, 3)
From (1, 3) to (1, 5)
From (1, 1) to (1, 3)
```

## Sample 3

### Input

```
5
xxxxxxxxx
x   x   x
x x x x x
x   x   x
xxxxxxxxx
2r2r2
```

### Output

```
From (7, 3) to (7, 1)
From (5, 3) to (7, 3)
From (3, 3) to (3, 1)
From (1, 3) to (3, 3)
From (7, 1) to (5, 1)
From (5, 1) to (5, 3)
From (3, 1) to (1, 1)
From (1, 1) to (1, 3)
```

## Sample 4

### Input

```
5
xxxxxxx
x   x x
x x x x
x x   x
xxxxxxx
1l2l2
```

### Output

```
From (3, 2) to (1, 3)
From (3, 2) to (5, 1)
```

## Sample 5

This is a large maze, so the input's on Gist instead.

### [Input](https://gist.githubusercontent.com/Quackmatic/6119dc82b3cfda54f072/raw/maze-mega.txt)

### Output

```
From (1, 9) to (9, 5)
From (137, 101) to (145, 97)
From (169, 53) to (173, 61)
From (211, 121) to (207, 113)
From (227, 33) to (219, 37)
```

## Sample 6

This is another large one.

### [Input](https://gist.githubusercontent.com/Quackmatic/7c548fbe4ccff2c08b5f/raw/maze-long.txt)

### [Output](https://gist.githubusercontent.com/Quackmatic/c1361bcebfdd50874f20/raw/maze-long-out.txt)

Each line of your solution's output for this input should be repeated 4 times, as the path is fully symmetrical.

# Notes

Remember that you can start a path facing in any of four directions, so one starting point might lead to multiple ending points if you start facing different directions - see sample four.

Title: [2015-04-24] Challenge #211 [Hard] Hungry puppies
Text: #Description

Annie has a whole bunch of puppies. They're lovable but also very rambunctious. One day, spur of the moment, Annie decides to get them all treats. She is looking forward to how happy they will all be, and getting ready to serve them the treats, when she realizes: the treats are not all the same size!

This is disastrous! The puppies, knowing the drill, have already lined themselves up in a neat line to receive their treats, so Annie must figure out how to best distribute the unevenly-sized treats so as to make as many puppies happy as possible.

The puppies' jealous reactions to uneven treat distribution is straightforward:

- If a puppy receives a bigger treat than both its neighbors do, it is happy (+1 happiness).
- If a puppy receives a smaller treat than both its neighbors do, it is sad (-1 happiness).
- If a puppy does not fit in either of the above categories, it is merely content. This means any puppy with at least one neighbor with the same size treat, or any puppy with one neighbor with a bigger treat and one with a smaller treat.

Note that the puppies on either end of the line only have a single neighbor to look at, so in their case their mood depends on whether that single neighbor received a bigger, smaller, or equal treat.

Write a program for Annie to recommend a treat distribution that maximizes puppy happiness.

#Formal inputs &amp; outputs
#Input

The input is a single line of positive integers representing the sizes of the treats Annie purchased. For example:

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

Assume there are as many puppies as there are treats. In this case, there are 8 puppies to be served 8 treats of 3 different sizes.

#Output

The output must provide two facts. First, it must display what the maximum achievable happiness is, as a single integer on its own line

```
3
```

Then, it must specify a treat ordering that achieves this number.

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

The puppies on either end of the queue get bigger treats than their sole neighbors, so they are happy. The one in the middle receives a bigger treat than both its neighbors, so it as well is happy. No puppy received a treat that is smaller than both its neighbors', so no puppies are unhappy. Thus, 3 happy puppies minus 0 unhappy puppies results in 3 happiness.

Pictorally:

```
 2  1  1  2  1  1  1  3
:) :| :| :) :| :| :| :)
```

An example of a bad solution would be:

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

The puppies on either end of the line are sad, since their only neighbors have bigger treats, while there is a single happy puppy (the one with the size 3 treat), since it was the only one that had a treat bigger than its neighbors'. This results in a sub-optimal score of -1.

Again, pictorally:

```
 1  2  2  1  1  1  3  1
:( :| :| :| :| :| :) :(
```

Note that it may be possible for there to be several different orderings of the treats that give the maximum happiness. As long as you print out one of them, it doesn't matter *which* one.

#Example inputs and outputs
##Input 1:

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

##Output 1

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

##Input 2:

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

##Output 2:

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

#Challenge inputs

##Challenge input 1

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

##Challenge input 2

```
1 1 2 2 3 4 4 5 5 5 6 6
```

#Bonus

```
1 1 2 2 2 2 2 2 3 4 4 4 5 5 5 6 6 6 7 7 8 8 9 9 9 9 9 9 9 9
```

Title: [2015-04-17] Challenge #210 [Hard] Loopy Robots
Text: # Description

Our robot has been deployed on an infinite plane at position (0, 0) facing north. He's programmed to *indefinitely* execute a command string. Right now he only knows three commands

* S - Step in the direction he's currently facing
* R - Turn right (90 degrees)
* L - Turn left (90 degrees)

It's our job to determine if a command string will send our robot into an endless loop. (It may take many iterations of executing the command!) In other words, will executing some
command string enough times bring us back to our original coordinate, in our original orientation.

Well, technically he's stuck in a loop regardless.. but we want to know if he's going in a circle!

# Input Description

You will accept a command string of arbitrary length. A valid command string will only contain the characters "S", "R", "L" however it is not required that a command string utilizes all commands. Some examples of valid command strings are

* S
* RRL

* SLLLRLSLSLSRLSLLLLS

# Output Description

Based on robot's behavior in accordance with a given command string we will output one of two possible solutions

A) That a loop was detected and how many cycles of the command string it took to return to the beginning of the loop

B) That no loop was detected and our precious robot has trudged off into the sunset

# Input

* "SR" (Step, turn right)
* "S" (Step)

# Output

* "Loop detected! 4 cycle(s) to complete loop" [[Visual](http://i.imgur.com/kGsoPSX.png)]
* "No loop detected!"

# Challenge Input

* SRLLRLRLSSS
* SRLLRLRLSSSSSSRRRLRLR
* SRLLRLRLSSSSSSRRRLRLRSSLSLS
* LSRS

Title: [2015-04-10] Challenge #209 [Hard] Unpacking a Sentence in a Box
Text: Those of you who took the time to work on a Hamiltonian path generator can build off of that.

# Description

You moved! Remember on Wednesday we had to pack up some sentences in boxes. Now you've arrived where you're going and you need to unpack.

You'll be given a matrix of letters that contain a coiled sentence. Your program should walk the grid to adjacent squares using only left, right, up, down (no diagonal) and every letter exactly once. You should wind up with a six word sentence made up of regular English words.

# Input Description

Your input will be a list of integers *N*, which tells you how many lines to read, then the row and column (indexed from 1) to start with, and then the letter matrix beginning on the next line.

```
6 1 1
T H T L E D
P E N U R G
I G S D I S
Y G A W S I
W H L Y N T
I T A R G I
```

(Start at the T in the upper left corner.)

# Output Description

Your program should emit the sentence it found. From the above example:

```
    THE PIGGY WITH LARYNGITIS WAS DISGRUNTLED
```

# Challenge Input

```
    5 1 1
    I E E H E
    T K P T L
    O Y S F I
    U E C F N
    R N K O E
```

(Start with the I in the upper left corner, but this one is a 7 word sentence)

# Challenge Output

```
    IT KEEPS YOUR NECK OFF THE LINE
```

Title: [2015-04-03] Challenge #208 [Hard] The Universal Machine
Text: # [](#HardIcon) _(Hard)_: The Universal Machine

Imagine an infinitely long, one-dimensional list of symbols. The list is infinite in both directions, and each symbol is indexed by a number, where the middle of the list is zero. This is called a **tape**. The symbols on the tape can be any symbol from an **alphabet**, which is just a set of possible symbols. If our example alphabet consists of the symbols `0`, `1` and `#`, then a valid tape would look like:

```
    #0110#10101#111#01##
    |
```

(The `|` marks the location of the middle of the tape, position zero.) Of course, we can't represent an infinite tape at once. Therefore, we add another possible symbol to our alphabet, `_` (underscore), to denote the lack of a symbol. This `_` symbol fills the rest of the tape, all the way out to infinity, like so (ellipsis denotes repeat):

```
    . . . __________________#0110#10101#111#01##__________________ . . .
                            |
```

Now, imagine we have a **machine** that can look at this tape, but it can only see one symbol on the tape at once. To look at this tape, it has a **read head**. In our tape diagrams, the read head is marked with a caret (`^`). For example, here's the read head at position 7:

```
    #0110#10101#111#01##
    |      ^
```

This read head can move one symbol to the left or right, but it can't skip ahead arbitrarily or jump to a specific location. Besides the read head, the machine also has a **state**. This is just an alphanumeric string, with no spaces, like a variable of the machine. It could be `Red`, it could be `Clockwise`, it could be `Catch22`, it could be `Tgnqovjaxbg`, as long as it's alphanumeric.

Now, this machine is capable of performing a **step**. A step will change the symbol under the read head to another symbol from the alphabet, and then either move the read head left or right. The type of step that happens depends on the current state, and the current symbol under the read head. We can define a rule for our machine which says something like this:

> If the current symbol under the read head is **p**, and the current state is **A**, then change the state to **B**, change the symbol under the read head to **q** and move the read head in direction **d**.

p and **q** can be the same symbol, and **A** and **B** can be the same state. For example:

> If the current symbol under the read head is `0`, and the current state is `State1`, then change the state to `State1`, change the symbol under the read head to `1` and move the read head left.

This rule is called a **transition function**, and the typical notation is:

    𝛿(A, p) = (B, q, d)

So our example rule is:

    𝛿(State1, 0) = (State1, 1, <)

So, if our machine is in the state `State1` and our tape looks like this:

```
#0110#10101#111#01##
|     ^
```

Then, after applying our transition function above, we're now in `State1` (again) and the tape now looks like this:

```
#0110#11101#111#01##
|    ^
```

You'll typically have quite a few transition functions to cover every possible state/symbol combination. After each step, the machine compares the new state to a special state known as the **accepting state**. If the current state is the accepting state, then the machine stops, as the computation is complete.

Whew, that's a lot of information! Here's the synopsis of what you need to describe one of these machines:

* The **alphabet**: all possible symbols (along with `_`, which is implicitly part of the alphabet.)
* The **tape** at the start of the computation.
* The **starting position** of the read head on the tape; this is assumed to be zero.
* The **list of possible** states that our machine can be in.
* The **starting state**, and the **accepting state**.
* A list of **transition functions**, that cover every possible symbol/state combination on the given tape.

This type of machine is known as a [**Turing machine**](http://en.wikipedia.org/wiki/Turing_machine), named after the famous groundbreaking computer scientist and mathematician [Alan Turing](http://en.wikipedia.org/wiki/Alan_Turing). It sounds hopelessly verbose in practice, but a Turing machine is insanely useful as a theoretical model for computation. To gloss over quite a few details: if a machine can simulate any such Turing machine as described above, then it's described as **Turing-complete**. Today, you'll be writing a program to simulate a turing machine given the above required parameters; therefore, you'll need to use a Turing-complete language to solve this challenge. :)

# Formal Inputs and Outputs

## Input Description

First, you will be given the alphabet of a Turing machine (excluding `_`, which is always part of the alphabet) as a sequence of non-whitespace characters, like so:

```
01
```

Next, you will be given a space-separated list of possible states for the machine, like this:

```
Mov B Bi OK
```

You will then be given the initial state, and the accepting state, on two lines:

```
Mov
OK
```

After this, you will be given the initial tape to use. The first character is assumed to be at position 0, with following characters at successive locations (1, 2, 3, etc.), like so:

```
01100100
```

Finally, you're given a list of transition rules. These are in the format `StateBefore SymbolBefore = StateAfter SymbolAfter DirectionToMove`, like this list:

```
Mov 0 = Mov 0 >
Mov 1 = Mov 1 >
Mov _ = B _ <
B 0 = B 0 <
B 1 = Bi 1 <
B _ = OK _ >
Bi 0 = Bi 1 <
Bi 1 = Bi 0 <
Bi _ = OK _ >
```

The direction is either `<` for left, or `>` for right.

## Output Description

You are to output the tape after the computation has finished. You are also to output the symbol `|` (pipe) under the middle of the tape (position zero), and output the symbol `^` (caret) under the position of the read head after the computation has finished. If the `^` and `|` would be in the same place (ie. the read head finishes at the middle of the tape), just print only the `|`.

```
10011100
|
```

# Sample Turing Machines

## Machine 1: Two's Complement

This machine computes the two's complement of the binary number in the input. Notice how the transition functions can use the `_` symbol, even though it's not explicitly part of the alphabet.

### Input

```
01
Mov B Bi OK
Mov
OK
01100100
Mov 0 = Mov 0 >
Mov 1 = Mov 1 >
Mov _ = B _ <
B 0 = B 0 <
B 1 = Bi 1 <
B _ = OK _ >
Bi 0 = Bi 1 <
Bi 1 = Bi 0 <
Bi _ = OK _ >
```

### Output

```
10011100
|
```

## Machine 2: Moving Morse Code

This machine takes input as dots (`.`) and dashes (`/`), including a delimiter symbol, `k`. The dots and dashes are moved to after the `k`.

### Input

```
./k
Init Mdot MDash Ret OK
Init
OK
/./../.../..../k
Init _ = Init _ >
Init . = Mdot _ >
Init / = MDash _ >
Init k = OK k >
Mdot . = Mdot . >
Mdot / = Mdot / >
Mdot k = Mdot k >
Mdot _ = Ret . <
MDash . = MDash . >
MDash / = MDash / >
MDash k = MDash k >
MDash _ = Ret / <
Ret . = Ret . <
Ret / = Ret / <
Ret k = Ret k <
Ret _ = Init _ >
```

### Output

```
__________________k/./../.../..../
|                   ^
```

Notice all the spaces in the output, as the dots and dashes are now not centered on the middle of the tape.

## Machine 3: Copying

This machine takes a binary input string, including a delimiter symbol at the end. The binary string is copied to after the delimiter symbol.

### Input

```
01xy#
C0 C1 Ret Search OK
Search
OK
0110100#
Search 0 = C0 x >
Search 1 = C1 y >
Search # = OK # >
C0 0 = C0 0 >
```

```
C0 1 = C0 1 >
C0 # = C0 # >
C0 _ = Ret 0 <
C1 0 = C1 0 >
C1 1 = C1 1 >
C1 # = C1 # >
C1 _ = Ret 1 <
Ret 0 = Ret 0 <
Ret 1 = Ret 1 <
Ret # = Ret # <
Ret x = Search 0 >
Ret y = Search 1 >
```

### Output

```
0110100#0110100
|   ^
```

# Notes and Further Reading

The Wolfram MathWorld [page on Turing Machines](http://mathworld.wolfram.com/TuringMachine.html) has some more description of the concept of turing machines. Sometimes cell 'colours' are used rather than 'symbols', but the over-arching concept is always the same.

Title: [2015-03-32] Challenge 208 [Bonus] The Infinite Stallman Theorem
Text: # [](#BonusIcon) _(Bonus)_: The Infinite Stallman Theorem

Loosely, the [*infinite monkey theorem*](http://en.wikipedia.org/wiki/Infinite_monkey_theorem) states that, given an infinite number of monkeys randomly typing at typewriters for an unbounded amount of time, one will eventually produce a work of Shakespeare from start to finish. After the Japanese government performed this thought experiment using an infinitely-nested fractal pocket dimension with some success in 2007 (despite the incident with the micro black holes), application of the theorem has had some practical value in the field of [software optimization](http://en.wikipedia.org/wiki/Program_optimization).

Human-developed programs often suffer from performance issues, such as the during the compilation of large programs. Even today's compilers, such as the GNU Compiler Collection or Clang, take a non-trivial amount of time to compile complex systems. Partly, this boils down to limitations of human thought process when designing the architecture of such systems. This problem can be alleviated by randomly-generating the program instead, using the theorem described above. This is known as using an **evolutionary algorithm**, named by Charles Darwin who produced the first artifically-generated Smalltalk compiler in 1866.

Today, your challenge is to randomly generate a C++ compiler, by simulating a monkey entering random characters on a typewriter, and testing the resulting source code. The randomly-generated compiler can be implemented in a language of your choice.

# Formal Inputs and Outputs

## Input Description

The input to this challenge consists of a single line, describing the C++ standard which the generated compiler should accept. This can consist of any of the following strings:

* `c++98` for the C++98 standard.
* `c++03` for the 2003 revision of the C++98 standard.
* `c++11` for the C++11 standard (previously known as C++0x).
* `c++14` for the 2014 revision of the C++11 standard.

## Output Description

You are to output a tarball of the source of a randomly-generated C++ compiler, compliant to the standard provided as input to the challenge.

# Sample Inputs and Outputs

## Sample Input

```
c++14
```

Title: [2015-03-27] Challenge #207 [Hard] Bioinformatics 3: Predicting Protein Secondary Structures
Text: Wrapping up our bioinformatics theme with the third and final installment 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

The Chou-Fasman method is an empirical technique for the prediction of secondary structures in proteins, originally developed in the 1970s by Peter Y. Chou and Gerald D. Fasman. The method is based on analyses of the relative frequencies of each amino acid in alpha helices, beta sheets, and turns based on known protein structures. From these frequencies a set of probability parameters were derived for the appearance of each amino acid in each secondary structure type, and these parameters are used to predict the probability that a given sequence of amino acids would form a helix, a beta strand, or a turn in a protein. The method is at most about 50–60% accurate in identifying correct secondary structures, and is mostly of historical significance at this point (it's been updated by better methods).

The Chou-Fasman method predicts helices and strands in a similar fashion, first searching linearly through the sequence for a "nucleation" region of high helix or strand probability and then extending the region until a subsequent four-residue window carries a probability of less than 1. As originally described, four out of any six contiguous amino acids were sufficient to nucleate helix, and three out of any contiguous five were sufficient for a sheet. The probability thresholds for helix and strand nucleations are constant but not necessarily equal; originally 1.03 was set as the helix cutoff and 1.00 for the strand cutoff.

You can find a table showing propensities for an amino acid to help form an alpha-helix or a beta-sheet at [this link](http://employees.csbsju.edu/hjakubowski/classes/ch331/protstructure/tablechoufas.htm) or [this one](http://prowl.rockefeller.edu/aainfo/chou.htm) along with an algorithm description.

You can learn more about the [Chou-Fasman method via Wikipedia](http://en.wikipedia.org/wiki/Chou%E2%80%93Fasman_method). Also, slide 17 of [this deck](http://www.slideshare.net/RoshanKarunarathna1/chou-fasman-algorithm-for-protein-structure) describes the approach quite cleanly.

In this challenge you'll be given a protein sequence and asked to suggest its secondary structure.

# Input

```
MET LYS ILE ASP ALA ILE VAL GLY ARG ASN SER ALA LYS ASP ILE ARG THR GLU GLU ARG ALA ARG
VAL GLN LEU GLY ASN VAL VAL THR ALA ALA ALA LEU HIS GLY GLY ILE ARG ILE SER ASP GLN THR
THR ASN SER VAL GLU THR VAL VAL GLY LYS GLY GLU SER ARG VAL LEU ILE GLY ASN GLU TYR
GLY GLY LYS GLY PHE TRP ASP ASN HIS HIS HIS HIS HIS HIS
```

# Output

Here's the results of an **[online Chou-Fasman prediction](http://www.biogem.org/cgi-bin/cho-fas.pl)** for the 2RNM sequence showing predicted helices and sheets:

```
Met Lys Ile Asp Ala Ile Val Gly Arg Asn Ser Ala Lys Asp Ile Arg Thr Glu Glu Arg Ala Arg
H   H   H   H   H   H               H   H   H   H   H   H   H   H   H
```

```
Val Gln Leu Gly Asn Val Val Thr Ala Ala Ala Leu His Gly Gly Ile Arg Ile Ser Asp Gln Thr
H  H  H  H  H  H  H  H  H  H  H  H
B  B  B  B  B  B  B  B  B  B                 B  B  B  B

Thr Asn Ser Val Glu Thr Val Val Gly Lys Gly Glu Ser Arg Val Leu Ile Gly Asn Glu Tyr

         H  H  H
B  B  B  B  B  B  B  B

Gly Gly Lys Gly Phe Trp Asp Asn His His His His His His
```

And here is the **empirically determined** secondary structure from x-ray crystallography, based on http://pdbj.org/mine/structural_details/2rnm. Note that no alpha helices were found in the structure, so that row is blank. You can see how far off the prediction method and experiment can be:

```
MET LYS ILE ASP ALA ILE VAL GLY ARG ASN SER ALA LYS ASP ILE ARG THR GLU GLU ARG ALA ARG

                        B  B  B  B  B  B

VAL GLN LEU GLY ASN VAL VAL THR ALA ALA ALA LEU HIS GLY GLY ILE ARG ILE SER ASP GLN THR

B  B  B      B  B

THR ASN SER VAL GLU THR VAL VAL GLY LYS GLY GLU SER ARG VAL LEU ILE GLY ASN GLU TYR

B  B  B  B  B  B  B  B  B           B  B  B  B       B  B

GLY GLY LYS GLY PHE TRP ASP ASN HIS HIS HIS HIS HIS HIS
```

# Notes

Other interesting proteins you could analyze include 1VPX or 1BKF, they'll give you some mixed structures. Use the European Protein Databank site (for example for [1VPX](http://www.ebi.ac.uk/pdbe-srv/view/entry/1vpx/secondary.html)) to confirm your results.

Title: [2015-03-26] Challenge #207 [Bonus] Erdos Number Calculator
Text: In honor of the 102nd birthday of the famous mathematician, a problem named after him.

# Description

Paul Erdős (1913–1996) was an influential mathematician who spent a large portion of his later life writing papers with a large number of colleagues, working on solutions to outstanding mathematical problems. The idea of the Erdős number was originally created by the mathematician's friends as a tribute to his enormous output. However, in later years it gained prominence as a tool to study how mathematicians cooperate to find answers to unsolved problems.

The Erdös number describes the "collaborative distance" between mathematician Paul Erdős and another person, as measured by authorship of mathematical papers. Erdös himself has a number of 0, anyone he co-authored a paper with has a number of 1, anyone they co-authored a paper with (without Erdös) has a number of 2, and so forth.

Several studies have shown that leading mathematicians tend to have particularly low Erdős numbers. For example, only 134,007 mathematicians have an Erdős number, with a median value of 5. In contrast, the median Erdős number of Fields Medalists is 3. Only 7,097 (about 5%) of mathematicians with a collaboration path have an Erdős number of 2 or less.

For this challenge you'll be working with a small, toy database of Erdős and related publications and be asked to calculate the Erdős number for several authors.

# Input Description

You'll be given 2 integers on the first line, *N* and *M*. *N* is the number of of papers in APA format showing authors, titles, journals, and the like; think of it as a mini database. *M* is the number of authors to identify the Erdős number for. Note that all of the names should be in the same format of last name, then first and middle initials.

# Output

Your program should emit the name of the mathematician and their Erdős number.

# Challenge Input

```
7 4
Thomassen, C., Erdös, P., Alavi, Y., Malde, P. J., & Schwenk, A. J. (1989). Tight bounds on the chromatic sum of a connected graph. Journal of Graph Theory, 13(3), 353-357.
Burr, S., Erdös, P., Faudree, R. J., Rousseau, C. C., & Schelp, R. H. (1988). Some complete bipartite graph—tree Ramsey numbers. Annals of Discrete Mathematics, 41, 79-89.
Burris, A. C., & Schelp, R. H. (1997). Vertex-distinguishing proper edge-colorings. Journal of graph theory, 26(2), 73-82.
Balister, P. N., Gyo˝ ri, E., Lehel, J., & Schelp, R. H. (2007). Adjacent vertex distinguishing edge-colorings. SIAM Journal on Discrete Mathematics, 21(1), 237-250.
Erdös, P., & Tenenbaum, G. (1989). Sur les fonctions arithmétiques liées aux diviseurs consécutifs. Journal of Number Theory, 31(3), 285-311.
Hildebrand, A., & Tenenbaum, G. (1993). Integers without large prime factors. Journal de théorie des nombres de Bordeaux, 5(2), 411-484.
Balister, P. N., Riordan, O. M., & Schelp, R. H. (2003). Vertex-distinguishing edge colorings of graphs. Journal of graph theory, 42(2), 95-109.
Schelp, R. H.
Burris, A. C.
Riordan, O. M.
Balister, P. N.
```

# Challenge Output

```
Schelp, R. H. 1
Burris, A. C. 2
Riordan, O. M. 2
Balister, P. N. 2
```

# Notes

This challenge is a shameless rip off of http://www.programming-challenges.com/pg.php?page=downloadproblem&format=html&probid=110206. It was too good to pass up; I did, however, compile my own challenge inputs and outputs.

A full list of Erdös publications is up here http://www.renyi.hu/~p_erdos/Erdos.html.

# Finally

Have a good challenge idea? Consider submitting it to /r/dailyprogrammer_ideas

Title: [2014-03-20] Challenge #206 [Hard] Recurrence Relations, part 2
Text: # [](#HardIcon) _(Hard)_: Recurrence Relations, part 2

In [Monday's challenge](/r/dailyprogrammer/comments/2z68di/), we wrote a program to compute the first *n* terms of a simple recurrence relation. These recurrence relations depended only on the directly previous term - that is, to know *u*(n), you only need to know *u*(n-1). In today's challenge, we'll be investigating more complicated recurrence relations.

In today's recurrence relations, the relation given will only depend on terms *preceding* the defined tern, not terms *following* the defined term. For example, the relation for *u*(n) will never depend on *u*(n+1). Let's look at the Fibonacci sequence as defined by [OEIS](http://oeis.org/A000045):

```
    u(0) = 0
    u(1) = 1
    u(n) = u(n-1) + u(n-2)
```

This relation provides a definition for the first two terms - the 0th term and the 1st term. It also says that the *n*-th term is the sum of the two previous terms - that is, the *(n-1)*-th term and the *(n-2)*-th term. As we know terms 0 and 1, we therefore know term 2. As we know term 1 and 2, we know term 3, and so on - for this reason, the Fibonacci sequence is **completely defined** by this recurrence relation - we can compute an infinite number of Fibonacci numbers after the first two, given two defined terms.

However, now let's look at this recurrence relation:

```
    u(0) = 0
    u(1) = 1
    u(2) = 3
    u(n) = u(n-1) * u(n-2) + u(n-5)
```

We're given the 0th, 1st and 2nd terms. However, the relation for the *n*-th term depends on the *(n-5)*-th term. This means we can't calculate the value of *u*(3), as we'll need the term 5 before that - ie. *u*(-2), which we don't have. We can't calculate *u*(4) for the same reason. We find that, to try and define the 3rd term and beyond, we don't have enough information, so this series is **poorly defined** by this recurrence relation. Therefore, all we know about the series is that it begins `[0, 1, 3]` - and, as far as we know, that's the end of the series.

Here's another example of a recurrence relation with a twist:

```
    u(1) = 0
    u(n) = u(n-2) * 2 + 1
```

This relation defines the 1st term. It also defines the *n*-th term, with respect to the *(n-2)*-th term. This means we know the 3rd term, then the 5th term, then the 7th term... but we don't know about the even-numbered terms! Here is all we know of the series:

```
    0, ?, 1, ?, 3, ?, 7, ?, 15, ?, ...
```

There are an infinite number of terms that we *do* know, but there are terms in-between those that we don't know! We only know half of the series at any given time. This is an example of a series being **partially defined** by a recurrence relation - we can work out some terms, but not others.

Your challenge today is, given a set of initial terms and a recurrence relation, work out *as many further terms as possible*.

# Formal Inputs and Outputs

## Input Description

You will accept the recurrence relation in **reverse Polish notation** (or postfix notation). If you solved [last Wednesday's challenge](/r/dailyprogrammer/comments/2yquvm/), you may be able to re-use some code from your solution here. To refer to the *(n-k)*-th term, you write `(k)` in the RPN expression. Possible operators are `+`, `-`, `*` and `/` (but feel free to add any of your own). For example, this recurrence relation input defines the *n*-th term of the Fibonacci sequence:

```
(2) (1) +
```

This means that the *n*-th term is the *(n-2)*-th term and the *(n-1)*-th term, added together. Next, you will accept any number of pre-defined terms, in the format `index:value`. For example, this line of input:

```
2:5.333
```

Defines the 2nd term of the series to be equal to 5.333. For example, the initial terms for the Fibonacci sequence are:

```
0:0
1:1
```

Finally, you will accept a number - this will be the maximum *n* of the term to calculate. For example, given:

```
40
```

You calculate as many terms as you possibly can, up to and including the 40th term.

## Output Description

The output format is identical to the Easy challenge - just print the term number along with the term value. Something like this:

```
0: 0
1: 1
2: 1
3: 2
4: 3
5: 5
6: 8
7: 13
8: 21
```

is good.

# Sample Input and Outputs

## Fibonacci Sequence

This uses the OEIS definition of the Fibonacci sequence, starting from 0.

### Input

```
(1) (2) +
0:0
1:1
20
```

### Output

```
0: 0
1: 1
2: 1
3: 2
4: 3
5: 5
6: 8
7: 13
8: 21
```

```
9: 34
10: 55
11: 89
12: 144
13: 233
14: 377
15: 610
16: 987
17: 1597
18: 2584
19: 4181
20: 6765
```

## Oscillating Sequence

This defines an oscillating sequence of numbers starting from the 5th term. The starting term is not necessarily zero!

### Input

```
0 (1) 2 * 1 + -
5:31
14
```

### Output

```
5: 31
6: -63
7: 125
8: -251
9: 501
10: -1003
11: 2005
12: -4011
13: 8021
14: -16043
```

## Poorly Defined Sequence

This sequence is poorly defined.

### Input

```
(1) (4) * (2) 4 - +
0:3
1:-2
3:7
4:11
20
```

### Output

The 5th term can be defined, but no further terms can.

```
0: 3
1: -2
3: 7
4: 11
5: -19
```

## Staggered Tribonacci Sequence

This uses the [OEIS definition](https://oeis.org/A000073) of the Tribonacci sequence, but with a twist - the odd terms are undefined, so this is partially defined.

### Input

```
(2) (4) (6) + +
0:0
2:0
4:1
30
```

### Output

```
0: 0
2: 0
4: 1
6: 1
8: 2
10: 4
12: 7
14: 13
16: 24
18: 44
20: 81
22: 149
24: 274
26: 504
28: 927
30: 1705
```

# Notes

Relevant links:

* [Reverse Polish notation](http://en.wikipedia.org/wiki/Reverse_Polish_notation)
* [Recurrence relation](http://en.wikipedia.org/wiki/Recurrence_relation)

Declarative languages might be handy for this challenge!

Title: [2015-03-13] Challenge #205 [Hard] DNA and Protein Sequence Alignment
Text: #Description

If you are studying a particular pair of genes or proteins, an important question is to what extent the two sequences are similar. To quantify similarity, it is necessary to align the two sequences, and then you can calculate a similarity score based on the alignment.

There are two types of alignment in general. A global alignment is an alignment of the full length of two sequences, for example, of two protein sequences or of two DNA sequences. A local alignment is an alignment of part of one sequence to part of another sequence.

Alignment treats the two inputs as a linear sequence to be lined up as much as possible, with optional gaps and conversions allowed. The goal is to minimize these differences.

The first step in computing a sequence alignment is to decide on a scoring system. For this exercise, we'll simplify this and give a score of +2 to a match and a penalty of -1 to a mismatch, and a penalty of -2 to a gap.

Here's a small example. Our two DNA sequences to align:

```
CTCTAGCATTAG
GTGCACCCA
```

One alignment might look like this:

```
CTCTAGCATTAG
GT---GCACCCA
```

But that one adds three gaps. We can do a bit better with only one gap added (and a small shift in starting position):

```
CTCTAGCATTAG
  GT-GCACCCA
```

While not an exact match, it now minimizes the conversion penalties between the two and aligns them as best we can.

For more information and how to do this using an R package, see the chapter ["Pairwise Sequence Alignment"](http://a-little-book-of-r-for-bioinformatics.readthedocs.org/en/latest/src/chapter4.html), or [this set of lecture notes from George Washington University](http://www.seas.gwu.edu/~simhaweb/cs151/lectures/module12/align.html). The key algorithm is [Needleman-Wunsch](http://en.wikipedia.org/wiki/Needleman%E2%80%93Wunsch_algorithm).

For this challenge your task is to write a program that accepts two sequences and globally aligns them. If you want to make this harder and integrate the BLOSUM matrices, you may.

#Input Description

You'll be given two sequences on two lines, one line per sequence. They'll be the same type of input, DNA or protein.

#Output Description

Your program should emit the aligned sequences with gaps introduced represented by dashed ("-").

#Input

DNA example

```
GACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTAC
ACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTAC
```

Protein example

MTNRTLSREEIRKLDRDLRILVATNGTLTRVLNVVANEEIVVDIINQQLLDVAPKIPELENLKIGRILQRDILLKGQKSGILFVAAESLIVIDLLPTAITTYLTKTHH
PIGEIMAASRIETYKEDAQVWIGDLPCWLADYGYWDLPKRAVGRRYRIIAGGQPVIITTEYFLRSVFQDTPREELDRCQYSNDIDTRSGDRFVLHGRVFKN

MLAVLPEKREMTECHLSDEEIRKLNRDLRILIATNGTLTRILNVLANDEIVVEIVKQQIQDAAPEMDGCDHSSIGRVLRRDIVLKGRRSGIPFVAAESFIAIDLL
PPEIVASLLETHRPIGEVMAASCIETFKEEAKVWAGESPAWLELDRRRNLPPKVVGRQYRVIAEGRPVIIITEYFLRSVFEDNSREEPIRHQRSVGTSARSGRS
ICT

#Output

DNA example

```
GACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTAC
```

```
    ACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTACGTAC
```

Protein example

MTNRTLSREEIRKLDRDLRILVATNGTLTRVLNVVANEEIVVDIINQQLLDVAPKIPELENLKIGRILQRDILLKGQKSGILFVAAESLIVIDLLPTAITTYLTKTHH
PIGEIMAASRIETYKEDAQVWIGDLPCWLADYGYWDLPKRAVGRRYRIIAGGQPVIITTEYFLRSVFQDTPREELDRCQYSNDIDTRSGDRFVLHGRVFKN

MLAVLPEKREMTECHLSDEEIRKLNRDLRILIATNGTLTRILNVLANDEIVVEIVKQQIQDAAPEMDGCDHSSIGRVLRRDIVLKGRRSGIPFVAAESFIAIDLL
PPEIVASLLETHRPIGEVMAASCIETFKEEAKVWAGESPAWLELDRRRNLPPKVVGRQYRVIAEGRPVIIITEYFLRSVFEDNSREEPIRHQRS--VGT-SA-R---
SGRSICT

#Notes

Once you have a simple NW algorithm implemented, you can alter the cost matrices. In the bioinformatics field, the PAM and BLOSUM matrices are the standards. You can find them here: ftp://ftp.ncbi.nih.gov/blast/matrices/

Title: [2015-03-06] Challenge #204 [Hard] Addition chains
Text: #Description

An "addition chain" is a sequence of numbers that starts with 1 and where each number is the sum of two previous numbers (or the same number taken twice), and that ends at some predetermined value.

An example will make this clearer: the sequence [1, 2, 3, 5, 10, 11, 21, 42, 84] is an addition chain for the number 84. This is because it starts with 1 and ends with 84, and each number is the sum of two previous numbers. To demonstrate:

```
                (chain starts as [1])
    1 + 1   = 2     (chain is now [1, 2])
    1 + 2   = 3     (chain is now [1, 2, 3])
    2 + 3   = 5     (chain is now [1, 2, 3, 5])
    5 + 5   = 10    (chain is now [1, 2, 3, 5, 10])
    1 + 10  = 11    (chain is now [1, 2, 3, 5, 10, 11])
    10 + 11 = 21    (chain is now [1, 2, 3, 5, 10, 11, 21])
    21 + 21 = 42    (chain is now [1, 2, 3, 5, 10, 11, 21, 42])
    42 + 42 = 84    (chain is now [1, 2, 3, 5, 10, 11, 21, 42, 84])
```

Notice that the right hand side of the equations make up the chain, and left hand side of all the equations is a sum of two numbers that occur earlier in the chain (sometimes the same number twice).

We say that this chain is of length 8, because it took 8 additions to generate it (this is one less than the total amount of numbers in the chain).

There are a several different addition chains of length 8 for the number 84 (another one is [1, 2, 4, 8, 16, 32, 64, 68, 84], for instance), but there are no shorter ones. This is as short as we can get.

Your task today is to try and generate addition chains of a given length and last number.

(by the way, you may think this looks similar to the Fibonacci sequence, but it's not, there's a crucial difference: you don't just add the last two numbers of the chain to get the next number, you can add *any* two previous numbers to get the next number. The challenge is figuring out, for each step, which two numbers to add)

#Formal inputs & outputs

##Input description

You will be given one line with two numbers on it. The first number will be the length of the addition chain you are to generate, and the second the final number.

Just to remind you: the length of the addition chain is equal to the number of additions it took to generate it, which is the same as **one less** than the total amount of numbers in it.

##Output description

You will output the entire addition chain, one number per line. There will be several different addition chains of the given length, but you only need to output one of them.

Note that going by the strict definition of addition chains, they don't necessarily have to be strictly increasing. However, any addition chain that is not strictly increasing can be reordered into one that is, so you can safely assume that all addition chains are increasing. In fact, making this assumption is probably a very good idea!

#Examples

##Input 1

```
7 43
```

##Output 1

(one of several possible outputs)

```
1
2
3
5
10
20
40
43
```

##Input 2

```
9 95
```

##Output 2

(one of several possible outputs)

```
1
2
3
5
7
14
19
38
57
95
```

#Challenge inputs

##Input 1

```
    10 127
```

##Input 2

```
    13 743
```

#Bonus

```
    19 123456
```

If you want *even more* of a challenge than that input, consider this: when I, your humble moderator, was developing this challenge, my code would not be able to calculate the answer to this input in any reasonable time (even though solutions exist):

```
    25 1234567
```

If you can solve that input, you will officially have written a much better program than me!

#Notes

I would like to note that while this challenge looks very "mathy", you don't need any higher level training in mathematics in order to solve it (at least not any more than is needed to understand the problem). There's not some secret formula that you have to figure out. It's still not super-easy though, and a good working knowledge of programming techniques will certainly be helpful!

In other words, in order to solve this problem (and especially the bonus), you need to be clever, but you don't need to be a mathematician.

Title: [2015-2-27] Challenge #203 [Hard] Minecraft: There and Back
Text: #Description:

In the popular game Minecraft (http://en.wikipedia.org/wiki/Minecraft) you navigate a 3-D block world. Each block can be various types. You gather blocks to place blocks. More or less.

Part of the challenge is navigating this world such that you have to mine down and be able to get back up. So for this challenge we will be throwing at you some combined challenges to solve. Users can select which level of involvement. If you feel you have time or ability solve which challenges you can.

The 3 challenges to solve (Easy, Intermediate and Hard)

* Generate a 3-D Minecraft Map with a fixed starting point and fixed point for the goal.
* Navigate the map to find a *shortest* and *safe*path down and back again. (if possible)
* Generate a 3-D map with a fixed starting point but a random end point. You must develop an agent program to seek out the unknown goal safely and return.

#The Map

To generate a world we are going to keep our minecraft world simple. Each block can be the following:

* Air - Basically nothing
* Dirt - Block which can be removed
* Sand - Block which can be removed but obeys differently than dirt
* Lava - Dangerous block which we have to avoid.
*Diamond block - Our goal block we wish to mine that block and leave air.

##Air Block

Our player can occupy an Air block. If they are standing on top of an air block they will drop down to the block below the air block. As you can imagine if it is another air block they keep dropping down until they hit the bottom of the map.

## Dirt Block

Our player can remove any dirt block adjacent to the player that is not diagonal. So if you image 3x3x3 blocks. And if the player is in the exact middle they can only remove dirt blocks up, down and the 4 blocks around him. The corners could not be removed because it would be diagonal.

A removed dirt block becomes air.

## Sand Block

A sand block works like a dirt block. It follows our gravity. If there is an air block below a sand block it will fall (leaving an air block where it was) until the block below the sand block is not an air block. After generating a map you will have to adjust the map to have any sand fall into place.

##Lava Block

A Lava block if you touch it you die. Not good. Lava as a liquid can flow. To keep it simple the rule for lava is if you have air below a lava block the block below lava becomes lava. It will keep becoming lava until the block below lava is not air. Think of it as a Sand Block but it does not "fall" and leave behind air blocks but "flows"

## Hero

The hero occupies only air blocks. He cannot be inside any other block. To move he will be removing blocks. He can remove Dirt and sand blocks trying to get to the diamond block. He can only remove blocks next to him but not diagonal. Once a block is removed or mined it becomes air. He cannot mine or move into lava. His goal is to mine the Diamond block.

#Easy Challenge:

Generate a 100x100x10 minecraft world. Once it generates you must act on it the laws defined above (sand and lava mostly)

Think of the world as x and y coordinates define the 2-D surface. Then you have a z coordinate to shift up or down a "plane". The top x-y surface of blocks will always be all air.
The block at 0, 0, 1 will always be dirt. Your Hero will start and occupy at 0, 0, 0.
The only diamond block in this world is at 99,99,9 . All the other blocks in the world will be randomly determined to be Air, Sand, Dirt or Lava.

#Navigation:

For the intermediate challenge you have to navigate the world. For the hero to move you can move to any air block. Again if they move to an air block if the block below it is not air they will move "down" automatically by "Falling" until they are above a non-air block. If the block they stand on is lava they die. They can be next to lava but never on top of lava as they will fall into the liquid and die.(Note for those who play mine craft we are making the liquid lava more simple. I realize Lava flows over blocks but we aren't going that complex)

Moving down is pretty easy. You just move your hero until they keep falling. The problem is going back up. Your hero can only "jump" if the blocks allow it. Example.

* D = safe block like dirt, sand or diamond to be on top of
* A = Air - nothing
* H = Hero occupying a block which is an air block

Imagine these blocks since the hero wants to move up to be on top of the blocks:

```
AAA
DAA
```

```
DDH
```

He has to move up. He can only move up by jumping up. Since the block above him is air and then the block above the block next to him has air above it and is next to the block he jumps up to he can safely move on top of that block to be as follows.

```
AAA
DHA
DDA
```

He can continue to jump and move over as he can jump up 1 block and over 1 safely always.

```
HAA
DAA
DDA
```

The problem is if he jumps up into an air block but the adjacent blocks to that block are not air over a safe block he cannot jump.

```
AAA
DAD
DHD
DDD
```

The above is a pit. The poor hero jumps up into the air block above him but the blocks next to that air block do not allow him to move.

Keep in mind I am showing you 2-D examples. Our world will be 3-D. If he can move up 1 block into air and any of the 4 blocks next to that adjacent not diagonal are air he can move safely into that air.

You cannot jump also if the block of air you will occupy is on top of a lava block (you die)

(Note in real mine craft your hero takes up 2 blocks height. We are making this more simple in that you will occupy 1 block)

Keeping all this in mine now you need to find the shortest and safest path to mine the diamond and get back. You start by occupying 0, 0, 0 which is air. Below it at 0, 0, 1 is a dirt block. So you are always safe. The diamond block is 99,99, 9 you want to move such that you can mine dirt or sand to create air to make a path to occupy 99,99,9 then you need to get back.

The key here is getting back. You cannot simply mine down You will create a "pit" that you cannot get back. If you cannot get a path to the diamond and back up to 0,0,0 you are unable to do the idea of the challenge of getting there and back. So when you make your path you will have to probably mine down and then mine over creating a "step" that allows you to "jump" back up to navigate safely.

##Lava and Sand Danger

Everytime you "mine" a sand or dirt block you make it an air block. You will have to check the case if sand or lava is above it. If Sand was above that block then the sand block will fall down until the block below is no longer air. If it was your goal to move into that space you cannot because you have to mine it again. Keep in mind there could be a chain reaction of sand. If you can a Sand Above a sand. The bottom sand drops down to an Air. It leaves behind an air and guess what that sand above that sand will drop down as well.

Lava drops down as well but it doesn't leave behind air, it flows (thus growing) If the hero mines the block above the lava will fall into his air spot and kill him. So don't do that.

If sand wants to fall on the air spot occupied by the hero it will kill him. So don't mine up if the block above that is sand (Note in minecraft you get pushed to an adjacent space so to keep it simple I am saying death but if you want to do a "push" here then go for it.)

#Intermediate level challenge:

Find the shortest and safe path down that lets you mine blocks to the diamond and then let you move back to the starting point following the above rules of jumping up and mining.

#Hard level challenge:

Generate a random map as always. The only difference from the easy generation is that the map will randomly place the 1 diamond block. The hero agent will seek out this diamond to get it. The hero also can only see blocks next to him. He will avoid moving down into air that has him falling more than 5 blocks in height (we didn't worry about this in intermediate as we had to leave a path back and that would mean he couldn't get back)
The hero will only remember or know about blocks he has been adjacent to. If for example he removes a dirt block above him. He does not know the block above that which is lava or sand and it kills him. However if he was adjacent to that block above the block he wants to mine he knows it will be sand or lava and he will not choose to mine it to seek out the diamond.

#Very Hard Challenge:

Do the hard challenge a path to the random diamond then find another path (or same path if safe) to get back to 0,0,0. If the first path was not always safe then the agent will try to navigate back to his starting spot if possible or until he dies.

#Questions:

This is a very long winded challenge. I will no doubt miss something. I hope you see the intent of the challenge and can address any missing element I did not cover. If you think it is important enough to bring up - go for it - share with all or ask and I will do what I can to answer. Sometimes it is hard to come up with air tight descriptions that cover ALL basis. In some cases the design of how to handle it is left to you to solve however you feel you want to solve it. Have patience with the challenge and see the intent. Thanks!

#FAQ:

* Failed maps seem to be common with pure random. If you wish to weight what is created to increase our hero's chances I would say go for it.

* No jump and removing blocks.
* No Placing blocks. We only remove.

Title: [2015-02-13] Challenge #201 [Hard] Mission Improbable
Text: # **(Hard)**: Mission Improbable

Imagine a scenario involving one event - let's call it event A. Now, this event can either happen, or it can not happen. This event could be getting heads on a coin flip, winning the lottery, you name it - as long as it has a 'true' state and a 'false' state, it's an event.

Now, the probability of event A happening, or the probability of event A not happening, is 100% - it must either happen or not happen, as there isn't any other choice! We can represent probabilities as a fraction of 1, so a probability of 100% is, well, 1. (A probability of 50% would be 0.5, 31% would be 0.31, etc.) This is an important observation to make - the sum of the probabilities of *all the possible outcomes* must be 1. The probability of getting a head on a fair coin flip is one half - 0.5. The probability of not getting a head (ie. getting a tail) is one half, 0.5, also. Hence, the sum of all the probabilities in the scenario is 0.5+0.5=1. This 'sum event' is called the sample space, or S.

We can represent this one-event scenario with a diagram, [like this](http://i.imgur.com/qwmIb6E.png). Each coloured blob is one outcome; all the outcomes are in S, and thus are all are in the big circle, representing S. The red blob represents the outcome of A *not* occurring, and the green blob represents the outcome of A occurring.

Now, let's [introduce some numbers](http://i.imgur.com/avK6iUQ.png). Let's say the probability of A occurring is 0.6 (60%). As A occurring, and A not occurring, are the only possible outcomes, then the probability of A not occurring must be 40%, or 0.4. This type of reasoning lets us solve basic problems, [like this one](http://i.imgur.com/buH6RQn.png). If the probability of A not occurring is 0.67, then what is the probability of A occurring? Well, the probability of S is 1, and so 0.67 plus our unknown must sum to 1 - therefore, the probability of A occurring is 1-0.67=**0.33**.

What about scenarios with more than one event? Look at [this diagram](http://i.imgur.com/xTxM2eV.png). This shows the sample space with two events, A and B. I've put coloured blobs for three of the four possible outcomes - of course, the fourth is in the empty region in A. Each region on the diagram is one possible outcome. Now, we come to something important. [This region on the diagram](http://i.imgur.com/xi0MNZ6.png) is **NOT** representing A - it is representing A **and not B**. [This region here](http://i.imgur.com/GJhGvs5.png) represents the probability of A as a whole - and, as you can see, the probability of A occurring is the probability of A and B, plus the probability of A and *not* B - in other words, the sum probability of all outcomes where A occurs.

Applying this additive rule lets us solve some more complex problems. [Here's a diagram representing Stan's journey to work this morning](http://i.imgur.com/eyQnbyk.png). Stan needs to catch two buses - the driver of the first bus is a grumpy old fellow and waits for hardly any time for Stan to get on; the driver of the second is much nicer, and waits for Stan where he can. Of course, if Stan misses the first bus, then it's likely that he will miss the second bus, too.

We know that, on 85% of days (0.85), Stan gets to work on time. We also said before that the driver of bus 2 is nice, and hence it's very rare to miss the second bus - the chance of getting on the first bus, and missing the second, is tiny - 1% (0.01). Stan tells us to work out how often he misses the first bus but not the second bus, given that he misses the second bus on 12% (0.12) of days.

Let's look at that last fact - the probability that Stan misses the second bus is 0.12. This means that the [sum of all probabilities in this region on the diagram must be 0.12](http://i.imgur.com/p9XM9uo.png). We already know that the probability of missing bus 2, but *not* missing bus 1 is 0.01. Hence, as there is only one other possible outcome involving missing bus 2, the probability of missing *both* buses must be 0.11, as 0.11+0.01=0.12! Thus our diagram now [looks like this](http://i.imgur.com/PqO8HI1.png). Now, out of the 4 possible outcomes in this scenario, we know three of them. We also know that the total sum of the probabilities of the four outcomes must equal one (the sample space); therefore, 0.85+0.01+0.11+**?**=1. This tells us that the probability of missing bus 1, but *not* missing bus 2 is 1-0.85-0.01-0.11=0.03. Therefore, we've solved Stan's problem. Your challenge today is, given a set of events and the probabilities of certain outcomes occurring, find the probability of an unknown outcome - or say if not enough information has been given.

# Input and Output

## Input Description

On the first line of input, you will be given a number **N**, and then the list of event names, like this:

```
3 A B
```

You will then be given **N** lines containing probabilities in this format:

```
A & !B: 0.03
```

Where the `&` indicates the left and right occur together, and the `!` indicates negation - ie. `A & !B` indicates that event A occurs and event B doesn't.

Finally, on the last line, you will be given an outcome for which to find the probability of, like this:

```
!A & !B
```

Thus, an input set describing Stan and his buses would look like this (where B1 is missing bus 1, B2 is missing bus 2):

```
3 B1 B2
!B1 & B2: 0.01
!B1 & !B2: 0.85
B2: 0.12
B1 & !B2
```

You may assume all probabilities are in increments of 1/100 - ie. 0.27, 0.9, 0.03, but not 0.33333333 or 0.0001.

## Output Description

Output the probability of the given unknown - in the example above,

```
0.03
```

# Example I/O

## Input

[(Represents this scenario as a diagram)](http://i.imgur.com/720vlok.png)

```
6 A B C
B: 0.7
C: 0.27
A & B & !C: 0
A & C & !B: 0
A & !B & !C: 0.13
!A & !B & !C: 0.1
B & C
```

## Output

```
0.2
```

## Input

```
3 B1 B2
!B1 & B2: 0.01
!B1 & !B2: 0.85
B2: 0.12
B1 & !B2
```

## Output

```
0.03
```

## Input

```
1 A B
A & B: 0.5
A
```

## Output

```
Not enough information.
```

# Addendum

Now might be the time to look into Prolog.

Title: [2015-02-06] Challenge #200 [Hard] Box in a Box
Text: #Description:

I have played around with this one a bit. I found it interesting. So let us imagine we can define a 3-D box of (height, width, depth) in dimensions. I then have a bunch of boxes I want to put in it. How do I figure out how get the most smallest boxes into the one big box?

Optimize the number. We don't want to use up as much space but get as many boxes as we can in 1 box.

Today's challenge is figuring out how to do this.

#Input:

You will be given the dimensions of the big box x, y, z. Then you will be given dimensions x, y, z of several smaller boxes below it.

Example:
the big box is 1st is 3x3x3 then we have to put all the boxes below it into it (yes 4,4,4 is bigger but someone in marketing really wants us to try...)

```
3 3 3

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

#Output:

```
Filled 8 boxes into the 3 3 3:
2 2 2
1 1 1
1 1 1
1 1 1
1 1 1
1 1 1
1 1 1
1 1 1
```

#Challenge Input:

```
10 10 10

4 4 4
5 5 1
4 5 1
7 7 7
5 5 5
3 3 3
5 5 5
9 8 7
4 5 1
```

```
    5 5 1
    4 4 4
    3 3 3
    4 4 4
```

Title: [2015-01-23] Challenge #198 [Hard] Words with Enemies -- The Game!
Text: #Description:

We had an easy challenge for part 1 of this challenge.

(http://www.reddit.com/r/dailyprogrammer/comments/2syz7y/20150119_challenge_198_easy_words_with_enemies/)

To expand this further we will make a game. For this challenge you will have to create a player vs AI game battling it out with words. Following some basic rules for the games you must design and implement this game.

#Rules of the Game:

* 5 Turns
* Each turn the user and AI are given random letters
* The user and AI must submit a dictionary checked word derived from these letters
* The words are compared. Using the easy challenge the winner of the duel is determined by whoever has the most left over letters.
* 1 point is awarded for each left over letter.
* At the end of 5 turns who ever gets the most points wins the game.

#Design:

There are many unanswered design issues with this game. I leave it as part of the challenge for you to develop and decide on that design. Please keep this in mind that part of the challenge beyond solving the coding aspect of this challenge is also solving the design issue of this challenge.

Some design suggestions to consider:

* How many random letters do you get each turn? How do you determine it?
* Do you wipe all letters clean between rounds and regenerate letters or do they carry over turn to turn with a way to generate new letters?
* Do you re-use letters left over for the next turn or just ignore them?
* Does the AI searching for a word have a random level of difficulty?

#AI design:

So you are giving your AI a bunch of letters. It has to find a legal word. Using a dictionary of words you can match up letters to form valid words.

Use this post to help find a dictionary to use that fits your needs
(http://www.reddit.com/r/dailyprogrammer/comments/2nluof/request_the_ultimate_wordlist/)

I really like the idea of a varied AI. You can make 1-3 levels of AI. Ultimately the AI can be coded to always find the biggest word possible. This could be rather difficult for a human to play against. I would suggest developing at least 2 or 3 different levels of AI (you might have to dumb down the AI) so that players can play against an easier AI and later play against the best AI if they want more a challenge.

#Checking the user input:

Users will input a word based on letters given. Your solution must check to make sure the word entered uses only the letters given to the human user but also that it makes a word in the dictionaries (see above)

#Input:

Varied as needed for the game to work

#Output:

Varied as needed for the game to work

#Example of a UI flow:

```
Welcome to Words with Enemies!
Select an AI difficulty:
1) easy
2) Hard
--> 1
You have selected Easy! - Let's begin!

Turn 1 -- Points You: 0 Computer: 0
----------------------------------------
Your letters: a b c d e k l m o p t u
Your word-> rekt
I am sorry but you cannot spell rekt with your letters. Try again.
Your letters: a b c d e k l m o p t u
Your word-> top
Valid Word! Open Fire!!!!
AI selects "potluck"

top vs potluck -- Computer wins.
You had 0 letters left over - you score 0 points
AI had 4 letters left over - AI score 4 points

Turn 2 -- Points You: 0 Computer: 4
----------------------------------------
Your letters: e i o k a l m q t u w y
```

Title: [2015-01-16] Challenge #197 [Hard] Crazy Professor
Text: #Description

He's at it again, the professor at the department of Computer Science has posed a question to all his students knowing that they can't brute-force it. He wants them all to think about the efficiency of their algorithms and how they could possibly reduce the execution time.

He posed the problem to his students and then smugly left the room in the mindset that none of his students would complete the task on time (maybe because the program would still be running!).

#The problem

What is the 1000000th number that is not divisble by any prime greater than 20?

#Acknowledgements

Thanks to /u/raluralu for this submission!

#NOTE

counting will start from 1. Meaning that the 1000000th number is the 1000000th number and not the 999999th number.

Title: [2015-01-10] Challenge #196 [Hard] Precedence Parsing
Text: # [](#HardIcon) **(Hard)**: Precedence Parsing

If you have covered algebra then you may have heard of the BEDMAS rule (also known as BIDMAS, PEMDAS and a lot of other acronyms.) The rule says that, when reading a mathematical expression, you are to evaluate in this order:

* **B**rackets, and their contents, should be evaluated first.

* **E**xponents should be evaluated next.

* **D**ivision and **M**ultiplication follow after that.

* **A**ddition and **S**ubtraction are evaluated last.

This disambiguates the evaluation of expressions. These BEDMAS rules are fairly arbitrary and are defined mostly by convention - they are called *precedence* rules, as they dictate which operators have precedence over other operators. For example, the above rules mean that an expression such as `3+7^2/4` is interpreted as `3+((7^2)/4)`, rather than `(3+7)^(2/4)` or any other such way.

For the purposes of this challenge, let's call the fully-bracketed expression the *disambiguated* expression - for example, `1+2*6-7^3*4` is disambiguated as `((1+(2*6))-((7^3)*4))`, giving no room for mistakes. Notice how *every* operation symbol has an associated pair of brackets around it, meaning it's impossible to get it wrong!

There is something that BEDMAS does *not* cover, and that is called *associativity*. Let's look at an expression like `1-2-3-4-5`. This contains only one operator, so our precedence rules don't help us a great deal. Is this to be read as `(1-(2-(3-(4-5))))` or `((((1-2)-3)-4)-5)`? The correct answer depends on the operator in question; in the case of the subtract operator, the correct answer is the latter. The left-most operation (`1-2`) is done first, followed by `-3`, `-4`, `-5`. This is called *left-associativity*, as the left-most operation is done first. However, for the exponent (`^`) operator, the right-most operation is usually done first. For example `2^6^9^10`. The first operation evaluated is `9^10`, followed by `6^`, `2^`. Therefore, this is disambiguated as `(2^(6^(9^10)))`. This is called *right-associativity*.

In this challenge, you won't be dealing with performing the actual calculations, but rather just the disambiguation of expressions into their fully-evaluated form. As a curve-ball, you won't necessarily be dealing with the usual operators `+`, `-`, ... either! You will be given a set of operators, their precedence and associativity rules, and an expression, and then you will disambiguate it. The expression will contain only integers, brackets, and the operations described in the input.

## Disclaimer

These parsing rules are a bit of a simplification. In real life, addition and subtraction have the same precedence, meaning that `1-2+3-4+5` is parsed as `((((1-2)+3)-4)+5)`, rather than `((1-(2+3))-(4+5))`. For the purpose of the challenge, you will not have to handle inputs with equal-precedence operators. Just bear this in mind, that you cannot represent PEMDAS using our challenge input, and you will be fine.

# Input and Output Description

## Input Description

You will input a number **N**. This is how many different operators there are in this expression. You will then input **N** further lines in the format:

```
    symbol:assoc
```

Where `symbol` is a single-character symbol like `^`, `#` or `@`, and `assoc` is either *left* or *right*, describing the associativity of the operator. The precedence of the operators is from highest to lowest in the order they are input. For example, the following input describes a subset of our BEDMAS rules above:

```
3
^:right
/:left
-:left
```

Finally after that you will input an expression containing integers, brackets (where brackets are treated as they normally are, taking precedence over everything else) and the operators described in the input.

## Output Description

You will output the fully disambiguated version if the input. For example, using our rules described above, `3+11/3-3^4-1` will be printed as:

```
(((3-(11/3))-(3^4))-1)
```

If you don't like this style, you could print it with (reverse-)Polish notation instead:

```
3 11 3 / - 3 4 ^ - 1 -
```

Or even as a parse-tree or something. The output format is up to you, as long as it shows the disambiguated order of operations clearly.

# Sample Inputs and Outputs

This input:

```
3
^:right
*:left
+:left
1+2*(3+4)^5+6+7*8
```

Should disambiguate to:

```
(((1+(2*((3+4)^5)))+6)+(7*8))
```

This input:

```
5
&:left
|:left
^:left
<:right
>:right
3|2&7<8<9^4|5
```

Should disambiguate to:

```
((3|(2&7))<(8<(9^(4|5))))
```

This input:

```
3
<:left
>:right
.:left
1<1<1<1<1.1>1>1>1>1
```

Should disambiguate to:

```
(((((1<1)<1)<1)<1).(1>(1>(1>(1>1)))))
```

This input:

```
2
*:left
+:left
1+1*(1+1*1)
```

Should disambiguate to:

```
(1+(1*(1+(1*1))))
```

Title: [2015-01-05] Challenge #196 [Practical Exercise] Ready... set... set!
Text: # [](#PEIcon) **(Practical Exercise)**: Ready... set... Set!

The [last practical exercise](/r/dailyprogrammer/comments/2nr6c4/20141129_challenge_190_practical_exercise_the/) was well-received so I'm going to make another one. This one is less complicated and, if you're still finding your feet with object-oriented programming, should be great practice for you. This should be doable in functional languages too.

The idea of a Set can be very math-y when you delve deeper but this post only skims the surface, so it shouldn't pose any issue!

## Background

A *set* is a mathematical concept that represents a collection of other objects. Those other objects can be numbers, words, operations or even sets themselves; for the (non-extension) purposes of the challenge they are integers only. A *finite set* is a set with only a finite number of items (unlike, for example, the set of all real numbers **R** which has uncountably infinite members.)

A set is generally represented with curly brackets with the items separated by commas. So, for example, the set containing `-3`, `6` and `11` could be written as `{-3, 6, 11}`. This notation is called an *extensional definition*.

There are some distinctions between a set and the list/array data structure:

* Repeated items are ignored, so `{-3, 6, 11}` is exactly the same as `{-3, -3, 6, 11}`. To understand why this is so, think less of a set being a container of items, but rather the items are members of a set - much like how you can't be a subscriber on /r/DailyProgrammer twice.

* Order doesn't matter - `{-3, 6, 11}` is the same as `{6, 11, -3}` and so on.

* Sets are generally seen as *immutable*, which means that rather than adding an item **A** to a set **S**, you normally create a new set with all the members of **S**, *and* **A**. Immutable data structures are quite a common concept so this will serve as an intro to them if you've not came across them already.

* A set can be empty - `{}`. This is called the empty set, weirdly enough.

Sets have 3 main operations.

* Union, with the symbol ∪. An item is a member of set **S**, where **S**=**A**∪**B**, if it's a member of set **A** or set **B**.
For example, let **A**=`{1, 4, 7}` and let **B**=`{-4, 7, 10}`. Then, **A**∪**B**=`{-4, 1, 4, 7, 10}`.

* Intersection, with the symbol ∩. An item is a member of set **S**, where **S**=**A**∩**B**, if it is a member of set **A** *and* set **B**.
For example, let **A**=`{1, 4, 7}` and let **B**=`{-4, 7, 10}`. Then, **A**∩**B**=`{7}`, as only 7 is a member of both sets.

* Complement, with the symbol '. An item is a member of set **S**, where **S**=**A**', if it's not a member of **A**.
For example, let **A**=`{1, 4, 7}`. Then, **A**' is *every integer* except 1, 4 and 7.

# Specification

You are to implement a class representing a set of integers.

* To hold its content, you can use an array, list, sequence or whatever fits the language best. Consider encapsulating this (making it `private`) if your language supports it.

* The class should expose a method `Contains`, which accepts an integer and returns whether the set contains that integer or not.

* The constructor of the class should accept a list/array of integers which are to be the content of the set. Remember to ignore duplicates and order. Consider making it a variadic constructor (variable number of arguments) if your language supports it.

* The class should have static methods for `Union` and `Intersection`, which both accept two sets and return another set containing the union or intersection of those two sets respectively. Remember, our sets are *immutable*, so create a new set rather tham modifying an existing one. Consider making these as binary operators (eg. `+` for union and `*` for intersection) if your language supports it.

* The class should have another static method for `Equals` or `equals`, which accepts two sets and returns a boolean value. This determines if the two sets contain the same items and nothing else.

Finally, the set should be convertible to a string in some way (eg. `ToString`, `toString`, `to_s` depending on the language) which shows all items in the set. It should show them in increasing order for readability.

If your language already has a class for sets, ignore it. The purpose of this exercise is to learn from implementing the class, not use the pre-existing class (although in most cases you would use the existing one.)

## 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 Ruby, you would not write a `ToString` method - you would write a `to_s` method, as that is the standard in Ruby. In C++ and C#, you would not necessarily write static `Union`, `Intersection` methods - you have the ability to overload operators, and you should do so if it produces [idiomatic code](http://en.wikipedia.org/wiki/Programming_idiom). The research for this is part of the task. 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.

## Extension 1 (Intermediate)

If you are feeling up to it, change your class for a set of integers and create a [generic](http://en.wikipedia.org/wiki/Generic_programming) set class (or, if your language has dynamic typing, a set of any comparable type.) Depending on your language you might need to specify that the objects can be equated - I know in C# this is by `IEquatable` but other language will differ. Some languages like Ruby don't even need to.

## Extension 2 (Hard)

This will require some thinking on your end. Add a `Complement` static method to your class, which accepts a set and returns another set containing everything *except* what's in the accepted set.
Of course, you can't have an array of every integer ever. You'll need to use another method to solve this extension, and adapt the rest of the class accordingly. Similarly, for the string conversion, you can't print an infinite number of items. For this reason, a set containing everything containing everything except 3 and 5 should print something like `{3, 5}'` (note the `'`). You could similarly use an overloaded operator for this - I've picked `!` in my solution.

## Addendum

Happy new year! I know /u/Coder_d00d has already wished you so, but now I do too. Have fun doing the challenge, help each other out and good luck for the new year.

Title: [2015-01-02] Challenge #195 [All] 2015 Prep Work
Text: #Description:

As we enter a new year it is a good time to get organized and be ready. One thing I have noticed as you use this subreddit and finish challenges you repeat lots of code in solutions. This is true in the area of reading in data.

One thing I have done is develop some standard code I use in reading and parsing data.

For today's challenge you will be doing some prep work for yourself.

#Tool Development

Develop a tool or several tools you can use in the coming year for completing challenges. The tool is up to you. It can be anything that you find you repeat in your code.

An example will be shown below that I use. But some basic ideas

* Read input from user
* Input from a file
* Output to user
* Output to a file

Do not limit yourself to these. Look at your previous code and find the pieces of code you repeat a lot and develop your own library for handling that part of your challenges. Having this for your use will make solutions easier to develop as you already have that code done.

#Example:

I tend to do a lot of work in C/objective C -- so I have this code I use a lot for getting input from the user and parsing it. It can be further developed and added on by me which I will.

(https://github.com/coderd00d/standard-objects)

#Solutions:

Can be your code/link to your github/posting of it -- Also can just be ideas of tools you or others can develop.

Title: [2014-12-12] Challenge #192 [Hard] Project: Web mining
Text: #Description:

So I was working on coming up with a specific challenge that had us some how using an API or custom code to mine information off a specific website and so forth.

I found myself spending lots of time researching the "design" for the challenge. You had to implement it. It occured to me that one of the biggest "challenges" in software and programming is coming up with a "design".

So for this challenge you will be given lots of room to do what you want. I will just give you a problem to solve. How and what you do depends on what you pick. This is more a project based challenge.

#Requirements

* You must get data from a website. Any data. Game websites. Wikipedia. Reddit. Twitter. Census or similar data.

* You read in this data and generate an analysis of it. For example maybe you get player statistics from a sport like Soccer, Baseball, whatever. And find the top players or top statistics. Or you find a trend like age of players over 5 years of how they perform better or worse.

* Display or show your results. Can be text. Can be graphical. If you need ideas - check out http://www.reddit.com/r/dataisbeautiful great examples of how people mine data for showing some cool relationships.

Title: [2014-12-5] Challenge #191 [Hard] Tricky Stick Stacking
Text: # [](#HardIcon) **(Hard)**: Tricky Stick Stacking

Similar to the previous [hard challenge with the arrows](/r/dailyprogrammer/comments/2m82yz/), this challenge will similarly require a hard degree of thought to solve (providing, of course, you develop the algorithm yourself,) while being relatively easy to understand.

Imagine you have a 2D plane, into which you can place sticks, [like so](http://i.imgur.com/mkt7n2D.png). All of the sticks are perfectly straight, and placed into this plane from the top (positive Y) down. The sticks will never overlap or cross over one another. Your task today is to simply determine in which order the sticks must be pulled out of the plane without hitting any other sticks.

There are a few rules for this:

* A stick is [pulled out of the plane *directly upward (ie. along the positive Y axis)*](http://i.imgur.com/8eFNtwh.png), [like so](http://i.imgur.com/MpquP7S.png). This means that you can't pull a stick out [sideward or at an angle](http://i.imgur.com/zGQL5xV.png) (for example, to avoid another stick.)

* A stick cannot be pulled out if there's [another stick directly above it](http://i.imgur.com/RWtPm05.png).

In some possible possible scenarios, there is only one possible order to pull the sticks out of the plane. [This scenario](http://i.imgur.com/16WBjSf.png) only has one possible order: 1, 2, 4, 3. [This scenario](http://i.imgur.com/gSkKVIg.png) however has two possible orders, as the last two remaining sticks are not interfering with one another's removal, so you can remove them in any order.

# Formal Inputs and Outputs

## Input Description

Each stick is described by a number and the co-ordinates of its 2 ends, like so:

```
n:x1,y1,x2,y2
```

Where the stick number **n** is between the points (x1, y1) and (x2, y2). You will first input a number **S** which is the number of sticks in the scenario. You will then take a further **S** lines of input in the above format. **n** must be an integer but the co-ordinates can be any real number.

## Output Description

You are to output one possible order of removal of the sticks (where each stick is identified by its number **n**. There may be more than one.

# Sample Inputs and Outputs

## Sample Input

[(Represents this scenario)](http://i.imgur.com/nDpDJag.png)

```
4
1:0,3,4,5
2:2,3,8,1
3:4,0,5,1
4:1,3,4.2,1
```

## Sample Output

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

## Sample Input

[(Represents this scenario)](http://i.imgur.com/gSkKVIg.png)

```
5
1:3,3,8,1
2:11,2,15,2
3:6,3,12,4
4:10,5,10,10
5:9,11,18,12
```

## Sample Output

This scenario has 2 possible outputs:

```
5, 4, 3, 1, 2
```

or:

```
5, 4, 3, 2, 1
```

## Sample Input

[(Represents this scenario)](http://i.imgur.com/l8X9Tgg.png)

```
6
1:1,6,12,6
2:1,7,1,15
3:11,1,13,10
4:14,10,15,6
5:15,2,15,5
6:12,1,14,11
```

## Sample Output

```
2, 1, 3, 6, 4, 5
```

## Sample Input

```
5
1:2,2,2,8
2:1,1,11,2
3:10,1,15,3
4:5,5,13,8
5:6,4,9,3
```

## Sample Output

(all 3 are valid)

```
1, 4, 5, 2, 3
4, 1, 5, 2, 3
4, 5, 1, 2, 3
```

## Sample Input

```
6
1:6,2,14,7
2:12,10,15,9
3:12,3,12,6
4:3,1,17,2
5:4,3,11,2
6:3,10,12,12
```

## Sample Output

(both are valid)

```
6, 2, 1, 3, 5, 4
6, 2, 1, 5, 3, 4
```

## Sample Input

```
5
1:2,1,15,15
2:15,5,15,12
3:10,8,13,2
4:13,4,15,4
5:8,9,12,13
```

## Sample Output

```
5, 1, 2, 4, 3
```

Title: [2014-11-29] Challenge #190 [Practical Exercise] The Complex Number
Text: # [](#PEIcon) **(Practical Exercise)**: The Complex Number

The Friday challenge was not able to be submitted, so I'm going to deviate from the Friday standard here and do a submission which will benefit a different group of Daily Programmers. The vast majority of problems here are for computer scientists, and I feel this leaves out the rest of you - ie. those who are here more for the programming practice than the logical puzzles. Therefore, rather than being expected to solve a logic problem, you will be expected to implement a piece of software from a required specification, thus serving as an exercise in good programming practice and making use of language features available to you.

In this exercise you will implement functionality for complex numbers. (If your language already supports such functionality, pretend it doesn't exist.) Please note that this challenge is an object-oriented one. I apologise now to people who prefer procedural or functional languages, and I will try to make such an exercise in the future. Before you do this, let me introduce you to what a complex number is.

## Background

The complex number system was created by mathematicians to more intuitively solve certain problems involving square roots. It has long been known that you cannot conventionally compute the square root of a negative number, as there is no number which, when multiplied by itself, will produce a negative number. If the original number is positive, the squared number will obviously also be positive. If the original number is negative, the squared number is also positive as multilplying two negative numbers together produces a positive number.

However, this meant that certain mathematical equations involving square roots had no solutions. This was quite an inconvenience for mathematicians at the time - it meant that certain polynomial equations could not be solved, as they ended up trying to work out the square root of a negative number. At some point, someone had the bright idea of ignoring the fact that you can't square root negatives. What if you pretended that the square root of -1 did exist? This is exactly what happened, and the value was defined as the *imaginary unit*, or *i* (imaginary as in the classical understanding of numbers, it doesn't actually exist). Therefore, *i*=√(-1). Using algebra this lets you square root other negative numbers as multiples of *i*, as √(ab) = √(a) * √(b).

* √(-4) = √4 * √(-1) = √4 * *i* = 2 * *i* = 2*i*

* √(-7) = √7 * √(-1) = √7 *i*

And so on. These numbers are called *imaginary numbers*. On their own they are useful, but they really come into their own when paired with normal numbera (aka *real* numbers, to distinguish them from *imaginary* numbers.) An example of a *complex* number would be 2+3*i* or 0.5-2.2*i*. These complex numbers are split into two bits, as you can see: the real component and the imaginary component. For example, given the complex number 3-7*i*, the real component is **3** and the imaginary component is -7*i*. Hence, a normal real number can be represented as a complex number with imaginary component 0*i*, like 2+0*i*.

Adding or subtracting two complex numbers is relatively simple. To do so, just add/subtract each component individually. For example, 1+3*i* add 3-2*i* equals 4+*i*. This requires no further explanation as there isn't much else to it.

Multiplying complex numbers is a bit more involved but still simple. Multiply the two complex numbers as you would an algebraic expression. For example, to multiply 1+3*i* and 3-2*i*, multiply each component together and add them all:

	1	3*i*
3	3	9*i*
-2i	-2*i*	-6*i*^2

Now, recall that *i*=√(-1). Hence, *i*^(2)=-1. Therefore, -6*i*^(2)=6. This means 1+3*i* multiplied by 3-2*i* equals 3+9*i*-2*i*+6, which is 9+7*i*.

To visualise it, you could plot these complex numbers on the number line. But wait... how would you do that? How can you represent the imaginary component on the number line without it floating somewhere above the line? In fact, that's essentially exactly what happens - an *Argand diagram* is used to do this. An Argand diagram representing the complex number 3-2*i* [looks like this](http://i.imgur.com/xycfwUk.gif). This diagram can be used to compute a value of a complex number called the **modulus**, which is, is essence, the 'distance from zero' on the diagram - ie. the length of the grey line, which can be computed with Pythagoras' theorem. In this case, the modulus is √(3^(2)+(-2)^(2)), which is √13.

Finally, there is another value of complex numbers, that is easy to work out. To work out the **complex conjugate** of a complex number, simply invert the sign of the imaginary component. For example, the complex conjugate of 3-2*i* is 3+2*i*. Simple.

# Specification

You are to implement a class representing a complex number.

* It is to be represented by floating-point number fields for the Real and Imaginary components.

* It is to expose a method `GetModulus` which returns a floating point number representing the modulus of the complex number.

* It is to expose a method `GetConjugate` which returns another Complex number representing the complex conjugate.

* It is to have 3 static/shared/classifier methods, each taking 2 parameters, for the 3 operations `Add`, `Subtract` and `Multiply`, each performing its respective operation and returning a Complex with the result of the operation.

* It is to expose a method `ToString` which converts the complex number to its string representation correctly: eg. `"3-2i"`, `"1-i"` or `"13"`.

The UML diagram for the Complex class [looks like this](http://i.imgur.com/PJYBCgd.png).

## Extension

If you are feeling up to it, implement these, too:

* It is to expose a method `GetArgument` which returns a floating point angle, in radians, representing the [argument of the complex number](http://en.wikipedia.org/wiki/Argument_%28complex_analysis%29).

* It is to have another static method taking 2 parameters for the operation `Divide`, which [divides two complex numbers](http://mathworld.wolfram.com/ComplexDivision.html).

The UML diagram for the extended Complex class [looks like this](http://i.imgur.com/z1ENG9F.png).

## 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 Ruby, you would not write a `ToString` method - you would write a `to_s` method, as that is the standard in Ruby. In C++ and C#, you would not write static `Add`, `Multiply` methods. You would instead overload the `+`, `-`, `*`, `/` operators, and rather than writing a `GetModulus` method you would write a `Modulus` property. Knowing and using these features that programming language provide is an important part of software development.

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.

Title: [2014-11-21] Challenge #189 [Hard] Write a Quine
Text: #Description:

A Quine is a very interesting little program that does only one thing: it prints out exactly its own source code. Quines are tricky to write, but figuring out how to do it is a very rewarding and fun little challenge.
Some rules for this challenge:

* The program can use no I/O except for printing out to standard output. It can't read (or write) anything from standard input, or any file (or network socket, or whatever). That is to say, you can't make a program that simply reads the source code and prints it out.

* The output of the program and the source code for the program have to match exactly, literally byte for byte (including newlines and comments, if you include any). If you're on a unix system, you can check for this by using the diff utility.

* The source code of your Quine has to be longer than 1 character. The reason for this is to prevent "degenerate" Quines, like having an empty program that prints out nothing.

* Often people compete about who can write the shortest Quine in a given programming language. Don't worry about that for this challenge, make your Quines as long as you want.

There are many websites that describe in detail exactly how to write a Quine, but you are encouraged not to look those up. Figuring out how to do it for yourself is very rewarding. However, if you're hopelessly stuck, you can go ahead and research it. Wikipedia provides a very good description of how to do it.

#Input:

None for this challenge.

#Output:

The source code of your program exactly, byte for byte.

#Bonus:

Write a two-language Quine. That is, write a program in language A that prints out code for language B, and when you run the code for language B, it prints out the original code for language A.

That is, if your two languages are python and ruby, you should be able to run this:

```
    $ python A.py > B.rb
    $ ruby B.rb > C.py
    $ diff A.py C.py
    $
```

That is, when running A.py in python, it produces the ruby source code B.rb, and when you run B.rb in ruby, it produces C.py, and A.py and C.py are exactly the same.

Title: [2014-11-14] Challenge #188 [Hard] Arrows and Arrows, part 1
Text: # [](#HardIcon) **(Hard)**: Arrows and Arrows, part 1

Wednesday's challenge was released later than I wanted it to be (my fault entirely), so I'll make it up to you by posting this one early. I fear some previous hard challenges have appeared unapproachable to some people due to their logical or mathematical complexity. I aim to make a Hard challenge today which is innately simple, but will still require a Hard degree of thought (assuming you come up with the algorithm yourself.)
Take this grid of characters:

```
    v<^><>>v><>^<>vvv^^>
    >^<>^<<v<>>^v^v><^<<
    v^^>>>>>><v^^<^vvv>v
    ^^><v<^^<^<^^>>>v>v>
    ^<>vv^><>^<^^<<^^><v
    ^vv<<<><>>>>^<>^^^v^
    ^<^^<^>v<v^<>vv<^v<>
    v<>^vv<^>vv>v><v^>^^
    >v<v><^><<v>^^>>^<>^
    ^v<>^<>^>^^^vv^v>>^<
    v>v^^<>><<<^^><^vvv^
```

Let's imagine they all represent arrows, pointing to a cell next to them. For example, `v` points downward, and `<` points left. Let's also imagine the grid is infinite - ie. a `>` arrow at the right-hand side will 'wrap around' and point to the leftmost character on the same row, meaning the board has no limits. Now, we're going to follow the direction of the arrows. Look at the top-left cell. It's a `v`, so it points down to the cell below it, which is a `>`. That points to the cell to its right, which is a `^`. This points up to the cell above it, which is a `<`. This points to the cell to its left... which is exactly where we started. See how this has formed a 'loop'? You could go round and round and round forever. Remember, the board wraps around, so this grid is also a loop:

```
 >>>>>>>>>>>>
```

And so is this, if you follow the arrows:

```
    ^^>
    >^^
    ^>^
```

This looping structure is called a **cycle**. The discrete mathematicians in this sub should have all collectively just said *'aha!'*, as they should know already be thinking of how to approach the challenge from that last sentence. If you're not a discrete mathematician, read on. Your challenge today is simply described: given a grid such as the one above, find the *largest cycle* in it.

One important point: the 'length' of the cycle is just the part of the cycle that repeats. For example, the cycle is *not* made longer by adding an 'intro' to it:

```
>>v
^<<
 ^
 ^
 ^
 ^
```

The length of this cycle is 6 regardless of where you start from, as that is the length of the 'cycle'.

# Formal Inputs and Outputs

## Input Description

You will input 2 numbers first - these are the width and height of the grid you'll be working with. Then you will input a grid in the same format as described above.

## Output Description

You are to output the length of the longest cycle on the grid, possibly along with some representation of where that cycle is on the board (eg. print the cycle in another color.)

# Sample Inputs and Outputs

## Sample Input

This input should test the ability of your program to find longer cycles over shorter cycles, and ignore arrows not in a cycle.

```
5 5
>>>>v
^v<<v
^vv^v
^>>v<
^<<<^
```

## Sample Output

Longest cycle: `16`
Position:

```
>>>>v
^   v
^   v
^  v<
^<<<
```

## Sample Input

This should test the ability of your program to find cycles that wrap around.

```
45 20
^^v>>v^>>v<<<v>v<>>>>>>>>^vvv^^vvvv<v^^><^^v>
```

```
>><<>vv<><<<^><^<^v^^<vv>>^v<v^vv^^v<><^>><v<
vv<^v<v<v<vvv>v<v<vv<^<v<<<<<<<<^<><>^><^v>>>
<v<v^^<v<>v<>v<v<^v^>^<^<<v>^v><^v^>>^^^<><^v
^>>>^v^v^<>>vvv>v^^<^<<<><>v>>^v<^^<>v>>v<v>^
^^^<<^<^>>^v>>>>><>>^v<^^^<^^v^v<^<v^><<^<<<>
v<>v^vv^v<><^>v^vv>^^v^<>v^^^>^>vv<^<<v^<<>^v
<<<<<^<vv<^><>^^>>>^^^^<^<^v^><^v^v>^vvv>^v^^
<<v^<v<<^^v<>v>v^<<<<<>^^v<v^>>>v^><v^v<v^^^<
^^>>^<vv<vv<>v^<^<^^><><^vvvv<<v<^<<^>^>vv^<v
^^v^>>^>^<vv^^<>>^^v>v>>v>>v^vv<vv^>><>>v<<>>
^v<^v<v>^^<>>^>^>^^v>v<<<<<>><><^v<^^v><v>^<<
v>v<><^v<<^^<^>v>^><^><v^><v^^^>><^^<^vv^^^>^
v><>^><vv^v^^>><>^<^v<^><v>^v^<^<>>^<^vv<v>^v
><^<v>>v>^<<^>^<^^>v^^v<>>v><<>v<<^><<>^>^v<v
>vv>^>^v><^^<v^>^>v<^v><>vv>v<^><<<<v^<^vv<>v
<><<^^>>^<>vv><^^<vv<<^v^v^<^^^^vv<<>^<vvv^vv
>v<<v^><v<^^><^v^<<<>^<<vvvv^^^v<<v>vv>^>>^<>
^^^^<^<>^^vvv>v^<<>><^<<v>^<<v>>><>>><<^^>vv>
<^<^<>vvv^v><<<vvv<>>>>^<<<^vvv>^<<<^vv>v^><^
```

## Sample Output

Longest cycle: `44`
Position:

```
              >>>>>^
              ^<
               ^
              >^
              ^
   >^
   ^
>>>^
^
^<
 ^
 ^
 ^
>^
^
^
^  v<<
^<<< ^
               ^<<
                 ^<<
```

# Notes

If you're a discrete mathematician or know of graph theory, you could try treating the grid as a directed graph and use a cycle finding algorithm on it. If not, try and come up with your own algorithm. [I wrote a tool for you to generate random inputs](http://jsfiddle.net/Quackmatic/s976w08c/2/). If you find (or make) a cool loop in an input, post it here!

# Bonus

Notice how the path length will always be an even number if the arrows do not wrap around? Try to explain why. Food for thought.

Title: [11/05/2014] Challenge #187 [Hard] Lumberjack Floating Log Problem
Text: #Description:

Our lumberjacks have been busy lately. Before winter the lumberjacks must get the logs to the lumber mill. Our lumberjacks use a local river system to float logs down river to the lumber mill.

One of our lumberjacks was a former software engineer who gave up his keyboard and mouse for an axe. He has suggested to the lumberjack foreman that using a program he can solve a problem they been having.

They want to find out how many logs can float in the river without causing a pile up. If you put too many logs in the river they will get stuck. However if you put just enough in and help them float down paths in the complex river they can optimize how many logs can be sent to the lumbermill.

Your challenge is to solve two problems.

* How many logs can be sent down the river system to maximize the use of the river without causing a pile up.

* The routes must be optimal and the shortest path possible given the logs already sent on the river. Show the optimal path.

#River:

The river is directed from a source down into a large pond by the lumbermill. There are many routes to take. Each route can support so many "log routes". Think of a log route as a route a log takes down the stream. For this log to reach the pond it takes away capacity from the route to hold logs. Once a part of a route has enough logs passing through it - it can no longer support more logs.

The following directed river gives you "nodes". The direction matters as you can only go in 1 direction. And the number represents how many "log paths" can travel over that segment of river before new log routes can no longer route on that segment (they have to find another segment that is not at full capacity)

A is our Start. All logs enter the river at point A.

* A->B - holds 6 log paths
* A->C - holds 2 log paths
* B->E - holds 3 log paths
* B->D - holds 3 log paths
* D->C - holds 2 log paths
* D->F - holds 1 log path
* C->G - holds 5 log paths
* E->H - holds 1 log paths
* E->I - holds 2 log paths
* F->H - holds 1 log path
* G->H - holds 2 log paths
* G->I - holds 2 log paths
* H->I - holds 4 log paths

I is the lumber mill pond.

So log routes will go from A to I. You want the shortest path to route logs. But as routes get used eventually they hit segment limits and you will have to find a new route to take for the next log to find the shortest path.

#Log Paths

So an optimal path on our river would be A->B->E->I -- 4 segments. However each of those segments will now have 1 less log that can go on it. When we send another log we might A->B->E->I again for the next log. But the third log will not be able to take this path because the E->I segment has 2 logs going on that path so the problem must find another path as the E->I segment is now maxed on what logs can go on it.

#Output:

Send a log and show the optimal path. Your output will show the log # (the first, 2nd, 3rd log sent down the river) and the shortest path on the river it can take (given all the previous log routes being used)

Eventually hit a point where no new log can be sent because the river cannot handle it. Anymore logs will cause a pile up. At this point we will know how many logs can our river handle.

So your output should show as an example

Log #1 takes A->B->E->I - path of 4

Log #2 takes A->B->E->I - path of 4

Log #3 takes A->C->G->I - path of 4

...

Log #n takes (path) - path of (size of path)

River is now full. Can send n logs.

#Spoiler Warning

This challenge is key to keep your solutions under spoiler protection. Not just your code but any verbal text talking about how you solve it. So if you wish to use "text" to say like "Oh well I solve this by...." please spoiler that or your solution will be removed. Thank you.

Title: [10/17/14] Challenge #184 [Hard] Classification Algorithms and intro to Machine Learning
Text: Hi everyone! This challenge is gonna be a special one. There will be no challenge! Well there will be research! Not very interesting huh ? Lets try shall we ? I just want to spread some awareness of a versatile concept in computer science.

Today we will be learning about classification algorithms. Continuing our previous Challenge #183[Hard] if you completed, you should know the whole point of dimensionality reduction done in the challenge was basically to **Reduce the data**. Why reduce ? Thats mainly for later use when we can classify the data to its separate classes.

Lets try to understand classification. Consider it in this way. Say you have sets of documents. Some are history documents, Some biology, Some psychology and so on. The web is full of documents. Now if you want to design a search of what does every document classify to then we can consider the classes to be history, biology, psychology and so on.

Now you will ask how are the documents fed to the algorithm basically. There are multiple ways but the main way is normally that a document can be considered as a collection of words. Every word here is an attribute or a property of the document. And if you consider the attribute to be the column names of the matrix then you can consider the number of times the word occurs to be the value of the word in the document.

So here is an example dataset.

a1	a2	a3	a4	a5	a6	a7	a8	
1	1	0	0	0	0	0	0	psychology
0	0	1	1	1	0	0	0	biology
0	0	0	0	0	1	1	1	chemistry

from the above small set you can see that the documents which have the attribute a1 and a2 can be classified into psychology, a3, a4 and a5 into biology and a7 and a8 into chemistry. It can be clearly seen here.

Now thats just a small data set. When it comes to building classification for huge data sets a lot of constraints and complexities arise based on the problem. for example, some attributes can be important for some type of classification. Like when considering finding out the best player in Counter Strike, I would consider a player with a higher sniper strength to have a higher importance and such.

There are multiple classification algorithms. For example Naive bayes, K means etc. This basic classification can be considered as a small step in Machine Learning. Classification algorithms are basically divided into two types. Supervised and Unsupervised. Supervised basically means that the attributes are labeled and unsupervised is the type of learning where the data is unlabelled which means we have to deduce its importance on our own using the different algorithms.

I dont want to give a challenge today since i feel a lot of people wont be able to do it and its not feasible to do it in one week. Machine Learning basically means that your algorithm *learns* the data and you predict classes. Its a difficult concept but very interesting. All top companies use it. You can see from google to facebook to uber to intel, every face recognition, search, pattern recognition software use it and is a growing industry concept and i wanted to bring some awareness.

#So yes the challenge..

Do some research on Machine leaning and classification algorithms and do discuss! (Remember Google is your best friend! :D)

Title: [10/10/2014] Challenge #183 [Hard] Dimensionality Reduction
Text: # [](#HardIcon) _(Hard)_: Dimensionality Reduction

I have submitted in such a long time so i though i give a hard challenge! This week's and next week's hard challenge will be a machine learning/data mining challenge which are in quite high demand and have applications in today's top companies like facebook, google, quora, twitter and hundreds of multiple other companies. It will be a long challenge so do note that there will be another hard challenge next week which will be the continuation to this one.

This challenge consists of three parts and we will be doing two parts this week.

## Problem Description

Part 1:

Do read the note below part 1 before proceeding.

* Create a sparse matrix with a large number of dimension like 1000 rows and 120,000 columns with different values in it.

* Since some people might have memory problems its alright if you reduce the number of columns to say 12000 or 1200 or even lesser if you feel necessary. That would be fine too for learning purposes.

* Create a list of labels for the corresponding sparse matrix with the same number of rows and have a fixed number for the type of labels such as 20 or 25. Again i give you the freedom to reduce the number of labels if necessary. The point of the challenge is to learn the idea of dimensionality reduction.

* Create a testing set which is a smaller sparse matrix with corresponding labels

____

Note: In case you want to play with real data do make it a point to visit these pages

* http://www.quora.com/Where-can-I-find-large-datasets-open-to-the-public

* http://stackoverflow.com/questions/381806/large-public-datasets

For public available datasets over which you can do part 2. You can skip part 1 if you use the public datasets ;)

____

Part 2:

Input:

1. Training input which is a Random Sparse matrix of large number of rows and columns say 1000 x 120000 matrix from the **part 1**.

2. Classification label for each row in the training input **part 1**.

* Perform dimensionality reduction using algorithms like Principal Component Analysis

Do note you can use any language necessary. I would suggest matlab to be honest since it will make your work easier ;)

## Some helpful Links

* what is a sparse matrix ?
http://en.wikipedia.org/wiki/Sparse_matrix

* what is supervised learning ?
http://en.wikipedia.org/wiki/Supervised_learning

* What is dimensionality reduction ?
http://en.wikipedia.org/wiki/Dimensionality_reduction

* Some info on testing set, training set..
http://stats.stackexchange.com/questions/19048/what-is-the-difference-between-test-set-and-validation-set

* What is k-fold cross validation ?
http://en.wikipedia.org/wiki/Cross-validation_(statistics)#k-fold_cross-validation

Title: [10/03/2014] Challenge #182 [Hard] Unique Digits
Text: #Description:

An interesting problem to solve:

Looking at the Base 10 number system it has the digits 0 1 2 3 4 5 6 7 8 9

If I were given the digits 5 7 and 9 how many unique numbers could be formed that
would use all these digits once?

For example some easy ones would be:

579
975
795

And so on. but also these would work as well.

111579
1120759

These could go on forever as you just add digits. There would be many numbers just padding numbers to the unique numbers.

Some might think that these next three might be valid but they are not because they do not contain all 3 digits:

57
75
95

So to cap off the range let us say numbers that do not go beyond 7 digits (so 7 places in your numbers)

I am also interested in other base number systems. Like how many unique numbers using 5 6 could I find in base 8 (octal) or A E 0 1 in a base 16 (hexidecimal) ?

Your challenge is to be able to take 2 sets of inputs and find out how many unique digits up to 7 places can be found given those 2 inputs.

#Input:

<Base system> <digits>

* Base system is a base counting system. This number can be between 2 to 16.
* Digits will be a list of digits that are ALL shown only once in the number

#Output:

All the unique numbers given up to 7 digits long only using the digits given once. followed by their base 10 value. At the bottom of the listing a "count" of how many numbers you found.

So say I was looking for base 2 and my unique digits were 1 I would see this:

```
    1 - 1
    10 - 2
    100 - 4
    1000 - 8
    10000 - 16
    100000 - 32
```

```
1000000 - 64
Count: 7
```

#challenge inputs:
These are several pairings to run. For the sake of size do not list your outputs - Maybe just the "counts" you found.
If you wish to share the outputs use like a gist or link the output for people to go look at.

```
2 1
8 3 5 6
10 1 3 9
16 A E 1 0
```

#challenge input to try:

For all base systems 2 to 16 find the numbers 0 1 in them.

#challenge difficulty

This is an unknown. Not sure if easy, intermediate or hard. Regardless lets give it a try. Could be very easy. Could be very hard.

Title: [26/09/2014] Challenge #181 [Hard] Deconstructing Audio
Text: #Description

You're part of an innovative new company whose primary goal is to improve the music catalogue and its databases for integration with Apple,Linux and Microsoft products. You notice a significant lack of metadata given by users and wonder if there's a way to automate the process instead.

#Formal Inputs & Outputs

Given an audio file that contains music (this won't work on speech or anything irregular) you must create a program that can determine the [BPM/Tempo](http://en.wikipedia.org/wiki/Tempo) of that audio file.

##Input description

On input you should pass your file through for analysis.

##Output description

The program should output the Beats per minute of a song

For example

```
120bpm
```

or

```
79bpm
```

[Here](http://songbpm.com/) is a good website to test your results against

#Notes/Hints

For the less musically inclined, make sure your music is in 4/4(common time) before analyzing. Analyzing odd time signatured songs might make this significantly harder. This brings us neatly to the bonus challenge...

There are a few ways to go about this challenge from the exceedingly simple; Pulling the data from an already existing database. Or the actual way, using various signal processing techniques to arrive at an accurate result.

Here is a good article on beat detection and implementing the algorithm

http://archive.gamedev.net/archive/reference/programming/features/beatdetection/index.html

You may also want to check out [Comb filtering](http://en.wikipedia.org/wiki/Comb_filter)

#Bonus

Output the time signature of the song

Title: [9/19/2014] Challenge #180 [Hard] Sorting Visualisation
Text: # [](#HardIcon) _(Hard)_: Sorting Visualisation

This challenge is up a bit early as I'm busy tomorrow so I'll probably forget. Anyway, after reading the comments on [this week's Weekly Discussion](http://www.reddit.com/r/dailyprogrammer/comments/2ggunp/), I wrote this week's Hard challenge based on two commonly requested things:

* Graphical visualization

* Usage of algorithms

and I decided to combine the two. This will also be a relatively open-ended challenge, as they seem to be quite popular among the developers - i.e. you - here. For this challenge, you will input a set of real numbers, and visualise the sorting of that set into ascending order, with an algorithm(s) of your choice, with any mode of visualisation you can imagine.

## Input Description

You will be given a set of numbers that are between 0 and 1 (inclusive). The method of input is up to you.

## Output Description

Visualise the sorting of the data, in a step-by-step manner, in any way you like. It can be console-based, graphical based, web-based, 3D based or even physically with an Arduino or the like, if you're feeling particularly adventurous!

# Further Reading

To get to grips with some different sorting algorithms, let's look at four here.

## Bubble Sort

Bubble sort is the simplest of the four. You simply step through the list, looking at pairs of elements that are next to each other. If the pair is not in order, you swap them and look at the next pair, like so:

```
4 1 2 3 5
<->

1 4 2 3 5
  <->

1 2 4 3 5
    <->
```

```
        1 2 3 4 5
          x

        1 2 3 4 5
```

If the list is not sorted after doing this, you go through the list again until it is. Done! This is simple but slow. Onto the next one...

## Selection Sort

Selection sort is, to me, the most intuitive of the four, and is probably similar to what you do when you sort a pack of playing cards. Simply start with your list L and an empty list S. While L is not empty, move the lowest value from L to the end of S, like so:

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

        [3 5 6 8 7 2 4] [1]

        [3 5 6 8 7 4] [1 2]

        [5 6 8 7 4] [1 2 3]

        [5 6 8 7] [1 2 3 4]

        [6 8 7] [1 2 3 4 5]

        [8 7] [1 2 3 4 5 6]

        [8] [1 2 3 4 5 6 7]

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

And now S is our sorted list. Simple again, however this too is slow on larger lists.

## Merge Sort

Fast and surprisingly simple, once you get your head round it. First, split the list into lists with only 1 item:

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

Then, take pairs of lists and merge them. How to merge them, you say? It's fairly simple - to merge lists A and B into new list C, do the following. While A and B are not empty, look at the first item in both lists. Append the lowest of the two to the end of list C. If either A or B is empty and the other isn't, just put the remaining items from the non-empty list at the end of C. OK.

Now, we have the following lists after merging 3 times:

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

        [1 3 5 6] [2 4 7 8]

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

The final list there is your list in order. Done!

## Quicksort

This one is perhaps the most difficult of the four, but it's still not too hard. The first step is to take the list - let's call it L - and partition it into two halves, with a 'pivot' value in the middle. A good way to choose the pivot is to pick 3 random values from L and choose the median. Anyway, after we've split the list in half - into two lists, A and B - we look at the elements in A, which we will make our lower list, and compare each value against the pivot value. If the element is greater than the pivot value, put it at into list B (our higher list), in no particular position. Now, do the same for list B; look at each element and see if it is lower than the pivot. If it is, put it into list A at no particular position. Our list L now looks like:

```
    A pivot B
```

now sort A and B the same way we sorted L. If A or B contain either no elements or one element, it is already sorted, and if there are only 2 values, you can just swap them.

# Stuck?

Here are a few videos to kick-start your imagination!

* [15 Sorting Algorithms in 6 Minutes](https://www.youtube.com/watch?v=kPRA0W1kECg)
* [Bubble Sort folk dance](https://www.youtube.com/watch?v=lyZQPjUT5B4)
* [Quick Sort visualized](https://www.youtube.com/watch?v=8hEyhs3OV1w)

Title: [9/12/2014] Challenge #179 [Hard] Traveller Game Part 2 (Torchlight)
Text: #Description:

For today's challenge you must do the [Intermediate Traveller Game](http://www.reddit.com/r/dailyprogrammer/comments/2g1c80/9102014_challenge_179_intermediate_roguelike_the/) challenge from wednesday. If you have already done it then you have a head start.

We will modify our Traveller game by adding Torch light. Seeing the whole map is too easy. If you are limited in what you can see then you have a tougher time planning your moves.

You will modify your game the following ways.

* Add Torch view You only see 3 spaces away from your hero
* Add 5 Random Wall barriers -- These are 3 walls in a row either vertical or horizontal. Or have a fixed map with hallways/wallls. Your choice.
* Continue to generate random gold/goal spots for scoring.
* Same Map size as the itnermediate.

#Examples:

Here are 3 examples of how the torchlight should work.

```
    Full Sight
                %%%%%%%%%%
                %..$.....%
                %......$.%
                %...@....%
                %....$...%
                %.$......%
                %%%%%%%%%%

                Torch Level 3
                  %
                 $..
```

```
 .....
...@...
 ...$.
  ...
   %
```

Full Sight (corner case)

```
%%%%%%%%%%
%@.$.....%
%......$.%
%........%
%....$...%
%.$......%
%%%%%%%%%%
```

Torch Level 3

```
%%%%
%@.$.
%...
%..
.
```

Full Sight (Barrier case)

```
%%%%%%%%%%
%..$.....%
%.%%...$.%
%...@....%
%.%%%%%%.%
%.$......%
%%%%%%%%%%
```

Torch Level 3

```
   %
   ..
 %%...
...@...
 %%%%%
```

#Harder:

Torches have a power of 5 instead of 3 -- every 2 Steps the Torch degenerates in power to 4 then 3 then 2 then 1 then none. In the room you will random place other "T" for torches or a light source which will refresh your torch power by +2 up to a max of 10. Again your Torch view will degenerate by 1 every 2 steps used (so if you can gain more than 5 torch power up to 10 but then it will degenerate 10-9-8 etc)

You will add 10 random pit traps. If the hero ends in the pit trap they die and game is over.

Title: [9/05/2014] Challenge #178 [Hard] Regular Expression Fractals
Text: #Description:
For today's challenge you will be generating fractal images from regular expressions. This album describes visually how it works:

* [https://imgur.com/a/QWMGi]

For the challenge you don't need to worry about color, just inclusion in the set selected by the regular expression. Also, don't implicitly wrap the regexp in \^...$. This removes the need to use .* all the time.

#Input:

On standard input you will receive two lines. The first line is an integer n that defines the size of the output image (nxn). This number will be a power of 2 (8, 16, 32, 64, 128, etc.).
The second line will be a regular expression with literals limited to the digits 1-4. That means you don't need to worry about whitespace.

#Output:

Output a binary image of the regexp fractal according to the specification. You could print this out in the terminal with characters or you could produce an image file. Be creative! Feel free to share your outputs along with your submission.

#Example Input & Output:

##Input Example 1:

```
256
[13][24][^1][^2][^3][^4]
```

##Output Example 1:

* [http://i.imgur.com/zhSr365.png]

## Input Example 2 (Bracktracing) :

```
256
(.)\1..\1
```

## Output Example 2:

* [http://i.imgur.com/iLu7Pq4.png]

# Extra Challenge:

Add color based on the length of each capture group.

Title: [8/29/2014] Challenge #177 [Hard] SCRIPT it Language
Text: #Description:

We all enjoy strings. We all enjoy breaking up texts. Time to go bigger than just a few sentences.

Out of curiosity we will be breaking down a movie script. The movie I have picked is Monty Python and the Holy Grail.

So what do you mean by breaking it down? Our challenge is to crunch some numbers on this movie and figure out some fun statistics.

You will first go get the text of this script off the web. Part of the challenge is how to deal with this.

I really like this [Monty Python and the Holy Grail Script] (http://www.sacred-texts.com/neu/mphg/mphg.htm) script of the movie.

#By Scene:

* By Scene (From 1 to 36 in order) - how many words are spoken. (Anything between [] and () are not spoken words)
* Top 3 Spoken Words (and how many times they were used) and percentage of all the words spoken in that scene.
* List of all characters in the scene and next to them How many "Lines" and "Words" they used.
* The list of characters in scene should be sorted based on count of "Words" used from high to low in count.

* A "Line" is any sentence that ends with your typical end of sentence punctuation.
* Anything in [] or () we will call a "stage direction" Just count how many directions are given. Note: Words in a stage direction do not count towards words spoken or used in script.

#By Whole Movie:

At the end of the crunch we want this data.

* Number of Lines
* Number of Words
* Number of Stage Directions
* Number of characters
* Sorted by most words the list of all Characters and how many Words and Lines they each got - Please also add a percentage of total. So if a character spoke 100/1000 lines they will have Lines 100 (10%)
* Top 10 Words sorted in Order from Most to least (Ties count as 1 Spot so if the top 2 words are "The" and "A" then it should be like 1) "The" "A"
* Top 3 Scenes with the most Words spoken (Again if ties - both are listed as 1 spot)

* In the movie there are a bunch of characters known as the Knights of Ni. They cannot say the word "it" (forbidden) - Count how many times this forbidden word is used and list a count of "Forbidden Word of the Knights of Ni"

#Output:

Given the above you will have to format and display the data. I leave the design up to you. But it should be easy to read and understand.

#Extra Challenge:

Find a way to show this data more meaningful than just list of hard data. Develop a Histogram or format the data into a format that makes a cool looking pie chart/table/graph.

Title: [8/20/2014] Challenge #176 [Hard] Spreadsheet Developer pt. 2: Mathematical Operations
Text: # [](#EasyIcon) _(Hard)_: Spreadsheet Developer pt. 2: Mathematical Operations

Today we are building on [what we did on Monday](/r/dailyprogrammer/comments/2dvc81/8182014_challenge_176_easy_spreadsheet_developer/). We be using the selection system we developed last time and create a way of using it to manipulate numerical data in a spreadsheet.

The spreadsheet should ideally be able to expand dynamically in either direction but don't worry about that too much. We will be able to perform 4 types of operation on the spreadsheet.

* Assignment. This allows setting any number of cells to one value or cell. For example, `A3:A4&A5=5.23` or `F7:G11~A2=A1`.

* Infix operators - `+`, `-`, `*`, `/` and `^` (exponent). These allow setting any number of cells to the result of a mathematical operation (only one - no compound operations are required but you can add them if you're up to it!) For example, `F2&F4=2*5` or `A1:C3=2^D5`. If you want, add support for mathematical constants such as *e* (2.71828183) or *pi* (3.14159265).

* Functions. These allow setting any number of cells to the result of a function which takes a variable number of cells. Your program must support the functions `sum` (adds the value of all the given cells), `product` (multiplies the value of all the given cells) and `average` (calculates the mean average of all the given cells). This looks like `A1:C3=average(D1:D20)`.

* Print. This changes nothing but prints the value of the given cell to the screen. This should only take 1 cell (if you can think of a way to format and print multiple cells, go ahead.) This looks like `A3`, and would print the number in A3 to the screen.

All of the cells on the left-hand side are set to the same value. Cell values default to 0. The cell's contents are not to be evaluated immediately but rather when they are needed, so you could do this:

```
    A1=5
    A2=A1*2
    A2 >>prints 10
    A1=7
    A2 >>prints 14
```

After you've done all this, give yourself a whopping big pat on the back, go [here](/r/IAmA/comments/227tme/) and apply to work on the Excel team - you're pretty much there!

# Formal Inputs and Outputs

## Input Description

You will be given commands as described above, one on each line.

## Output Description

Whenever the user requests the value of a cell, print it.

# Example Inputs and Outputs

## Example Input

```
    A1=3
    A2=A1*3
    A3=A2^2
    A4=average(A1:A3)
    A4
```

## Example Output

```
    31
```

Title: [8/15/2014] Challenge #175 [Hard] Hall of Mirror[]
Text: # [](#HardIcon) _(Hard)_: Hall of `Mirror[]`

Today we're going to embark on some advanced geometry. You'll want to freshen up your angles and vectors because there will be a lot of them today!

We're going to be simulating the path of a light ray in 2D space through a hall of mirrors - a mirror being a plane of finite length that, upon the light ray hitting it, will reflect the light ray with the same angle of incidence like [this image here](http://i.imgur.com/NcJrpRT.png). The mirrors are double-sided and have zero thickness.

You will be given a set of mirrors, defined by a start and end point, and a light ray, represented by a starting position, a starting vector (that may or may not be normalized) and a distance. You will have to simulate the light ray travelling for the given distance accounting for any reflections on the mirrors, assuming Euclidan geometry and *no* fancy stuff like refraction, special relativity or similar.

# Formal Inputs and Outputs

## Input Description

You will be given a number **N**, which is the number of mirrors in the world. You will then be given **N** lines of input in the format:

```
X1 Y1 X2 Y2
```

Where (X1,Y1) and (X2,Y2) represent the start and end points of a mirror.

After that you will be given one last line of input in the format:

```
PX PY VX VY D
```

Where (PX,PY) represents the starting position of the light ray in the world, (VX,VY) is the vector representing the light ray's direction in the world (be sure to normalize this beforehand) and D is the distance it will travel.

## Output Description

You will print a line in the format:

```
PX PY
```

Where (PX,PY) is the final position of the vector in the world.

# Sample Inputs & Output

## Sample Input

```
1
-1 0 1 0
-1 -1 1 1 2.828427
```

## Sample Output

```
1 -1
```

# Notes

You will need to have knowledge of the following things to solve this challenge:

* Vectors
* Matrices, depending on how you solve the challenge
* Angles and line geometry

Title: [8/08/2014] Challenge #174 [Hard] Convex Hull Problem
Text: # [](#HardIcon) _(Hard)_: Convex Hull Problem

I have [a collection of points, called **P**](http://i.imgur.com/yDhKB22.png). For this challenge the points will all be on a 2D plane. The Convex Hull problem is to find a convex polygon made from points in **P** which contains all of the points in **P**. There are several approaches to this problem, including brute-force (not good) and several O(n^(2)) solutions (naive, not brilliant) and some fairly in-depth algorithms.

Some such algorithms are described [here (a Java applet, be warned - change the display to 2d first)](http://www.cse.unsw.edu.au/~lambert/java/3d/hull.html) or on [Wikipedia](http://en.wikipedia.org/wiki/Convex_hull_algorithms#Algorithms). The choice is yours, but because you're in /r/DailyProgrammer try and challenge yourself! Try and implement one of the more interesting algorithms.

For example, a convex hull of P:

* [Cannot be this](http://i.imgur.com/VCmqplP.png) because a point is excluded from the selection

* [Also cannot be this](http://i.imgur.com/C4IhIxa.png) because the shape is not convex - the triangles enclosed in green are missing

* [Looks like this](http://i.imgur.com/rbvhJZa.png). The shape is convex and contains all of the points in the image - either inside it or as a boundary.

## Input Description

First you will be given a number, **N**. This number is how many points are in our collection **P**.

You will then be given **N** further lines of input in the format:

```
    X,Y
```

Where X and Y are the co-ordinates of the point on the image. Assume the points are named in alphabetical order as A, B, C, D, ... in the order that they are input.

## Output Description

You must give the convex hull of the shape in the format:

```
    ACFGKLO
```

Where the points are described in no particular order. (as an extra challenge, make them go in order around the shape.)

# Notes

In the past we've had some very pretty images and graphs from people's solutions. If you feel up to it, add an image output from your challenge which displays the convex hull of the collection of points.

Title: [8/01/2014] Challenge #173 [Hard] Road Trip Game
Text: #Description:

The [Oregon Trail] (http://en.wikipedia.org/wiki/The_Oregon_Trail_(video_game\)) is a very iconic game. Essentially it is a road trip going from a start location to an end location. You must manage and overcome various challenges and obstacles. The game was intended for education to teach about the life of a pioneer in North America in the 19th century.

For this Friday Hard challenge you will make your own road trip game. To allow freedom for creativity I will not be placing too many narrow requirements on you for this challenge. The difficulty of this challenge is design and implementation.

Your game must meet the following requirements:

* It must involve travel. You are going from a starting point to an end point. Maybe you complete the journey. Probably most often you do not.

* It must have a scoring system. The better the score the better you do.

* It must involve at least 1 resource in limited supply that must be managed.

A quick note on the resource. The Oregon trail has several resources like food, arrows, parts for the wagon to fix it and so on. It gives a way to gain/use/lose these resources. Without the proper amount you fail your journey. The resources should fit your game's theme. If you do it in space, fuel for a spacecraft. If you are on a boat, you need tar to fix holes or cloth to repair sails. Etc.

#Input:

Up to you how you manage the game. Part of this being hard is the design falls on you.

#Output:

Text/Graphics/Other - up to you. Ideally you need an interface that a human can use and it should have some minor appeal/ease of use.

Title: [7/18/2014] Challenge #171 [Hard] Intergalatic Bitstream
Text: #Description:

Keeping with our "Bit" theme this week. We will look into the future. It is 2114. We have colonized the Galaxy. To communicate we send 140 character max messages using [A-Z0-9]. The technology to do this requires faster than light pulses to beam the messages to relay stations.

Your challenge is to implement the compression for these messages. The design is very open and the solutions will vary.

Your goals:

* Compact 140 Bytes down to a stream of bits to send and then decompact the message and verify 100% data contained.

* The goal is bit reduction. 140 bytes or less at 8 bits per byte so thats 1120 bits max. If you take a message of 140 bytes and compress it to 900 bits you have 220 less bits for 20% reduction.

#Input:

A text message of 140 or less characters that can be [A-Z0-9]

#Output:

```
Read Message of x Bytes.
Compressing x*8 Bits into y Bits. (z% compression)
Sending Message.
Decompressing Message into x Bytes.
Message Matches!
```

* x - size of your message
* x* 8 = bits of your message
* z - the percentage of message compressed by
* y bits of your bit stream for transmission

So compress your tiny message and show some stats on it and then decompress it and verify it matches the original message.

#Challenge Inputs:

three messages to send:

```
REMEMBER TO DRINK YOUR OVALTINE


GIANTS BEAT DODGERS 10 TO 9 AND PLAY TOMORROW AT 1300
```

SPACE THE FINAL FRONTIER THESE ARE THE VOYAGES OF THE BIT STREAM DAILY PROGRAMMER TO SEEK OUT NEW COMPRESSION

#Congrats!

We are a trending subreddit for today 7-18-2014. Welcome to first time viewers of /r/dailyprogrammers checking out our cool subreddit. We have lots of programming challenges for you to take on in the past and many to look forward to in the future.

Title: [7/11/2014] Challenge #170 [Hard] Swiss Tournament with a Danish Twist
Text: #Description:

Swiss Tournament with a Danish Twist

For today's challenge we will simulate and handle a [Swiss System Tournament] (http://en.wikipedia.org/wiki/Swiss-system_tournament)
that also runs the Danish Variation where players will only player each other at most once during the tournament.

We will have a 32 person tournament. We will run it 6 rounds. Games can end in a win, draw or loss. Points are awarded. You will have to accomplish some tasks.

* Randomly Generate 32 players [using the Test Data challenge](http://www.reddit.com/r/dailyprogrammer/comments/28vgej/6232014_challenge_168_easy_final_grades_test_data/) you can generate names
* Generate Random Pairings for 16 matches (32 players and each match has 2 players playing each other)
* Randomly determine the result of each match and score it
* Generate new pairings for next round until 6 rounds have completed
* Display final tournament results.

#Match results and Scoring.

Each match has 3 possible outcomes. Player 1 wins or player 2 wins or both tie. You will randomly determine which result occurs.

For scoring you will award tournament points based on the result.

The base score is as follows.

* Win = 15 points
* Tie = 10 points
* Loss = 5 Points.

In addition each player can earn or lose tournament points based on how they played. This will be randomly determined. Players can gain up to 5 points or lose up to 5
tournament points. (Yes this means a random range of modifying the base points from -5 to +5 points.

##Example:

Player 1 beats player 2. Player 1 loses 3 bonus points. Player 2 gaines 1 bonus points. The final scores:

* Player 1 15 - 3 = 12 points
* Player 2 5 + 1 = 6 points

#Pairings:

Round 1 the pairings are random who plays who. After that and all following rounds pairings are based on the Swiss System with Danish variation. This means:

* #1 player in tournament points players #2 and #3 plays #4 and so on.
* Players cannot play the same player more than once.

The key problem to solve is you have to track who plays who. Let us say player Bob is #1 and player Sue is #2. They go into round 5 and they should play each other.
The problem is Bob and Sue already played each other in round 1. So they cannot play again. So instead #1 Bob is paired with #3 Joe and #2 Sue is played with #4 Carl.

The order or ranking of the tournaments is based on total tournament points earned. This is why round 1 is pure random as everyone is 0 points. As the rounds progress the tournament point totals will change/vary and the ordering will change which effects who plays who. (Keep in mind people cannot be matched up more than once in a tournament)

#Results:

At the end of the 6 rounds you should output by console or file or other the results.
It should look something like this. Exact format/heading up to you.

```
Rank  Player   ID  Rnd1       Rnd2   Rnd3   Rnd4   Rnd5   Rnd6   Total
===========================================================================
1              Bob            23     15            17            13            15            15
      16                 91
2              Sue            20     15            16            13            16            15
      15                 90
3              Jim            2      14            16            16            13            15
      15                 89
..
..
31             Julie   30     5             5             0             0             1             9
      20
32             Ken            7      0             0             1             5             1
      5                  12
```

#Potential for missing Design requirements:

The heart of this challenge is solving the issues of simulating a swiss tournament using a random algorithm to determine results vs accepting input that
tells the program the results as they occur (i.e. you simulate the tournament scoring without having a real tournament) You also have to handle the Danish requirements
of making sure pairings do not have repeat match ups. Other design choices/details are left to you to design and engineer. You could output a log showing pairings on each
round and showing the results of each match and finally show the final results. Have fun with this.

#Our Mod has bad Reading comprehension:

So after slowing down and re-reading the wiki article the Danish requirement is not what I wanted. So ignore all references to it. Essentially a Swiss system but I want players only to meet at most once.

The hard challenge of handling this has to be dealing with as more rounds occur the odds of players having played each other once occurs more often. You will need to do more than 1 pass through the player rooster to handle this. How is up to you but you will have to find the "best" way you can to resolve this. Think of yourself running this tournament and using paper/pencil to manage the pairings when you run into people who are paired but have played before.

Title: [7/4/2014] Challenge #169 [Hard] Convex Polygon Area
Text: # [](#HardIcon) _(Hard)_: Convex Polygon Area

A convex polygon is a geometric polygon (ie. sides are straight edges), where all of the interior angles are less than 180'. For a more rigorous definition of this, see [this page](http://www.mathopenref.com/polygonconvex.html).

The challenge today is, given the points defining the boundaries of a convex polygon, find the area contained within it.

## Input Description

First you will be given a number, **N**. This is the number of vertices on the convex polygon.
Next you will be given the points defining the polygon, in no particular order. The points will be a 2-D location on a flat plane of infinite size. These will always form a convex shape so don't worry about checking that

in your program. These will be in the form `x,y` where `x` and `y` are real numbers.

## Output Description

Print the area of the shape.

# Example Inputs and Outputs

## Example Input 1

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

## Example Output 1

```
6.5
```

## Example Input 2

```
7
1,2
2,4
3,5
5,5
5,3
4,2
2.5,1.5
```

## Example Output 2

```
9.75
```

# Challenge

## Challenge Input

```
8
-4,3
1,3
2,2
```

```
    2,0
    1.5,-1
    0,-2
    -3,-1
    -3.5,0
```

## Challenge Output

```
    24
```

# Notes

Dividing the shape up into smaller segments, eg. triangles/squares, may be crucial here.

# Extension

I quickly realised this problem could be solved much more trivially than I thought, so complete this too. Extend your program to accept 2 convex shapes as input, and calculate the combined area of the resulting intersected shape, similar to how is described [in this challenge](http://www.reddit.com/r/dailyprogrammer/comments/23b1pr/4182014_challenge_158_hard_intersecting_rectangles/).

Title: [6/20/2014] Challenge #167 [Hard] Park Ranger
Text: # [](#HardIcon) _(Hard)_: Park Ranger

Ranger Dan owns a wildlife park in an obscure country somewhere in Europe. The park is an absolute mess, though! Litter covers every walkway. Ranger Dan has been tasked with ensuring all of the walkways are clean on a daily basis. However, doing this on a daily basis can take some time - Dan to ensure that time is not wasted travelling down walkways that have already been checked. Each walkway is checked by walking along it once, from one end to another.

Dan's park is represented as a - you guessed it - graph (with a distance matrix), as covered in [Challenge 166bh](http://www.reddit.com/r/dailyprogrammer/comments/287jxh/6152014_challenge_166b_hard_a_day_in_the_life_of/) and [Challenge 152h](http://www.reddit.com/r/dailyprogrammer/comments/20cydp/14042014_challenge_152_hard_minimum_spanning_tree/). To get to grips with distance matrices and graphs in general, look at the descriptions for those two challenges. The walkways are represented as edges/arcs in the graph, and the vertices/nodes of the graph represent where two walkways join or split up.

Dan has the option of setting up two huts at any two vertices within the park - from where the walkway-checking journey can begin and end. You are being paid to write a program which will find which two vertices are the best place to put the huts in such a way that the time checking every walkway (edge) at least once (an [Eulerian path](http://en.wikipedia.org/wiki/Eulerian_walkway)) is as low as possible - or if it doesn't actually matter where the journey begins or ends. Whether it matters or not will depend on the graph of the park itself.

# Formal Inputs and Outputs

## Input Description

You will be given a number **N** which will represent the number of vertices in the graph of the park. N will be between 1 and 26 inclusive.

You will then be given a distance matrix, with newlines separating rows and commas separating columns. -1 is used to denote that there is no route connecting those two vertices. For the sake of simplicity, the vertices in the graph are assumed to be named A, B, C, D and so on, with the matrix representing them in that order, left-to-right and top-to-bottom, like [this network](http://i.imgur.com/RIfsghM.png) and its corresponding [distance matrix](http://i.imgur.com/iXuaqNT.png).

## Output Description

If it doesn't matter which vertices Dan starts and ends the journey from, print

```
Any
```

However, if starting and ending at two distinct vertices give a shortest (semi-Eulerian) path to check each walkway at least once, then print them like so:

```
A J
```

# Example Inputs and Outputs

## Example Input 1

```
10
-1,-1,-1,-1,30,38,10,21,48,33
-1,-1,-1,47,-1,25,48,-1,-1,37
-1,-1,-1,19,27,-1,37,43,15,37
-1,47,19,-1,-1,34,29,36,-1,42
30,-1,27,-1,-1,-1,-1,43,47,-1
38,25,-1,34,-1,-1,38,49,-1,43
10,48,37,29,-1,38,-1,-1,-1,48
21,-1,43,36,43,49,-1,-1,28,-1
48,-1,15,-1,47,-1,-1,28,-1,-1
33,37,37,42,-1,43,48,-1,-1,-1
0 odd vertices
```

## Example Output 1

```
Any
```

## Example Input 2

```
10
-1,12,28,-1,16,-1,34,-1,-1,27
12,-1,19,35,27,-1,-1,-1,-1,17
28,19,-1,20,15,25,35,-1,-1,-1
-1,35,20,-1,-1,-1,-1,-1,-1,15
16,27,15,-1,-1,-1,33,-1,-1,10
-1,-1,25,-1,-1,-1,27,32,19,36
34,-1,35,-1,33,27,-1,30,32,-1
-1,-1,-1,-1,-1,32,30,-1,18,12
-1,-1,-1,-1,-1,19,32,18,-1,-1
27,17,-1,15,10,36,-1,12,-1,-1
```

## Example Output 2

```
D E
```

# Challenge

## Challenge Input

(this represents a park with 20 vertices.)

```
20
-1,-1,-1,-1,15,-1,-1,57,-1,-1,-1,67,-1,-1,-1,23,-1,67,-1,66
-1,-1,-1,-1,-1,-1,53,-1,23,-1,-1,-1,-1,-1,54,-1,-1,-1,-1,-1
-1,-1,-1,-1,-1,63,-1,-1,-1,-1,66,84,84,-1,-1,-1,43,-1,43,-1
```

```
-1,-1,-1,-1,90,-1,-1,-1,-1,-1,37,20,-1,-1,-1,89,-1,28,-1,-1
15,-1,-1,90,-1,-1,-1,34,-1,-1,-1,21,-1,-1,-1,62,-1,80,-1,-1
-1,-1,63,-1,-1,-1,-1,-1,-1,-1,-1,-1,39,-1,-1,-1,45,-1,35,-1
-1,53,-1,-1,-1,-1,-1,-1,51,58,-1,-1,-1,90,76,-1,-1,-1,-1,84
57,-1,-1,-1,34,-1,-1,-1,-1,-1,-1,-1,-1,62,24,30,-1,-1,-1,-1
-1,23,-1,-1,-1,-1,51,-1,-1,75,-1,-1,-1,67,58,-1,-1,-1,-1,52
-1,-1,-1,-1,-1,-1,58,-1,75,-1,-1,-1,-1,76,-1,-1,-1,-1,-1,25
-1,-1,66,37,-1,-1,-1,-1,-1,-1,-1,-1,50,-1,-1,-1,-1,-1,-1,-1
67,-1,84,20,21,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,72,-1,49,-1,-1
-1,-1,84,-1,-1,39,-1,-1,-1,-1,50,-1,-1,-1,-1,-1,85,-1,-1,-1
-1,-1,-1,-1,-1,-1,90,62,67,76,-1,-1,-1,-1,-1,-1,-1,-1,-1,88
-1,54,-1,-1,-1,-1,76,24,58,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1
23,-1,-1,89,62,-1,-1,30,-1,-1,-1,72,-1,-1,-1,-1,-1,21,-1,-1
-1,-1,43,-1,-1,45,-1,-1,-1,-1,-1,-1,85,-1,-1,-1,-1,-1,38,-1
67,-1,-1,28,80,-1,-1,-1,-1,-1,-1,49,-1,-1,-1,21,-1,-1,-1,-1
-1,-1,43,-1,-1,35,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,38,-1,-1,-1
66,-1,-1,-1,-1,-1,84,-1,52,25,-1,-1,-1,88,-1,-1,-1,-1,-1,-1
```

## Challenge Output

```
K S
```

# Notes

You may need to reuse some code from [Challenge 166bh](http://www.reddit.com/r/dailyprogrammer/comments/287jxh/6152014_challenge_166b_hard_a_day_in_the_life_of/). This is a ~~fairly~~ difficult challenge and is a subset of the [Route Inspection](http://en.wikipedia.org/wiki/Chinese_postman_problem) problem. You'll need to look at all of the vertices with an odd valency.

The degree/valency of a vertex/node is defined as the number of edges/arcs incident to it. If every vertex has degree 0 then there will be an Eulerian cycle through the graph meaning that all checking paths through the park will have the same length - ie. print `Any`.

Title: [6/15/2014] Challenge #166b [Hard] A Day in the Life of a Network Router
Text: # [](#HardIcon) _(Hard)_: A Day in the Life of a Network Router

Every time you send or receive data across the internet, it has navigated itself through tens or hundreds of intermediate destinations to finally reach its target. This involves a ton of extremely well optimised algorithms to find the fastest way to get from A to B - and all of this happens without you knowing about it - until now. The network engineers at Notfast© Internet have detected a problem with a central node - it's not letting any packets through! They are hiring some engineers to manually route the packets while they go about fixing the problem.

You are given a [distance Matrix](http://en.wikipedia.org/wiki/Distance_matrix) (which we met [back in April](http://www.reddit.com/r/dailyprogrammer/comments/20cydp/14042014_challenge_152_hard_minimum_spanning_tree/) - go check out that for a more in depth discussion of graphs) to represent the portion of the network you are dealing with. The pings between nodes on the network will all be different, and it is the job of your algorithm to account for this. Your job and challenge will be to write a program that will find the route through the network from one node to another that has the shortest ping.

# Formal Inputs and Outputs

## Input Description

You will be given a number **N** which will represent the number of nodes on the network. N will be between 1 and 26 inclusive - although 1 would be a bit pointless.

You will then be given a distance matrix, with newlines separating rows and commas separating columns. -1 is used to denote that there is no route connecting those two nodes. For the sake of simplicity, the vertices in the graph are assumed to be named A, B, C, D and so on, with the matrix representing them in that order, left-to-right and top-to-bottom, like [this network](http://i.imgur.com/RIfsghM.png) and its corresponding [distance matrix](http://i.imgur.com/iXuaqNT.png).

Finally, you will be given 2 vertices (represented as letters A-Z), **V1** and **V2**. You are to find the path from **V1** to **V2**.

## Output Description

You are to print out the path from **V1** to **V2**, in the format ABCDEFG - one letter after the other.

# Example Inputs and Outputs

## Example Input

This represents the same example network given above.

```
8
-1,11,9,6,-1,-1,-1,-1
11,-1,-1,5,7,-1,-1,-1
9,-1,-1,12,-1,6,-1,-1
6,5,12,-1,4,3,7,-1
-1,7,-1,4,-1,-1,2,-1
-1,-1,6,3,-1,-1,8,10
-1,-1,-1,7,2,8,-1,6
-1,-1,-1,-1,-1,10,6,-1
B H
```

## Example Output

```
BEGH
```

(this represents [this path](http://i.imgur.com/Aajz4zj.png) through the network.)

# Challenge

## Challenge Input

(this represents a network with 26 nodes.)

```
26
-1,39,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,57,18,-1,-1,-1
39,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,94,-1,-1,-1,-1,-1,-1,74,86,-1,-1,-1
-1,-1,-1,86,-1,-1,-1,-1,-1,-1,52,-1,51,-1,-1,33,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1
-1,-1,86,-1,-1,-1,-1,-1,-1,-1,82,31,-1,-1,-1,-1,-1,51,-1,-1,-1,-1,-1,-1,-1,-1
-1,-1,-1,-1,-1,81,-1,78,20,39,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1
-1,-1,-1,-1,81,-1,-1,48,-1,-1,-1,-1,-1,-1,-1,-1,81,-1,83,-1,-1,-1,-1,-1,-1,-1
-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,18,-1,-1,-1,-1,-1,-1,92,-1,-1,-1,-1,-1
-1,-1,-1,-1,78,48,-1,-1,63,35,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1
-1,-1,-1,-1,20,-1,-1,63,-1,95,-1,-1,-1,-1,-1,-1,75,-1,-1,-1,-1,-1,-1,-1,-1,-1
-1,-1,-1,-1,39,-1,-1,35,95,-1,-1,-1,-1,-1,-1,-1,48,-1,-1,-1,-1,-1,-1,-1,-1,-1
-1,-1,52,82,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,64,-1,-1,-1,-1,-1,-1,73,-1
-1,-1,-1,31,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,81,-1,-1,-1,-1,-1,44,97,-1
-1,-1,51,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,28,-1,14,-1,-1,-1,-1,32,-1,-1,-1,-1,-1
-1,-1,-1,-1,-1,-1,18,-1,-1,-1,-1,-1,28,-1,-1,33,-1,-1,-1,-1,59,-1,-1,-1,-1,-1
-1,94,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,74,-1,33,-1,-1,-1,50
-1,-1,33,-1,-1,-1,-1,-1,-1,-1,-1,-1,14,33,-1,-1,50,-1,-1,-1,-1,-1,-1,-1,-1,-1
```

```
    -1,-1,-1,-1,-1,81,-1,-1,75,48,-1,-1,-1,-1,-1,50,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1
    -1,-1,-1,51,-1,-1,-1,-1,-1,-1,64,81,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,74,-1,-1
    -1,-1,-1,-1,-1,83,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,87,32,-1
    -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,74,-1,-1,-1,-1,-1,-1,-1,-1,79,-1,38
    -1,-1,-1,-1,-1,-1,92,-1,-1,-1,-1,-1,32,59,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1
    57,74,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,33,-1,-1,-1,-1,-1,-1,-1,26,-1,-1,-1
    18,86,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,26,-1,-1,-1,62
    -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,44,-1,-1,-1,-1,-1,74,87,79,-1,-1,-1,-1,-1,-1
    -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,73,97,-1,-1,-1,-1,-1,-1,32,-1,-1,-1,-1,-1,-1,-1
    -1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,50,-1,-1,-1,-1,38,-1,-1,62,-1,-1,-1
    G B
```

## Challenge Output

```
    481
    GNPCDLXTZWAB
```

# Notes

The essence of the challenge here is just finding the shortest route from one node to another in a simple undirected graph. There are several algorithms you could use - two of which are [Edsger Dijkstra's algorithm](http://en.wikipedia.org/wiki/Dijkstra%27s_algorithm) and the [A* algorithm](http://en.wikipedia.org/wiki/A*_search_algorithm).

Title: [6/6/2014] Challenge #165 [Hard] Simulated Ecology - The Forest
Text: #Description:

The study of balance is interesting. Take for example a forest. Forests are very complex eco-systems with lots of things happening. For this challenge we will
simulate a virtual forest and watch over simulated time the effects of a forest. We will see trees grow and be harvested. We will see the impact of
industry upon the forest and watch as the wild life "fights" back.

For this simulated forest we will be dealing with 3 aspects.

* Trees which can be a Sapling, Tree or Elder Tree.
* Lumberjacks (He chops down down trees, he eats his lunch and goes to the Lava-try)
* Bears (He maws the lumberjacks who smells like pancakes)

##Cycle of time:

The simulation will simulate by months. You will progessive forward in time with a "tick". Each "tick" represents a month. Every 12 "ticks" represents a year.
Our forest will change and be in constant change. We will record the progress of our forest and analyze what happens to it.

##Forest:

The forest will be a two dimensional forest. We will require an input of N to represent the size of the forest in a grid that is N x N in size.
At each location you can hold Trees, Bears or Lumberjacks. They can occupy the same spot but often events occur when they occupy the same spot.

Our forest will be spawned randomly based on the size. For example if your value of N = 10. You will have a 10 by 10 forest and 100 spots.

10% of the Forest will hold a Lumberjack in 10 random spots. (using our 100 spot forest this should be 10 lumberjacks)
50% of the Forest will hold Trees (Trees can be one of 3 kinds and will start off as the middle one of "Tree") in random spots.
2% of the Forest will hold Bears.

How you get the size of the forest is up to you. Either users enter it in, read it from a file, pass by argument or hard coded. Your choice. But you have
to spawn the initial forest with the above percentages. I would recommend keeping N like 5 or higher. Small Forests are not much fun.

##Events:

During the simulation there will be events. The events occur based on some logic which I will explain below. The events essentially are the spawning of
new Trees, Lumberjacks, Bears or the decay of Trees, Lumberjacks and Bears. I will detail the events below in each description of the 3 elements of our forest.

## Trees:

Every month a Tree has a 10% chance to spawn a new "Sapling". In a random open space adjacent to a Tree you have a 10% chance to create a "Sapling".
For example a Tree in the middle of the forest has 8 other spots around it. One of these if they do not have a type of Tree in it will create a "Sapling".

After 12 months of being in existence a "Sapling" will be upgrade to a "Tree". A "Sapling" cannot spawn other trees until it has matured into a "Tree".

Once a "Sapling" becomes a tree it can spawn other new "Saplings". At this point once a "Sapling" matures into a "Tree" it exists and matures. When a "Tree"
has been around for 120 months (10 years) it will become an "Elder Tree".

Elder Trees have a 20% chance to spawn a new "Sapling" instead of 10%.

If there are no open adjacent spots to a Tree or Elder Tree it will not spawn any new Trees.

## Lumberjacks:

They cut down trees, they skip and jump they like to press wild flowers.

Lumberjacks each month will wander. They will move up to 3 times to a randomly picked spot that is adjacent in any direction. So for example a Lumberjack in the middle of your grid has
8 spots to move to. He will wander to a random spot. Then again. And finally for a third time.

When the lumberjack moves if he encounters a Tree (not a sapling) he will stop and his wandering for that month comes to an end. He will then harvest the Tree for lumber. Remove the tree. Gain 1 piece of lumber. Lumberjacks will not harvest "Sapling". They will harvest an Elder Tree.
Elder Trees are worth 2 pieces of lumber.

Every 12 months the amount of lumber harvested is compared to the number of lumberjacks in the forest. If the lumber collected equals or exceeds the amount of lumberjacks
in the forest a new lumberjack is hired and randomly spawned in the forest. Actually a math formula is used to determine if we hire 1 or many lumberjacks. We hire a number
of new lumberjacks based on lumber gathered. Let us say you have 10 lumberjacks. If you harvest 10-19 pieces of lumber you would hire 1 lumberjack. But if you harvest 20-29

pieces of lumber you would hire 2 lumberjacks. If you harvest 30-39 you would gain 3 lumberjacks. And so forth.

However if after a 12 month span the amount of lumber collected is below the number of lumberjacks then a lumberjack is let go to save money and 1 random lumberjack
is removed from the forest. However you will never reduce your Lumberjack labor force below 0.

## Bears:

They wander the forest much like a lumberjack. Instead of 3 spaces a Bear will roam up to 5 spaces. If a bear comes across a Lumberjack he will stop his wandering
for the month. (For example after 2 moves the bear lands on a space with a lumberjack he will not make any more moves for this month)

Lumberjacks smell like pancakes. Bears love pancakes. Therefore the Bear will unfortunately maw and hurt the lumberjack. The lumberjack will be removed from the
forest (He will go home and shop on wednesdays and have buttered scones for tea).

We will track this as a "Maw" accident. During the course of 12 months if there 0 "Maw" accidents then the Bear population will increase by 1.
If however there are any "Maw" accidents the Lumberjacks will hire a Zoo to trap and take a Bear away. Remove 1 random Bear. Note that if your Bear population reaches
0 bears then there will be no "Maw" accidents in the next year and so you will spawn 1 new Bear next year.

If there is only 1 lumberjack in the forest and he gets Maw'd. He will be sent home. But a new one will be hired immediately and respawned somewhere else in the forest.
The lumberjack population will not drop below 1.

## Time:

The simulation occurs for 4800 months (400 years). Or until the following condition occur.

* You have 0 Trees left in the forest. So no Saplings, Trees or Elder Trees exist.

#Output:

Every month you will print out a log of spawn or decay events. If nothing happens then nothing is logged.

Example:

```
    Month [0001]: [3] pieces of lumber harvested by Lumberjacks.
    Month [0001]: [10] new Saplings Created.
    Month [0002]: [2] pieces of lumber harvested by Lumberjacks.
    Month [0002]: [9] new Saplings Created.
    Month [0003]: [1] Lumberjack was Maw'd by a bear.
            Month [0120]: [10] Trees become Elder Trees
```

Every year you will print out a log of events for yearly events:

```
    Year [001]: Forest has 30 Trees, 20 Saplings, 1 Elder Tree, 9 Lumberjacks and 2 Bears.
            Year [001]: 1 Bear captured by Zoo.
            Year [001]: 9 pieces of lumber harvested 1 new Lumberjack hired.
            Year [002]: Forest has 50 Trees, 25 Saplings, 2 Elder Tree, 10 Lumberjacks and 1 Bears.
            Year [002]: 1 new Bear added.
            Year [003]: Forest has 100 Trees, 99 Saplings, 10 Elder Tree, 1 Lumberjacks, and 0 Bears.
            Year [003]: 1 new Bear added.
```

```
Year [003]: 3 Pieces of lumber harvested 3 new Lumberjacks hired.
```

#Optional Output 1:

At the end of the simulation you can bring out an ASCII graph showing the yearly populations of Bears, Trees, Lumberjacks and open space (BTL Graph)
I recommend 50 Spots and each spot = 2%.

Example:

```
 year 1: [BTTTTTTTTTTTTTTTTTTTTLLL__________________________]
 year 2: [BBTTTTTTTTTTTTTTTTTTTLLLL_________________________]
 year 3: [BTTTTTTTLLLLLLLL__________________________________]
 year 4: [BBBTTTTTTTTTTTTTTTTTLLLLLLLL______________________]
```

So for year 1 we had 2% Bears, 40% Trees (Saplings+Trees+Elder Trees), 6% Lumberjacks and the rest was open space
Each spot is 2%. We have 50 characters. So 100%. We round "up" for figuring out how many to display and just use "_" as filler at the end for open space.

#Optional Output 2:

You can over the course of the simulation output the "Map" in ASCII or any other form you wish. Use like "B" For bear "S" for sapling "T" for tree "E" for Elder Tree, "L" For lumberjack and "." for empty.
Some people can use "animated" ascii via like a ncurses library and show in realtime what is happening. (logs go to a file or not shown) Etc. Ultimately be creative
here in how you might want to show over time the impact of how the forest is changing.

Or you can just print out the forest every year or every 10 years.

#Ackward events/issues/etc:

When bears and lumberjacks roam if the random spot already has a bear or lumberjack in it a new spot is picked. If the 2nd attempt at a spot still has a same kind of element then it will stop roaming for the month. More or less we don't want more than 1 lumberjacks or bears in the same spot.

Bears can roam into a Tree spot. Nothing happens. If a bear roams into a lumberjack he maws him. If a lumberjack roams into a Bear spot he will get maw'd by the bear.

#Spawn/Decay/Removal Rates:

You might encounter issues with these. Feel free to tweak as needed. The challenge is more a test of design. Picking/playing with and testing these rates is part of design work. It might look good on paper but when tested it might not work without some minor tweaks.

Title: [6/1/2014] Challenge #164 [Hard] What the Funge is this!?
Text: #Description

Befunge is a programming language invented by Chris Pressey. The language was made with the goal of being extremely difficult to compile. Well, what makes it so difficult? Consider this 'Hello World' program written by /u/Elite6809 in Befunge:

```
0 952**7+    v
>:3-        v6
 >-  :3-::7v:6
 1        -8*
 *        :+
+8        2
66 v-+9**25<
:^    **464<
^       +8:<

  >       :v
  ^   ,+*88_@
```

At first glance, this may look like gibberish (similar to BrainFuck) This is because this language isn't read in the same way as normal programming languages, which are read in a linear fashion. Instead, it is read in two dimensions, by following an instruction pointer throughout the file. The pointer defaults at moving from left to right, but characters such as [^, v, <, >] will change the direction of the instruction pointer to any of the cardinal directions. Variables are stored on a stack, and all operations pop one or more values off the stack, then either push the result back onto the stack, or change the direction of the instruction pointer. This results in a puzzling programming language, forcing you to work with space management and a limited-access value stack.

Your job is to create a Befunge interpreter that will take in a list of user inputs (read in order by any & or ~ command) and a two-dimensional Befunge program, and output the result.

Be careful! Befunge is a self-modifying programming language (using the p command) and can change itself during runtime!

#Inputs & Outputs
##Input Description
Line 1 will consist of any values that will be used as input to the program. every time a user input command is called, it will use the next value in your list of inputs. If there is no input needed, it should be a single zero.

The rest of your input will be the Befunge program to interpret. Befunge-93 has a maximum size of 80 characters horizontally by 25 characters vertically, so it should be within those parameters.

##Output Description

The program should output a new value or character whenever an output command (. or ,) is called in your program.

##Sample Inputs

Ex.1 (Simple 'Hello World')

```
0

"!dlrow olleH">:#,_@
```

Ex.2 (Factorial program written by /u/AJ_Black)

```
9

0&>:1v
  |:-<
<$<v:\
    ^ *_$.@
```

##Sample Outputs

Sample outputs:

Ex.1:

Hello world!

Ex.2:

362880

#Bonus

Challenge:

Now that you've made an interpreter, put it to the test by making a Befunge program of your own. Make it serenade you by singing "99 Bottles,", or challenge yourself to a game of "higher or lower."

#Help

If you're stuck, try reading a few resources such as the Wiki page

http://en.wikipedia.org/wiki/Befunge

There's a good example of Befunge code here, written for our [Easy] challenge

http://www.reddit.com/r/dailyprogrammer/comments/26ijiu/5262014_challenge_164_easy_assemble_this_scheme/chrdyfa

#Notes

You can verify that your interpreter works using the following online interpreter -

http://www.bedroomlan.org/tools/befunge-93-playground

Title: [5/23/2014] Challenge #163 [Hard] Intersecting Lines in 2-D space
Text: #Descripton:

Given a typical x/y coordinate system we can plot lines. It would be interesting to know which lines intersect.

#Input:

A series of lines from 1 to many to put in our 2-D space. The data will be in the form:

```
    (label) (x1 y1) (x2 y2)
```

* (label) will be a letter A-Z
* (x1 y1) will be the coordinates of the starting point on line
* (x2 y2) will be the coordinates of the ending point on line

##example input:

```
A -2.5 .5 3.5 .5
B -2.23 99.99 -2.10 -56.23
C -1.23 99.99 -1.10 -56.23
D 100.1 1000.34 2000.23 2100.23
E 1.5 -1 1.5 1.0
F 2.0 2.0 3.0 2.0
G 2.5 .5 2.5 2.0
```

* Max X can be 1,000,000,000.00
* Max Y can be 1,000,000,000.00

#Output:

The program will list which lines intersect. And which have 0 intersects.

##Example Output:

```
Intersecting Lines:
A B
A C
A E
A G
F G
No intersections:
D
```

#Difficulty:

This is a coder_d00d(tm) unknown difficulty challenge. It could be easy. Could be hard. But it seems cool for a Friday.

* If you want to make it **easier**: input is only 2 lines and you return yes/no
* If you want to make it **harder**: output is the 2 lines and the (x y) point they intersect at.

Title: [5/16/2014] Challenge #162 [Hard] Novel Compression, pt. 3: Putting it all together
Text: # [](#HardIcon) _(Hard)_: Novel Compression, pt. 3: Putting it all together

Welcome to the third and final part of this week's Theme Week. Today is not so much a 'hard' challenge as such, but rather a culmination of this week's efforts. You will be putting your code from Monday and Wednesday into one program that can be operated [via the command line or terminal](http://en.wikipedia.org/wiki/Command-line_interface#Arguments), and will deal with files rather than textual input.

# Formal Inputs and Outputs

## Input Description

The program will take 3 arguments on the command line: the first one will be one of the following:

* `-c` Will compress the input.

* `-d` Will decompress the input.

If it is anything other than these, return an error message. The second argument will be a path to a file that the input data will be read from, and the third argument will be a path to a file that output data will be written to. If there are any more or less than three arguments given, return another error message.

## Output Description

Using the given operation (compress or decompress), the data in the input file will be processed, and the resulting data written to the output file.

# Example Input

There is a plain text copy of Green Eggs and Ham [available here](http://pastie.org/pastes/9180059/text?key=wmyubynyw72ten8m3gzpfw), edited to work with our compression algorithm, which can be used to test your program.

For example, on Windows:

```
    compressor -c eggs.txt eggs-c.txt
```

Or on nearly everything else:

```
    ./compressor -c eggs.txt eggs-c.txt
```

# Notes

It may be an idea to submit your code to a site such as [Github Gist](https://gist.github.com/) or another code sharing site, rather than pasting it in the comments, as the combined code may be quite long.

Hopefully by the end of this we'll have ported this program to a wide selection of languages.

Green Eggs and Ham is copyright of Dr Seuss?. I think the usage here is fair use as it is not for profit.

Title: [5/9/2014] Challenge #161 [Hard] Phone Network
Text: #Description:

Your company has built its own telephone network. This allows all your remote locations to talk to each other. It is your job to implement the program to establish calls between locations.

Calls are dedicated bandwidth on your network. It uses up resources on the network connection between locations. Because of this building a call between two locations on the network can be tricky. As a call is built it continues to use resources and new calls might have to route differently to find a way to reach the source and destination. If there are no ways to build a call then the call will fail.

#Input:

There will be two sets of input. First set deals with what your phone network looks like. The second set will be the series of calls you must handle.

##Network Input:

You must be able to read in network connections. They will be letter names for locations and a number. The number represents how many calls can go across the network link between these two locations. So for example if you have location A and location B and you can have 2 calls between these you will read in a link as:

```
    A B 2
```

Example of list of links for a telephone network:

```
A B 2
A C 2
B C 2
B D 2
C E 1
D E 2
D G 1
E F 2
F G 2
```

##Call Input:

You then have a list of calls to be placed on the network. Each call builds in the order you enter it and it is unknown if the resources will be there or not. You must read in all the calls. The calls simply have pairs listing the source and destination of the call. So for example if you wanted Location C to call Location G you would read in the call as:

```
C G
```

Example of calls to be placed on your example network:

```
A G
A G
C E
G D
D E
A B
A D
```

#Output:

Your program will build the call if it can and list back the route the call took. If the call cannot be placed due to too many calls taking up resources it will indicate the "Call Failed".

##Example output given the above inputs:

```
Call A G -- placed A B D G
Call A G -- placed A C E F G
Call C E -- placed C B D E
Call G D -- placed G F E D
Call D E -- failed
Call A B -- A B
Call A D -- failed
```

#Understanding the Bandwidth:

So a link A B has a unit of "2" - if a call goes across this connection then the amount of calls the link can handle is reduced down to 1. If 1 more call crosses the link then the resource is 0 and the link is full. Any calls trying to be placed cannot cross this link as the bandwidth does not exist to support the call.

Links between locations can support calls in any direction. So a link A B exists the call can go A to B or B to A. In some cases you might have a call that is going over this link as A B and another call going B A.

#Example 2:

```
A B 1
B C 2
C D 2
D E 2
E F 2
F G 2
G A 2
E H 1
H D 1

A C
A D
A D
F D
B D
B D
B E
C F
```

Output could vary but you should be able to build 5 calls with 2 failed.

Title: [5/2/2014] Challenge #160 [Hard] Trigonometric Triangle Trouble, pt. 2
Text: # [](#HardIcon) _(Hard)_: Trigonometric Triangle Trouble, pt. 2

[I'm posting this early because there's a chance I won't have access to the internet tomorrow. Better an hour early than a day late I suppose.]

A triangle on a flat plane is described by its angles and side lengths, and you don't need all of the angles and side lengths to work out everything about the triangle. (This is the same as last time.) However, this time, the triangle will not necessarily have a right angle. This is where more trigonometry comes in. Break out your trig again, people.

[Here's a representation of how this challenge will describe a triangle](http://i.imgur.com/Q3qUoRg.png). Each side is a **lower-case** letter, and the angle opposite each side is an **upper-case** letter - exactly the same as last time. Side a is opposite angle A, side b is opposite angle B, and side c is opposite angle C. However, angle C is not guaranteed to be 90' anymore, meaning the old right-angle trigonometry will not work; the choice of letter is completely arbitrary now. Your challenge is, using trigonometry and given an appropriate number of values, to find the rest of the values.

# Formal Inputs and Outputs

## Input Description

On the console, you will be given a number **N**. You will then be given **N** lines, expressing **some** details of a triangle in the format:

```
3
a=2.45912
A=39
B=56
```

a, A and B are just examples, it could be a, b and B or whatever.

Where all angles are in degrees. Note that, depending on your language of choice, a conversion to radians may be needed to use trigonometric functions such as *sin*, *cos* and *tan*.

## Output Description

You must print out **all** of the details shown below of the triangle in the same format as above.

```
a=2.45912
b=3.23953
c=3.89271
A=39
B=56
C=85
```

The input data will always give enough information and will describe a valid triangle.

# Sample Inputs & Outputs

## Sample Input

```
3
c=7
A=43
C=70
```

## Sample Output

```
a=5.08037
b=6.85706
c=7
A=43
B=67
C=70
```

# Notes

There are 5 more useful trigonometric identities you may find very useful. The 4 from Part 1 aren't great here as they are edge cases of trigonometry.

* [Sum of the angles is 180](http://latex.codecogs.com/gif.latex?%5Cmathbf%7BA%7D+%5Cmathbf%7BB%7D+%5Cmathbf%7BC%7D%3D180%5E%7B%5Ccirc%7D)

* [Sine Rule 1](http://latex.codecogs.com/gif.latex?%5Cfrac%7B%5Cmathbf%7Ba%7D%7D%7Bsin%20%5Cmathbf%7BA%7D%7D%3D%5Cfrac%7B%5Cmathbf%7Bb%7D%7D%7Bsin%20%5Cmathbf%7BB%7D%7D%3D%5Cfrac%7B%5Cmathbf%7Bc%7D%7D%7Bsin%20%5Cmathbf%7BC%7D%7D)

* [Sine Rule 2 (same as above but re-arranged)](http://latex.codecogs.com/gif.latex?%5Cfrac%7Bsin%20%5Cmathbf%7BA%7D%7D%7B%5Cmathbf%7Ba%7D%7D%3D%5Cfrac%7Bsin%20%5Cmathbf%7BB%7D%7D%7B%5Cmathbf%7Bb%7D%7D%3D%5Cfrac%7Bsin%20%5Cmathbf%7BC%7D%7D%7B%5Cmathbf%7Bc%7D%7D)

* [Cosine Rule 1](http://latex.codecogs.com/gif.latex?%5Cmathbf%7Ba%7D%5E2%3D%5Cmathbf%7Bb%7D%5E2+%5Cmathbf%7Bc%7D%5E2-2%5Cmathbf%7Bbc%7D%5Ccos%5Cmathbf%7BA%7D)

* [Cosine Rule 2 (same as above but re-arranged)](http://latex.codecogs.com/gif.latex?%5Cfrac%7B%5Cmathbf%7Bb%7D%5E2+%5Cmathbf%7Bc%7D%5E2-%5Cmathbf%7Ba%7D%5E2%7D%7B2%5Cmathbf%7Bbc%7D%7D%3D%5Ccos%5Cmathbf%7BA%7D)

Title: [4/25/2014] Challenge #159 [Hard] Rock Paper Scissors Lizard Spock - Part 3 Battle Bots
Text: #Theme Week:

We conclude this theme week with our final challenge. Those keeping up with the challenges will find this one easier than normal for a hard as you can use your solutions from the Easy and Intermediate to help you.

Those new to this one please see these challenges

* [Part 1] (http://www.reddit.com/r/dailyprogrammer/comments/23lfrf/4212014_challenge_159_easy_rock_paper_scissors/)
* [Part 2] (http://www.reddit.com/r/dailyprogrammer/comments/23qy19/4232014_challenge_159_intermediate_rock_paper/)

#Description:

Monday we used random pick.

Wednesday we used a learning AI.

Both of these have issues. They can be better. For this challenge we have 2 goals.

* You will implement your own picking AI bot using your own design.
* Battle the 3 days of bots to get data to form a conclusion.

## Your AI:

Develop your own AI. Try not to use cheats like looking at the player move and just doing a counter for it. Stick with the spirit of the match. The bot can only learn what it saw at the end of a match and does not have the perfect foresight of seeing the other player's move before it picks.

When you post your solution also post in words what your approach is. I recommend using the 4 spaces to hide this with spoilers like you would your code submissions.

## Battle:

You will gather win/tie data on 3 different setups.

* Monday's AI (pure random) vs Wednesday's AI (learned picks)
* Monday's AI (pure random) vs Friday's AI (today's AI)
* Wednesday's AI vs Friday's AI.

For each of this match up of bot vs bot you will run 10,000 games. You will gather the win rate of each bot in the match ups over these 10,000 games and the tie rate.

The conclusion we are trying to reach from this data is 2 things.

* Which AI seems to be performing the best for Rock Paper Scissors Lizard Spock.
* How is the tie rate trending. Does it seem low, high or up and down? Per this [Video] (https://www.youtube.com/watch?v=iapcKVn7DdY) the Character Dr. Sheldon suggests that Rock Paper Scissors Lizard Spock is better due to reduce chance of ties because of the more outcomes from a typical Rock Paper Scissor game.

#Input:

Need a way to pick the various bot match ups and have your program run 10,000 games and print out the results. The last 2 challenges set it up so this should be fairly easy to do.

#Output:

You will generate stats on the 3 match ups and then draw your conclusion on the data. What AI is the best and how is the ties trending.

#Extra Challenge:

Collaborate with other dailyprogrammers and find a way to have your AIs face off against each other.

Title: [4/18/2014] Challenge #158 [Hard] Intersecting Rectangles
Text: # [](#HardIcon) _(Hard)_: Intersecting Rectangles

Computing the area of a [single rectangle](http://i.imgur.com/0W5Oiav.png) is extremely simple: width multiplied by height.
Computing the area of two rectangles is a little more challenging. They can either be separate and thus have their areas calculated individually, [like this](http://i.imgur.com/IefYcFj.png). They can also intersect, in which case you calculate their individual areas, and subtract the area of the intersection, [like this](http://i.imgur.com/6GzHGrh.png).
Once you get to 3 rectangles, there are multiple possibilities: [no intersections](http://i.imgur.com/Ja2TUMv.png), [one intersection of two rectangles](http://i.imgur.com/OgYPfxG.png), [two intersections of two rectangles](http://i.imgur.com/orCodUz.png), [or one intersection of three rectangles (plus three intersections of just two rectangles)](http://i.imgur.com/xW1E588.png).
Obviously at that point it becomes impractical to account for each situation individually but it might be possible. But what about 4 rectangles? 5 rectangles? **N** rectangles?

Your challenge is, given any number of rectangles and their position/dimensions, find the area of the resultant overlapping (combined) shape.

# Formal Inputs and Outputs

## Input Description

On the console, you will be given a number **N** - this will represent how many rectangles you will receive. You will then be given co-ordinates describing opposite corners of **N** rectangles, in the form:

```
x1 y1 x2 y2
```

Where the rectangle's opposite corners are the co-ordinates (x1, y1) and (x2, y2).
Note that the corners given will be the top-left and bottom-right co-ordinates, in that order. Assume top-left is (0, 0).

## Output Description

You must print out the area (as a number) of the compound shape given. No units are necessary.

# Sample Inputs & Outputs

## Sample Input

(representing [this situation](http://i.imgur.com/l2xVFOi.png))

```
3
0 1 3 3
2 2 6 4
1 0 3 5
```

## Sample Output

```
    18
```

# Challenge

## Challenge Input

```
    18
    1.6 1.2 7.9 3.1
    1.2 1.6 3.4 7.2
    2.6 11.6 6.8 14.0
    9.6 1.2 11.4 7.5
    9.6 1.7 14.1 2.8
    12.8 2.7 14.0 7.9
    2.3 8.8 2.6 13.4
    1.9 4.4 7.2 5.4
    10.1 6.9 12.9 7.6
    6.0 10.0 7.8 12.3
    9.4 9.3 10.9 12.6
    1.9 9.7 7.5 10.5
    9.4 4.9 13.5 5.9
    10.6 9.8 13.4 11.0
    9.6 12.3 14.5 12.8
    1.5 6.8 8.0 8.0
    6.3 4.7 7.7 7.0
    13.0 10.9 14.0 14.5
```

## Challenge Output (hidden by default)

[89.48](/spoiler)

# Notes

Thinking of each shape individually will only make this challenge harder. Try grouping intersecting shapes up, or calculating the area of regions of the shape at a time.
Allocating occupied points in a 2-D array would be the easy way out of doing this - however, this falls short when you have large shapes, or the points are not integer values. Try to come up with another way of doing it.

Because this a particularly challenging task, We'll be awarding medals to anyone who can submit a novel solution without using the above method.

Title: [4/11/2014] Challenge #157 [Hard] ASCII Bird
Text: #Description:

In the news lately there has been a lot of press about a game called Flappy Bird. I have noticed many people have rushed to make clones of this game.

For those who want to know more about the game [Click here for wikipedia] (http://en.wikipedia.org/wiki/Flappy_Bird)

So I thought we need to join in on the craze and come up with our own version of Flappy Bird. ASCII Bird. It is flappy bird with ASCII.

More or less you control a bird flying through randomly generated obstacles scrolling right to left at you. You decide when the bird flaps to gain height and if you don't do anything he will fall. If he falls to the ground or hits an obstacle the game is over. For every obstacle he flys over or under with success he gains a point.

#Input:

We will take a single input from the player of the game. A number between 0-4. This represents the "flap" for our bird. The value would represent how high we like our bird to move.

#Output:

This is mostly a visual challenge. After we get the input we have to show the map.

* @ = our bird
* . = empty space
* # = obstacle.

The board will be 10 rows high by 20 columns.

##example:

```
        ..........#.......#.
        ..........#.......#.
        ..........#.........
        ..........#.........
        .@........#.........
        ....................
        ......#.............
        ......#........#....
        ......#........#....
        ......#........#....

        (score 0) 0-4?
```

After you enter a number the forward velocity of the bird will be 2 columns. In those 2 columns you must move the bird based on the velocity. If you typed 1-4 then the board shifts over 2 columns and the bird will go up that many (if it wants to go above the top row it will not)

If you type a 0 instead our bird will decay his flight by 2 rows down.

If flappy bird flys over or under an obstacle he will advance his score by 1 point. If he goes below the bottom row on a decay or makes contact with a obstacle he will die and the game is over (display the final score - maybe ask to play again)

The board is updated 2 columns at a time. You have to keep track of it. Randomly every 7-10 columns on either top or bottom you will generate an obstacle that is 2-4 in height hanging from the top or coming up from the bottom. Once you spawn an obstacle the next will spawn 7-10 columns away. (note each top and bottom needs to be tracked separate and are not related. This can create for some interesting maps)

##example after typing a 2 for our move with above then 2 moves of a 0

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

```
.@......#...........
........#...........
........#...........
....................
....#...............
....#........#......
....#........#......
....#........#......

(score 0) 0-4?

......#.......#...
......#.......#...
......#...........
......#...........
.@....#...........
..................
..#...............
..#........#......
..#........#......
..#........#......

(score 0) 0-4?


....#.......#.....
....#.......#.....
....#.............
....#.............
....#.............
..................
#@...............#
#........#.......#
#........#.......#
#........#.......#

(score 1) 0-4?
```

Our bird spawns in the middle of the rows in height and as above should have 1 column behind him. He will pretty much just move up or down in that column as the board "shifts" its display right to left and generating the obstacles as needed.

#Notes:

As always if you got questions/concerns post away and we can tackle it.

#Extra Challenge:

Make it graphical and go from ASCII Bird to Flappy Bird.

Title: [4/04/2014] Challenge #156 [Hard] uʍop ǝpᴉsd∩ ɓuᴉɥʇǝɯos ɹoɟ ʍoN
Text: #Title: Now for Something Upside down

#Description:

The [Easy] Challenge was delayed 1 day to be on April's Fools Day this week so the moderators could attempt to be clever and turn things upside down by making a super easy challenge to decode a message to just have people post hello world programs. The responses to that challenge was interesting.

To show how things got turned upside down this week's [Hard] challenge we are gonna make text appear upside down.

#Input:

* 1 to many lines of text to convert
* You must read it in from standard input or a file. (No fixed strings hard coded into the program with the input)
* Can handle as input by characters for converting [a-z] [A-Z] [ ] [?!.] [0-9] to upside down characters.

##Example:

```
This is some text that I am writing!
Soon it will be just 4 lines of upside down text.
How did they do it?
We will all know soon.
```

#Output:

The text modified to be upside down.

##Example:

```
˙uoos ʍouʞ llɐ llᴉʍ ǝM
¿ʇᴉ op ʎǝɥʇ pᴉp ʍoH
˙ʇxǝʇ uʍop ǝpᴉsdn ɟo sǝuᴉl ㄣ ʇsnɾ ǝq llᴉʍ ʇᴉ uooS
¡ɓuᴉʇᴉɹʍ ɯɐ I ʇɐɥʇ ʇxǝʇ ǝɯos sᴉ sᴉɥ┴
```

#Notes:

* As part of the [Hard] challenge we leave it to you to figure out how this is possible.
* Solutions might limit which languages you can use.

#More Challenges

In addition to above look into trying these out:

* convert upside down to normal
* find conversions for $&@';/\><+*=_- if any are possible
* given a word search the text count the word matches. Count how many times the word is normal or upside down

## Good single line Test String

The quick brown fox jumps over the lazy dog.?! 0 1 2 3 4 5 6 7 8 9

Title: [4/28/2014] Challenge #154 [Hard] Wumpus Cave Game
Text: #Description:
Across the land the people whisper "Beware the Wumpus. For it slumbers in the cave up yonder in the hills. Only the brave seek him."

This challenge will be about implementing a simple rogue like game. You will create a game engine that will accept simple commands from the user. You will parse the commands and process them. You will score the moves with a point system. The goal of the player is to score the most points with 1 life. The cave will be a randomly generated N sized cave.

#Design:

##Cave Creation:

On running the game the user picks the size of the cave by entering a number N. This creates a cave NxN in size. N must be 10 to 20 in size.

The cave has rooms that scale with the size of the cave. The location of these rooms are picked randomly and the amount of each type is fixed on single number or percentage of how many rooms in the cave.

Entrance: Only 1 of the rooms must be an entrance/exit point. This is where the player controlled hero spawns and can choose to leave the cave to end it.

Wumpus: 15% of the rooms must spawn a Wumpus. (A monster your hero seeks to slay). So if you have 100 rooms, 15 of them will spawn a Wumpus.

Pit Trap: 5% of the rooms must be a pit trap. If you walk into this room you fall to your doom. (And the game is over)

Gold: 15% of the rooms must have a gold to loot.

Weapon: 15% of the rooms must have a weapon on the ground for the player to pick up to use for slaying monsters.

Empty: The remainder of rooms not assigned one of the above will be empty.

##Game Engine:

The game engine is an endless loop. It will display to the user basic info for the game and prompt for a single letter command. It will parse the command then refresh the basic info and continue to prompt for a move.

How the Game Ends:

* The hero leaves the cave by the entrance.
* The hero dies by moving into a pit trap room.
* The hero dies by moving into a room with a Wumpus without having picked up a weapon.
* The player chooses X to hard exit out of the game right of way.

***
The player scores points. The higher the points the better they do at the game. The following is the point system.

Point system:

* Explore an empty room not visited before: 1 point
* Find and Pickup a weapon: 5 points
* Find and kill a Wumpus: 10 points
* Find and loot gold: 5 points

***

Game Commands:

When prompted the following commands can be entered and causes an action for the player:
(Note: Case insensitive -- uppercase shown for easy to read)

* ? -- help to show this list of moves a player can make
* N -- move north 1 space - cannot move north if the cave ends (outside of grid)
* S -- move south 1 space - cannot move south if the cave ends (outside of grid)
* E -- move east 1 space - cannot move east if the cave ends (outside of grid)
* W -- moves west 1 space - cannot move west if the cave ends (outside of grid)
* L -- loot either gold or weapon in the room
* R -- run out of the cave entrance and head to the local inn to share your tale
* X -- this is a hard exit out of the game. The game ends with no points awarded.

##Environment Changes:

As the game progresses the cave changes based on the actions.

* Once a weapon is picked up all other weapon rooms turn into gold rooms.

* Entering a Wumpus room with a weapon that has been picked up instantly slays the Wumpus and turns that room into an empty explored room (only points for kill the Wumpus are given not points for exploring an empty room as well)

* Picking up a weapon/gold will turn that room into an empty explored room (only points for the items and not for exploring an empty room)

##Understanding Walls & Environment:

There are walls surrounding your cave. So for example if you pick N to be 10 you will have a 10x10 cave. But really the cave is 12x12 with the Border of the Cave being Walls. You cannot go in a direction that would put you into a wall. (This is not a game for mining) Trying to move into a wall will display an error describing how you bump into a wall or such and continue then to redisplay the current room you are in and prompt for another command.

As you move in the cave you will be given hints to nearby dangers (see below on output). If to the n, s, e, w of your position you are next ta Wumpus you will "Detect a Foul Stench in the Air". If to the n, s, e, w of your position you are next to a pit trap you will "Hear a howling wind".

There are no clues to being near an empty room, gold or weapons.

#Input & Output:

##Start of Game:
either pass the N size of the cave as a start up value, you can prompt for it, you can hard code it. Whatever you like but somehow you must set the N value of the cave.

##Status:

The program will give status to the user in the following format

(Ascii Display of surrounding rooms)

(Description of Room you are in)

(Environment Clues/Description)

[x Points Earned] You are (Weaponless/Armed).

Enter Move (? for help) >

Ascii Display

You will show the 8 rooms surrounding you. Use the following ASCII values to represent rooms as such.

* @ - the hero in the middle of the 9 rooms (8 surrounding and the one in the middle which you occupy)
* ? - unexplored room that could be empty, weapon, gold, wumpus or a pit trap
* . - explored/empty room
* # - wall showing the boundary of the cave
* ^ - Entrance to the cave where you can run out
* W - weapon in an explored weapon room that you did not bother to loot which would be odd. You can't beat a Wumpus Unarmed.
* $ - gold in an explored gold room that you did not bother to loot. Not looting this means you did not understand the goal of the game.

Examples:

You are in the upper left corner of the cave.

```
###
#@?
#.?
```

Just left the entrance and started to explore. Hey why did you leave that gold there?

```
^??
.@$
.??
```

You are not having luck finding anything right now

```
###
.@.
...
```

Description of Room:

Examples of how you might describe the rooms. Feel free to customize to your liking or humor.

Entrance Room -- you see see the entrance here. You wish to run away?

Empty Room -- you see nothing which is something

Pit trap -- aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaahhhhhhhhh noooooooooooooooooo *Splat*

Wumpus Room -- Overwhelmed in Stench a Wumpus stands before you ready to eat you.

Gold Room - before you lies the the gold of adventure seekers who feed a Wumpus Recently

Weapon Room - Cast before you in a rock a sword awaits to be looted and name yourself King.

Environmental Clues/Description:

This is giving you clues to nearby threats as well as describing any battles if you enter a room with a Wumpus and you are armed.

If next to a pit room you see a message like "Howling Winds Fill the Room"
If next to a Wumpus room you see a message like "A fowl Stench fills the room"
If you enter a room with a wumpus you describe if you kill it or you get eaten based on if you have a weapon or not.
If you enter a pit trap room - have fun describing how one falls before showing the game over.

***

So putting it all together you might see these screen shots

```
###
#@?
#.?
Empty Room - there is nothing here but air.
You hear howling winds.
[10 points earned] You are weaponless.
Enter Move (? for help) >


###
.@.
...
Empty Room - there is nothing here but air.
[23 points earned] You are armed and dangerous.
Enter Move (? for help) >
```

##End of Game Message:

When the game ends due to the conditions display why the game is over. Say the game is over and show the final points.

Examples:

Say you find a wumpus unarmed.

```
A Wumpus attacks you and makes you his lunch.
***GAME OVER***
You scored 24 Points!
```

Say you find that pit trap:

```
You fall to your death. Your screams are heard by no one.
***GAME OVER***
You scored 1 whole point!
```

Say you exit out of the dungeon

```
You exit the Wumpus cave and run to town. People buy you ales as you tell the story of your adventure.
***GAME OVER***
You scored 120 points! Well Played!
```

#**Notes:**

I have done what I can to layout the challenge with a very large design requirement. There will be potential for holes or missing elements in the design or things I perhaps did not address in the design. Please find a suitable solution that fits your desire and implementation and consider this part of the challenge. However if you wish to ask questions about the design or point out obvious things missing from the design, please comment and I can make adjustments.

Be creative. There are lots of strings for feedback or descriptions. Come up with your own or perhaps find a way to do random strings to keep the game fresh and unique. Add other features or monsters or whatever. This design for the challenge is much like the pirate code - it is just a bunch of guidelines for you to bend to your need and liking.

Remember to add Error messages. If you loot an empty cave or move to a direction towards a wall you must display what happens and then either redisplay the whole status or just the prompt for a move. Up to you to decide.

This hard challenges builds on skills learned in doing easy and intermediate challenges. The difficulty comes from following a larger design than normal and putting it all together to make a very fun game. Have fun and enjoy the challenge!

Title: [21/3/14] Challenge #153 [Hard] Interpreting interpreters
Text: #**Description:**

An interpreter is a program that executes commands written in a programming language. Today you will be writing 2 of these!

Taking a language of your choice, write an interpreter and within that language, write a Brainfuck interpreter.

For instance;

Let's say your programming language of choice is Ruby. Your languages could be linked like so:

Ruby -> Scheme -> Brainfuck

Ruby parses and evaluates the Scheme syntax. The Scheme syntax will parse the Brainfuck syntax.

I chose Scheme as an example here because there is a lot of reading material on building an interpreter for Scheme.

#**Input**

You will be given Brainfuck code, within your program, convert this code back to its string equivalent.

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

#**Output**

```
    Hello World!
```

#**Challenge Input**

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

#**Bonus:**

For extra points, have your chain add an extra language.
E.g.

Ruby -> Scheme -> Brainfuck -> Whitespace

(Only the mentally ill would attempt such a feat.)

#**Further Reading**

[Structure and Interpretation of Computer Programs](http://mitpress.mit.edu/sicp/full-text/book/book-Z-H-4.html#%_toc_start)

This book will serve you extremely well. Large portions of the book are on interpreters/compilers and its main dialect is Scheme.

[AWIB](https://code.google.com/p/awib/)

This is a Brainfuck compiler written in Brainfuck. Potentially very useful to poke around and see how it works.

[Lispy](http://norvig.com/lispy.html)

A Lisp interpreter written in Python

Title: [14/04/2014] Challenge #152 [Hard] Minimum Spanning Tree
Text: # [](#HardIcon) _(Hard)_: Minimum Spanning Tree

First, a bit of back story. In graph theory, a graph is a set of points called *vertices*, joined up by lines or *edges*. Those edges can have a number called *weight* associated with them, which would represent distance, cost, or whatever you like. It's an abstract idea and is usually used for modeling real-world situations such as a neighborhood, a network of computers or a set of steps. A **spanning tree** is a subgraph (a graph deriving from another one) that connects *all* of the vertices of the parent graph.
This means several things:

* A spanning tree must contain every vertex of G - but not necessarily every edge.
* Because it's a tree, it must not have any loops or cycles - that is, it must have no closed shapes within it.
* You must only use edges from the original graph to form the spanning tree.

* The tree must be *connected*. This means all the edges must be joined in some way. This is illustrated with an example below.

Here are some examples to illustrate this concept. Take this [graph G](http://i.imgur.com/RIfsghM.png).
Here is [one possible spanning tree](http://i.imgur.com/yf8K1AK.png). Note there may be (and probably will be) more than one valid spanning tree for a given graph. Here are some examples of invalid spanning trees, for several reasons:

* [This one leaves out vertices C and E.](http://i.imgur.com/6CVjxpF.png)
* [This one contains a cycle so it's not a tree.](http://i.imgur.com/cibmve1.png)
* [This one is not fully connected - there is no path from B to F via this spanning tree.](http://i.imgur.com/eanfUzf.png)
* [This one uses an edge that is not in the original graph.](http://i.imgur.com/WMSDZf8.png)

Because representing graphs visually like this makes it complicated to do computations with them, you can represent graphs as a matrix instead, such as [this one](http://i.imgur.com/iXuaqNT.png). This is called a distance matrix because it shows you the distances between any two vertices - like those distance charts you find at the back of diaries. (These are *very* similar to adjacency matrices, except they show the weight of the connecting edges rather than just its existence.) Note how it is symmetric along the diagonal. A dash (-) means there is no edge connecting those two vertices.

Your challenge is, given any (non-directional) graph in matrix form as shown above, to find the **minimum spanning tree**. This is the spanning tree with the smallest possible sum distance of its edges. There may be more than one minimum spanning tree for any given tree. For example a minimum spanning tree for Graph G shown above is [here](http://i.imgur.com/RrXzZZY.png).

# Formal Inputs and Outputs

## Input Description

On the console, you will be given a number V. This will be between 1 and 26 inclusive, and represents the number of vertices in the graph.
You will then be given a distance matrix, with newlines separating rows and commas separating columns. **-1** is used to denote that there is no edge connecting those two vertices. For the sake of simplicity, the vertices in the graph are assumed to be named A, B, C, D and so on, with the matrix representing them in that order, left-to-right and top-to-bottom (like in the distance matrix example shown previously.)

## Output Description

You must print out the total weight of the MST, and then the edges of the minimum spanning tree represented by the two vertices such as DF, AE. They do not need to be in any particular order.

# Sample Inputs & Outputs

## Sample Input

```
8
-1,11,9,6,-1,-1,-1,-1
11,-1,-1,5,7,-1,-1,-1
9,-1,-1,12,-1,6,-1,-1
6,5,12,-1,4,3,7,-1
-1,7,-1,4,-1,-1,2,-1
-1,-1,6,3,-1,-1,8,10
-1,-1,-1,7,2,8,-1,6
-1,-1,-1,-1,-1,10,6,-1
```

## Sample Output

```
32
AD,DF,DE,EG,DB,GH,FC
```

# Challenge

## Challenge Input

(this input represents [this graph](http://i.imgur.com/ef5kdbx.png))

```
10
-1,29,-1,-1,18,39,-1,-1,-1,-1
29,-1,37,-1,-1,20,-1,-1,-1,-1
-1,37,-1,28,-1,43,47,-1,-1,-1
-1,-1,28,-1,-1,-1,35,-1,-1,-1
18,-1,-1,-1,-1,31,-1,36,-1,-1
39,20,43,-1,31,-1,34,-1,33,-1
-1,-1,47,35,-1,34,-1,-1,-1,22
-1,-1,-1,-1,36,-1,-1,-1,14,-1
-1,-1,-1,-1,-1,33,-1,14,-1,23
-1,-1,-1,-1,-1,-1,22,-1,23,-1
```

# Notes

There are algorithms to find the MST for a given graph, such as the [reverse-delete algorithm](http://en.wikipedia.org/wiki/Reverse-Delete_algorithm) or [Kruskal's method](http://en.wikipedia.org/wiki/Kruskal%27s_algorithm). The submitted solution does not have to work with directed graphs - the edges will always be bidirectional and thus the matrix will always be symmetrical.

Title: [02/28/14] Challenge #151 [Hard] Re-emvoweler 2
Text: # _(Hard)_: Re-emvoweler 2

[This week's Intermediate challenge](http://www.reddit.com/r/dailyprogrammer/comments/1yzlde/022614_challenge_150_intermediate_reemvoweler_1/) involved re-emvoweling, which means reconstructing a series of words given their consonants and vowels separately, in order.

For most inputs, there are a huge number of valid ways to re-emvowel them. Your goal this time is to produce a valid output that also resembles an English sentence, as much as possible. For each sample input, there is one best answer (the sentence I actually used to produce the input). Good solutions that don't produce the best answer are acceptable, but you should aim for answers that are significantly better than random.

A typical strategy is to begin by analyzing a corpus of English text for word frequencies or other patterns. You can use whatever techniques or data sources you wish, but your program must run completely autonomously, without relying on human judgment during runtime.

The challenge inputs this time are all based on English sentences taken from the web. The word "a" _does_ appear in these sentences. Other than the word "a", all words in the sentences appear in [the Enable word list](http://code.google.com/p/dotnetperls-controls/downloads/detail?name=enable1.txt), and none of the words contain punctuation.

Along with your solution, be sure to post your results from running the challenge inputs!

# Formal Inputs & Outputs

## Input description

Two strings, one containing only consonants, and one containing only vowels. There are no spaces in the input.

## 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. The output should resemble an English sentence (without capitalization and punctuation) as closely as possible.

# Sample Inputs & Outputs

## Sample Input

```
    thffcrrprtdthtblckdndcrfwhdbnrdrd
    eoieeoeaaoaeaaueaeeoee
```

## Sample Output

```
    the officer reported that a blockade and a curfew had been ordered
```

# Challenge Inputs

## Challenge Input 1

```
    thdcryptntmsbltdtrmnthtthplnsrnddfrtypftrnsprt
    eeioeaiaeoeeieaeaaeieeoaeoao
```

## Challenge Input 2

```
    nfcthblvdthrwsnthrcncptytbyndhmnndrstndngdtthmrvlscmplxtyndthclckwrkprcsnfthnvrs
    iaeeieeeeaaoeoeeeouaueaiueoeaeouoeiaeooeiiooeuiee
```

## Challenge Input 3

```
thhmrthpthsthtnsnvnthblmngsndtrckllcnsprtnsrthtthtlftngrtrvlngbckthrtyyrstnsrhsprntsmtndltmtlymtcngvntsvntgnwbfrlydscrbdsclss
c
    euoeaoeeioeeeooiouaaoieoeueaeaeoaeeaeaeiaieaoeueiaeeeauiaeaeaieiiaeoeaieieaaai
```

Title: [12/23/13] Challenge #130 [Hard] Coloring France's Departments
Text: # [](#HardIcon) *(Hard)*: Coloring France's Departments

The European country of [France](http://en.wikipedia.org/wiki/France) is segmented into many different [departments](http://en.wikipedia.org/wiki/Departments_of_France); 96 in the main continent with a few others overseas. Wikipedia, as always, has a great [visualization of these departments with their respective unique numbers here](http://upload.wikimedia.org/wikipedia/commons/b/b2/D%C3%A9partements_de_France_English.svg).

Some departments, like 66 (Pyrénées-Orientales), are only bordered by two other departments. Others, like department 87, are surrounded by much more (6 in this example). Your goal is to color a map of these regions with two requirements: 1) make sure that each adjacent department do not share a color, so you can clearly distinguish each department, and 2) minimize these numbers of colors.

The input will be a variation of the list of French departments, represented as an [adjacency list](http://en.wikipedia.org/wiki/Adjacency_list). This challenge is essentially solving for [Graph coloring](http://en.wikipedia.org/wiki/Graph_coloring), where you must print each department's color (a unique integer).

# Formal Inputs & Outputs
## Input Description

On standard console input, you will be given an integer N which represents the following N-lines of an adjacency list. These lines of data will always be in the format of integers A B C D ... where A is the source node / vertex that points to vertices B C D... etc. Remember that this data really means that A is the ID of a department, and B C D ... are the bordering departments.

Writing up the French department list as an adjacency list is very tedious; feel free to only work on a subset.

## Output Description

For each given node (a department), print the unique color identifier after it. A color identifier is unique integer, starting from 0, that represents a unique color. Remember that bordering departments (e.g. adjacent nodes) cannot have the same color index!

# Sample Inputs & Outputs
## Sample Input

Note that this list only contains 8 departments from the south-western corner of France as an example

```
    8
    64 40 32 65
    65 64 32 31
    31 65 32 82 81 11 9
    9 31 11 66
    66 9 11
    40 33 47 32 64
    32 40 47 82 31 65 64
    11 31 81 34 66 9
```

## Sample Output

```
    64 0
    65 1
    31 0
    9 1
    66 0
    40 1
    32 2
    11 2
```

# Challenge++:

If you want to go above and beyond for this challenge, programmatically draw a map of the French departments with actual colors from your unique set (you may randomly pick them or use a [color palette](http://en.wikipedia.org/wiki/Palette_(computing\))). Feel free to use the linked SVG file from Wikipedia, since it can be modified through text / XML manipulation.

Title: [11/15/13] Challenge #129 [Hard] Baking Pi
Text: # [](#HardIcon) *(Hard)*: Baking Pi

[Pi (π)](http://en.wikipedia.org/wiki/Pi), the super-cool irrational number, can be computed through a variety of ways. One way is using [continued fractions](http://en.wikipedia.org/wiki/Continued_fraction), which computes a more and more precise value of Pi over each iteration. The problem with this approach is you cannot distribute the work across several machines, because the computation itself cannot be split into smaller independent computations.

This is where [Spigot Algorithms](http://en.wikipedia.org/wiki/Spigot_algorithm) shine: you can compute the subset of base-16 digits for some constants (like Pi) independently across many machines. More specifically, the [Bailey–Borwein–Plouffe formula](http://en.wikipedia.org/wiki/Bailey%E2%80%93Borwein%E2%80%93Plouffe_formula) allows you to compute digits of Pi without having to compute any preceding digits.

Your goal is to write an application that computes base-16 digits of Pi across multiple machines over the Internet. You will have to implement a master dispatcher service, that sends out work commands and receives results, to networked machines that implement the BBP formula. The implementation details are up to you, including the network protocol and platform you want to write for. You must support at minimum four (4) machines computing in parallel, though for demonstration purposes you may run the processes locally on one machine.

Bonus: As an extra bonus challenge, implement a verification feature (e.g. machine 1 computes a digit, then machines 2 & 3 verify it is valid).

Reddit Gold: To thank Reddit for hosting such an awesome community, the first two valid solutions posted will win one-month of [Reddit Gold](http://www.reddit.com/gold/about).

Special Thanks: Special thanks to Daily Programmer's Elite6809 for clarifying why this challenge must compute in base-16 (or in a base that's a power of two), and not in the original description's base-10. The challenge text has been updated to reflect the fix.

# Formal Inputs & Outputs
## Input Description

There is no formal input description, though this is the desired behavior:

You launch your main dispatcher-application on Computer A. You then launch the computing-applications on Computer W, X, Y and Z. Computer A wants to compute the first four digits of Pi, and sends the appropriate network commands, one to each computer. Computer Y returns a result first, so the dispatcher receives the data, saves in your output file the line "0:2", and then gives the command to compute the 5th digit of Pi. Computers X, Z, W finish in that order, returning the results to the dispatcher, which in turn saves in the same format. They are then asked to compute digits 6, 7, and 8. This repeats until your dispatcher application sends a "stop" or "kill" command to the computing processes. It is up to you how many hexadecimal digits each process computes.

## Output Description

For each computed base-16 (hexadecimal) digit of Pi, write to a file a line of text in the format of <Digit-Index>:<Computed-Digit>. The order does not matter, and you may skip digits. An example of the file, after eight computed digits, would be as follows:

```
0:2
1:4
2:3
3:F
4:6
5:A
6:8
7:8
```

# Notes:

There are a few sites that already have large numbers of hexadecimal digits of Pi pre-computed; make sure to use them as a resource to validate your work! [Here's a great example on CalcCrypto](http://calccrypto.wikidot.com/math:pi-hex). Remember that these digits are in the Mantissa, so you compute the decimal values with negative exponent. As an example, the first 8 binary digits of Pi's Significand is "00100100". What would the decimal value be? Use the algebraic conversion formula:

```
0 * 2^-1 + 0 * 2^-2 + 1 * 2^-3 + 0 * 2^-4 + 0 * 2^-5 + 1 * 2^-6 + 0 * 2^-7 + 0 * 2^-8
```

The result here is "0.140625" (decimal), where the first two digits in the Significant are correct! We can apply the same for the above computed hex digits: (note how F was changed to 15 and A changed to 10, as appropriate)

```
2 * 16^-1 + 4 * 16^-2 + 3 * 16^-3 + 15 * 16^-4 + 6 * 16^-5 + 10 * 16^-6 + 8 * 16^-7 + 8 * 16^-8
```

The result is "0.14159265346" (decimal), 9 decimal digits matching with Pi!

Title: [11/8/13] Challenge #128 [Hard] Mon Petit Fourier
Text: # [](#HardIcon) *(Hard)*: Mon Petit Fourier

[Fast Fourier Transform](http://en.wikipedia.org/wiki/Fast_Fourier_transform) is an extremely powerful algorithm. Fourier transforms can convert a signal (used loosely here) between its time / space domain to its frequency domain. A less-technical explanation is that it's an algorithm that can take the signal made from a sum of sinusoidal waves, and returns a set of simple sinusoidal wave functions that, if summed, closely match the original signal given. [This YouTube video](http://www.youtube.com/watch?v=ObklYbQaX24) does a great rundown on what an FFT is.

Here's a simple example: imagine you are given a series of 100 plot points, where these points are on the plot formed by the function f(t) = 3 x sin(2 x t) + 0.5 x cos(10 x t). Assume that you're *only* given these points and not the equation, yet you still want to derive / find what the original equation was. Using an FFT, you can compute the original set of equation's sinusoidal frequency and amplitude, giving you back the original signal-components!

FFT is often used in data compression algorithms: JPEG, the image format, uses an FFT-like algorithm to take discrete color location information and creates generalized signals from them. If you're interested in the details, JPEG uses the [DCT algorithm](http://en.wikipedia.org/wiki/Discrete_cosine_transform) rather than FFT.

Your goal is to write a simple FFT tool that takes a set of 100 points from an unknown signal, and then decomposes them into the discrete frequency domain. Once your sinusoidal functions are generated from this original signal, print the percentage-difference between what you compute vs. the original signal.

# Formal Inputs & Outputs
## Input Description

You will be given 200 floating-point numbers, two space-delimited values on each line (so 100 lines total). The first number is the x-position of the point, with the second being the y-position. These points are guaranteed to be on a graph only composed of the sum of at most 8 sinusoidal functions.

## Output Description

After applying the FFT algorithm on these points, print the set of sinusoidal equations you believe are the original signal's components. Then, print the percentage difference between your results versus the given data.

# Sample Inputs & Outputs
## Sample Input

```
0 0.5
0.1 0.866159145
0.2 0.960181609
0.3 1.198931172
0.4 1.825246462
0.5 2.666244047
0.6 3.276202401
0.7 3.333300317
0.8 2.925970792
0.9 2.465977762
1 2.308356516
1.1 2.42770206
1.2 2.448316521
1.3 2.000227506
1.4 1.07333306
1.5 0.043516068
1.6 -0.65395217
1.7 -0.904204975
1.8 -0.997402976
1.9 -1.341221364
2 -2.066366455
```

2.1	-2.888591947
2.2	-3.354786635
2.3	-3.247489521
2.4	-2.776404323
2.5	-2.381171418
2.6	-2.326904306
2.7	-2.464362867
2.8	-2.375102847
2.9	-1.767835303
3	-0.76112077
3.1	0.20810297
3.2	0.766759295
3.3	0.927985717
3.4	1.058054916
3.5	1.519113694
3.6	2.317021747
3.7	3.078831313
3.8	3.381295838
3.9	3.128951502
4	2.634605709
4.1	2.328522031
4.2	2.363804067
4.3	2.480747944
4.4	2.254673233
4.5	1.49901645
4.6	0.45258077
4.7	-0.421841458
4.8	-0.843052513
4.9	-0.949141116
5	-1.149580318
5.1	-1.728546964
5.2	-2.564974798
5.3	-3.227467658
5.4	-3.357463607
5.5	-2.988907242
5.6	-2.510923134
5.7	-2.308052164
5.8	-2.408895717
5.9	-2.466115366
6	-2.085925244
6.1	-1.203738665
6.2	-0.160058945
6.3	0.593817432
6.4	0.890458091
6.5	0.979274185
6.6	1.276396816
6.7	1.96224277
6.8	2.797556956
6.9	3.327782198
7	3.288481669
7.1	2.839568594
7.2	2.413348036
7.3	2.316277157
7.4	2.450614873
7.5	2.411739155
7.6	1.871361732
7.7	0.893867554
7.8	-0.10564059

```
    7.9 -0.723706024
    8 -0.918903572
    8.1 -1.028922968
    8.2 -1.439481198
    8.3 -2.210286177
    8.4 -3.002712849
    8.5 -3.376380797
    8.6 -3.182549421
    8.7 -2.693102974
    8.8 -2.346846852
    8.9 -2.346518016
    9 -2.476998548
    9.1 -2.311682198
    9.2 -1.61691909
    9.3 -0.582206634
    9.4 0.336122761
    9.5 0.814718409
    9.6 0.939729562
    9.7 1.106623527
    9.8 1.636246738
    9.9 2.460931653
```

## Sample Output

```
    3 * SIN(2 * X)
    0.5 * COS(10 * X)
    0% Deviation
```

Title: [09/13/13] Challenge #127 [Hard] Language Detection
Text: # [](#HardIcon) *(Hard)*: Language Detection

You are part of the newly formed ILU team, whose acronym spells Internet Language Usage. Your goal is to help write part of a web-crawler that detects which language a wep-page / document has been written in. The good news is you only have to support detection of five languages (English, Spanish, French, German, and Portuguese), though the bad news is the text input has been stripped to just space-delimited words. These languages have hundreds of thousands of words each, some growing at a rate of [~25,000 new words a year](http://en.wikipedia.org/wiki/English_language#Number_of_words_in_English)! These languages also share many words, called [cognates](http://en.wikipedia.org/wiki/Cognate). An example would be the French-English word "lance", both meaning a spear / javelin-like weapon.

You are allowed to use whatever resources you have, except for existing language-detection tools. I recommend using the [WinEdt dictionary set](http://www.winedt.org/Dict/) as a starting point for the five languages.

The more consistently correct you are, the most correct your solution is considered.

# Formal Inputs & Outputs
## Input Description

You will be give a large lower-case space-delimited non-punctuated string that has a series of words (they may or may not form a grammatically correct). The string will be unicode, to support accents in all of the five languages (except English). Note that a string of a certain language may make references to nouns in their own respective language. As an example, the sample input is in French, but references the American publication "The Hollywood Reporter" and the state "California".

## Output Description

Given the input, you must attempt to detect the language the text was written in, printing your top guesses. At minimum you must print your top guess; if your code is not certain of the language, you may print your ordered "best guesses".

# Sample Inputs & Outputs
## Sample Input 0

l'école a été classé meilleure école de cinéma d'europe par la revue professionnelle de référence the hollywood reporter et 7e meilleure école de cinéma du monde juste derrière le california institute of the arts et devant l'université columbia

## Sample Output 0

French
English

## Sample Input 1

few things are harder to put up with than the annoyance of a good example

## Sample Output 1

English

Title: [07/12/13] Challenge #126 [Hard] Not-So-Normal Triangle Search
Text: # [](#HardIcon) *(Hard)*: Not-So-Normal Triangle Search

A three-dimensional triangle can be defined with three points in 3D space: one for each corner. One can compute the [surface-normal](http://en.wikipedia.org/wiki/Normal_(geometry\)) of this triangle by using the three points to compute the [cross-product](http://en.wikipedia.org/wiki/Cross_product).

You will be given a set of N points, such that N is greater than or equal to 3. Your goal is to find the maximum set of non-intersecting triangles that can be constructed with these N points (points may be shared between triangles) such that this set's average surface normal is as close to the given vector's direction as possible.

"Closeness" between the average surface normal and target vector is defined as minimizing for the smallest angle between the two (as computed through the [dot-product](http://en.wikipedia.org/wiki/Dot_product)). Though each triangle has two surface normals (one for each of the two sides), we don't care about which one you choose: just make sure that when printing the results, you respect the [right-hand rule](http://en.wikipedia.org/wiki/Right-hand_rule) for consistency. At **minimum**, this set must match the target vector with less than 10 degrees of difference.

Original author: /u/nint22. This challenge is a little more math-heavy than usual, but don't worry: the math isn't hard, and Wikipedia has all the formulas you'll need. Triangle-triangle intersection will be the most tricky part!

# Formal Inputs & Outputs
## Input Description

You will be given an integer N which represents the N-following lines, each being a 3D point in space. Each line has three [Real-numbers](https://en.wikipedia.org/wiki/Real_number) that are space -delimited. The last line, which will be line N+1, is the target vector that you are trying to align-with: it is also represented as three space-delimited Real-numbers.

## Output Description

Find the largest set of triangles whose average surface normals match the target vector direction within at minimum 10 degrees. Print the result as one triangle per line, where a triangle is defined as the three point indices used. If no set is found, print "No valid result found".

# Sample Inputs & Outputs
## Sample Input

```
5
0.6652 -0.1405 0.7143
0.2223 0.3001 0.7125
-0.9931 0.9613 0.0669
0.0665 0.6426 -0.4931
-0.1100 -0.3525 0.3548
0.577 -0.577 0.577
```

## Sample Output

The author is still working on a solution to generate some results with; first person to post good demo data gets a +1 gold medal! The following results are "bad"/"faked", and are only examples of "valid output format".

```
0 1 2
1 4 2
```

Title: [07/03/13] Challenge #125 [Hard] Robo Room Service
Text: # [](#HardIcon) *(Hard)*: Robo Room Service

You are the lead software engineer hired by a major hotel chain to program a new path-planning system for an automated room-service robot! You read that right: you are helping build a robot that will execute some basic tasks, such as moving around hotel laundry or patrol for security. The problem though is that your path-planning system is based on a [graph](http://en.wikipedia.org/wiki/Graph_(abstract_data_type\)), whereas the only data you have about the hotel's layout is in an ASCII-map!

Your goal is to convert a given ASCII-map (a big 2D array of valid spaces the robot can move to) into a graph data-structure. You must minimize the room count (generate as little rooms as possible), thus coalescing adjacent structures that have the same room-type. The resulting graph should have one node per room, with an edge between nodes representing the connection of an adjacent room.

Original author: /u/nint22. I'm posting this challenge as "hard", though it is more correctly somewhere between "intermediate" and "hard". This challenge was inspired by the Dwarf Fortress path-planner.

# Formal Inputs & Outputs
## Input Description

You will be given an integer W and H on standard input, which represents the the **W**idth and **H**eight of the ASCII map that will follow: this map is just a 2D array of ASCII digit characters ('0' to '9'). The digit '0' (zero) represents a non-accessible area of the hotel's layout, while any other digit represent a room. Different digits represent different room-types. Rooms are "connected" if they are directly-adjacent. A room is defined as any rectangular shape.

## Output Description

You must convert the above-described ASCII-map input into a graph of nodes (rooms) and edges (connections between rooms). You should do this as optimally as possible, meaning you should be generating as little rooms (nodes) as possible. With this resulting graph data-structure, you must print an [adjacency list](http://en.wikipedia.org/wiki/Edge_list). For each node N you have, assign it a unique number, and then (on the same line) print all connected rooms with their own respective unique number. Remember: room numbers do not map to room-types, meaning with some test data you could generate 10 rooms, but they are all of type 1. (Sample input 2 shows this example)

Note that the output has some ambiguity: the same map may produce multiple graphs that have the same overall structure. Don't worry about this, and just focus on printing the correct edge relationships (it is why we're asking you to print unique node numbers, not what the nodes actually associate to).

# Sample Inputs & Outputs
## Sample Input 1

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

## Sample Output 1

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

## Sample Input 2

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

## Sample Output 2

[Image explanation](http://i.imgur.com/oo7EWzT.png)

```
    1 4
    2 4
    3 4
    4 1 2 3 5
    5 4 6
    6 5 7 8 9
    7 6
    8 6
    9 6
```

Title: [06/1/13] Challenge #124 [Hard] Power-Processor Management
Text: # [](#HardIcon) *(Hard)*: Power-Processor Management

A co-worker has just finished designing and fabricating an incredibly powerful new processor architecture: this processor allows you to vary how fast you execute code, but in turn vary how much energy you consume. Your goal is to write a power-focused process scheduling system that minimizes both time and maximum processor speed for the given work.

The processor executes a set of programs, where each program Pn (where n is the program ID as an integer, so P0, P1, P2 are all valid programs) has the amount of work W (as an integer) to be done within a time interval of R (as an integer) and D (as an integer). One unit of time at one rate of power consumption does one unit of work: if you increase work done, the same increase is applied to power consumption. Time can be treated like seconds, thus one second of simulation at the rate of 10x power consumption, you can accomplish 10 units of work.

Note that the time intervals must be strictly enforced: you may not load a process until it is simulation-time R for the respective process. The end-time of a process must be before or on simulation-time D. This architecture allows process switching, thus you do not have to accomplish all work continuously. Process switching may occur at any point in time and occurs instantaneously. You may change work-speed (and thus power-consumption) only at the start of a new execution (so every time you swap a process).

Author: nint22, with the base idea from challenge #4254 ACM Competitive Collegiate Programming challenges repository.

# Formal Inputs & Outputs
## Input Description

You must read in, from standard console input, an integer T for the number of test cases. You should expect, for each test case, an integer N for the number of given programs you must execute. For each program, you will be given an integer an integer R for the start time, then (space-delimited) an integer D for end time, and then (space-delimited) an integer W for the amount of work. All input will be guaranteed well formed.

## Output Description

For each test-case, you must print how much simulation time it took to accomplish all given work, and the maximum work/power ratio set, both as integers and space-delimited.

# Sample Inputs & Outputs

## Sample Input

[The following inputs is visualized here in their **solution form**.](http://i.imgur.com/aL80vsb.png)

```
    1
    5
    1 4 2
    3 6 3
    4 5 2
    4 7 2
    5 8 1
```

## Sample Output

```
    8 5
```

# Note
"Minimize for both time and maximum power rate" is a weak definition, since you could end up in a situation where one or the other is absurdly optimized (we could do almost all work as fast as possible if we let the power rate be infinite...). So, for the sake of making this reasonable, we define "minimize for both.." with the constraint that *both* numbers should be as low as possible, even if that means they are local minima, and there is a significantly lower value for either one.

Title: [05/24/13] Challenge #123 [Hard] Snake-Fill
Text:
[](#HardIcon) *(Hard)*: Snake-Fill
The snake-fill algorithm is a "fictional" algorithm where you must fill a given 2D board, with some minimal obstacles, with a "snake". This "snake" always starts in the top-left corner and can move in any directly-adjacent direction (north, east, south, west) one step at a time. This snake is also infinitely long: once it has moved over a tile on the board, the tile is "filled" with the snakes body. A snake cannot revisit a tile: it is unable to traverse a tile that it has already traversed. Essentially this is the same logic that controls a snake during a [game of snake](http://en.wikipedia.org/wiki/Snake_(video_game\)).

Your goal is to take a board definition, as described below, and then attempt to fill it as best as possible with a snake's body while respecting the snake's movement rules!

Author: nint22
# Formal Inputs & Outputs
## Input Description
You will be first given two integers on a single line through standard input. They represent the width and height, respectively, of the board you are to attempt to fill. On the next line, you will be given an integer N, which represents the following N lines of obstacle definitions. Obstacles are pairs of integers that represent the X and Y coordinate, respectively, of an impassable (blocked) tile. Any impassable block does not allow snake movement over it. Note that the origin (0, 0) is in the top-left of the board, and positive X grows to the right while positive Y grows to the bottom. Thus, the biggest valid coordinate would be (Width - 1, Height - 1).
## Output Description
First, print the number of tiles filled and then the number of tiles total: do *not* count occluded (blocked) tiles. Remember, the more tiles filled by your snake, the more correct your solution is. Then, for each movement your snake has done in its attempt to fill the board, print the position is has moved to. This has to be listed in correct and logical order: one should be able to verify your solution by just running through this data again.
# Sample Inputs & Outputs
## Sample Input
[The following inputs generates this board configuration](http://i.imgur.com/WclGAwX.png). Note that the darker blocks are occluded (blocked) tiles.

```
10 10
5
3 0
3 1
3 2
4 1
5 1
```
## Sample Output
Note: The following is dummy-data: it is NOT the correct solution from the above sample input. Blame nint22: he is a gentlemen whom is short on time.

```
95 / 95
0 0
0 1
1 1
... (See note)
```
# Challenge Input
None given
## Challenge Input Solution
None given
# Note
As per usual: this challenge may seem easy, but is quite complex as the movement rules limit any sort of "tricks" one could do for optimizations. Anyone who does some sort of graphical **and** animated version of their results get a +1 silver for their flair!

Title: [05/10/13] Challenge #123 [Hard] Robot Jousting
Text: # [](#HardIcon) *(Hard)*: Robot Jousting

You are an expert in the new and exciting field of *Robot Jousting*! Yes, you read that right: robots that charge one another to see who wins and who gets destroyed. Your job has been to work on a simulation of the joust matches and compute *when* there is a collision between the two robots and *which* robot would win (the robot with the higher velocity), thus preventing the destruction of very expensive hardware.

Let's define the actual behavior of the jousting event and how the robots work: the event takes place in a long hallway. Robots are placed initially in the center on the far left or far right of the hallway. When robots start, they choose a given starting angle, and keep moving forward until they hit a wall. Once a robot hits a wall, they stop movement, and rotate back to the angle in which they came into the wall. Basically robots "reflect" themselves off the wall at the angle in which they hit it. For every wall-hit event, the robot loses 10% of its speed, thus robots will slow down over time (but never stop until there is a collision).

[Check out these two images as examples of the described scene](http://imgur.com/a/NSzpY). Note that the actual robot geometry you want to simulate is a perfect circle, where the radius is 0.25 meters, or 25 centimeters.

# Formal Inputs & Outputs
## Input Description

You will be given three separate lines of information: the first has data describing the hallway that the robots will joust in, and then the second and third represent information on the left and right robots, respectively.

The first line will contain two integers: how long and wide the hallway is in meters. As an example, given the line "10 2", then you should know that the length of the hallway is 10 meters, while the width is just 2 meters.

The second and third lines also contain two integers: the first is the initial angle the robot will move towards (in degrees, as a signed number, where degree 0 always points to the center of the hallway, negative points to the left, and positive points to the right). The second integer is the speed that the robot will initially move at, as defined in millimeters per second. As an example, given the two lines "45 5" and "-45 2", we know that the left robot will launch at 45 degrees to its left, and that the second robot will launch 45 degrees to its left (really try to understand the angle standard we use). The left robot starts with an initial speed of 5 mm/s with the right robot starting at 2 mm/s. Assume that the robot radius will always be a quarter of a meter (25 centimeters).

## Output Description

Simply print "Left robot wins at X seconds." or "Right robot wins at X seconds." whenever the robots collide: make sure that the variable X is the number of seconds elapsed since start, and that the winning robot is whichever robot had the higher velocity. In case the robots never hit each other during a simulation, simply print "No winner found".

# Sample Inputs & Outputs
## Sample Input

```
10 2
30 5
-10 4
```

## Sample Output

Please note that this is FAKE data; I've yet to write my own simulation...

```
Left robot wins at 32.5 seconds.
```

# Challenge Note

Make sure to keep your simulation as precise as possible! Any cool tricks with a focus on precision management will get bonus awards! This is also a very open-ended challenge question, so feel free to ask question and discuss in the comments section.

Title: [05/10/13] Challenge #122 [Hard] Subset Sum Insanity
Text: # [](#HardIcon) *(Hard)*: Subset Sum

The [subset sum](http://en.wikipedia.org/wiki/Subset_sum_problem) problem is a classic computer science challenge: though it may appear trivial on its surface, there is no known solution that runs in [deterministic polynomial time](http://en.wikipedia.org/wiki/P_(complexity)) (basically this is an [NP-complete](http://en.wikipedia.org/wiki/Subset_sum_problem) problem). To make this challenge more "fun" (in the same way that losing in Dwarf Fortress is "fun"), we will be solving this problem in a three-dimensional matrix and define a subset as a set of integers that are directly adjacent!

Don't forget our [previous week-long](http://www.reddit.com/r/dailyprogrammer/comments/1dk7c7/05213_challenge_121_hard_medal_management/) [Hard] challenge competition ends today!

# Formal Inputs & Outputs
## Input Description

You will be given three integers `(U, V, W)` on the first line of data, where each is the length of the matrices' respective dimensions (meaning U is the number of elements in the X dimension, V is the number of elements in the Y dimension, and W is the number of elements in the Z dimension). After the initial line of input, you will be given a series of space-delimited integers that makes up the 3D matrix. Integers are ordered first in the X dimension, then Y, and then Z ([the coordinate system is clarified here](http://i.imgur.com/nxChpUZ.png)).

## Output Description

Simply print all sets of integers that sum to 0, if this set is of directly-adjacent integers (meaning a set that travels vertically or horizontally, but never diagonally). If there are no such sets, simply print "No subsets sum to 0".

# Sample Inputs & Outputs
## Sample Input

```
2 2 3
-1 2 3 4 1 3 4 5 4 6 8 10
```

## Sample Output

```
-1 1
```

Note: This is set of positions (0, 0, 0), and (0, 0, 1).

# Challenge Input

```
    8 8 8
    -7 0 -10 -4 -1 -9 4 3 -9 -1 2 4 -6 3 3 -9 9 0 -7 3 -7 -10 -9 4 -6 1 5 -1 -8 9 1 -9 6 -1 1 -8 -6 -5 -3 5 10 6 -1 2 -2 -7 4 -4 5 2 -10 -8 9 7 7 9 -7
2 2 9 2 6 6 -3 8 -4 -6 0 -2 -8 6 3 8 10 -5 8 8 8 8 0 -1 4 -5 9 -7 -10 1 -7 6 1 -10 8 8 -8 -9 6 -3 -3 -9 1 4 -9 2 5 -2 -10 8 3 3 -1 0 -2 4 -5 -2 8 -8
9 2 7 9 -10 4 9 10 -6 5 -3 -5 5 1 -1 -3 2 3 2 -8 -9 10 4 10 -4 2 -5 0 -4 4 6 -1 9 1 3 -7 6 -3 -3 -9 6 10 8 -3 -5 5 2 6 -1 2 5 10 1 -3 3 -10 6 -6 9 -
3 -9 9 -10 6 7 7 10 -6 0 6 8 -10 6 4 -4 -1 7 4 -9 -3 -10 0 -6 7 10 1 -9 1 9 5 7 -2 9 -8 10 -8 -7 0 -10 -7 5 3 2 0 0 -1 10 3 3 -7 8 7 5 9 -7 3 10 7
10 0 -10 10 7 5 6 -6 6 -9 -1 -8 9 -2 8 -7 -6 -8 5 -2 1 -9 -8 2 9 -9 3 3 -8 1 -3 9 1 3 6 -6 9 -2 5 8 2 -6 -9 -9 1 1 -9 5 -4 -9 6 -10 10 -1 8 -2 -6 8 -
9 9 0 8 0 4 8 -7 -9 5 -4 0 -9 -8 2 -1 5 -6 -5 5 9 -8 3 8 -3 -1 -10 10 -9 -10 3 -1 1 -1 5 -7 -8 -5 -10 1 7 -3 -6 5 5 2 6 3 -8 9 1 -5 8 5 1 4 -8 7 1 3 -
5 10 -9 -2 4 -5 -7 8 8 -8 -7 9 1 6 6 3 4 5 6 -3 -7 2 -2 7 -1 2 2 2 5 10 0 9 6 10 -4 9 7 -10 -9 -6 0 -1 9 -3 -9 -7 0 8 -5 -7 -10 10 4 4 7 3 -5 3 7 6
3 -1 9 -5 4 -9 -8 -2 7 10 -1 -10 -10 -3 4 -7 5 -5 -3 9 7 -3 10 -8 -9 3 9 3 10 -10 -8 6 0 0 8 1 -7 -8 -6 7 8 -1 -4 0 -1 1 -4 4 9 0 1 -6 -5 2 5 -1 2 7
-8 5 -7 7 -7 9 -8 -10 -4 10 6 -1 -4 -5 0 -2 -3 1 -1 -3 4 -4 -6 4 5 7 5 -6 -6 4 -10 -3 -4 -4 -2 6 0 1 2 1 -7
```

# Challenge Note

Like any challenge of this complexity class, you are somewhat constrained to solving the problem with brute-force (sum all possible sub-sets). We really want to encourage any and all new ideas, so really go wild and absolutely do whatever you think could solve this problem quickly!

Title: [05/2/13] Challenge #121 [Hard] Medal Management
Text: # [](#HardIcon) *(Hard)*: Medal Management

The moderators of /r/DailyProgrammer give out medals (either gold or silver) as community rewards / community achievements. Though everyone has the two medal icons next to their names, the actual amount you have are reflected as two integers (gold first, then silver). The side-bar to the right has a section titled "Achievements System", which describes how medals are earned.

The problem though is that mods have to use the sub-Reddit's administration page to add the basic flair to a user and to change the medal count in any way. Though not hard, it certainly isn't a simple process, so we would like your help in building a better solution!

Your goal is to write a single web-page in JavasScript that "wraps" these admin features together in a nice single form. Essentially it should be a page with **minimal server-side code** or **you can ditch the idea of a page and just make a browser add-on (Chrome or FireFox please)**, when given Reddit login credentials, allows:

* Loading a user's flair string and type
* Saving a user's flair string and type
* Allowing a one-click +1 Gold and +1 Silver for any given Reddit username
* Load a user's /r/DailyProgrammer post history for any given Reddit username

Reddit provides an external API interface for these purposes: [learn more about the web-based API here](http://www.reddit.com/dev/api).

Though this will be a typical [hard] level challenge, we _will_ be giving out a gold medal _and_ Reddit gold (3 months) for the person who gives a fully-featured solution. Note that solutions must be open-source (hey, we want to use your system!) and you will be given full credits to it in our sub-Reddit's side-bar. Starting from today (Friday), all solutions are due in exactly 7 days: the competition ends at 11:55pm, American pacific time, UTC−8. It'll take about day to confirm who wins.

To help get started, check out these Reddit JavaScript APIs: (note that none are a "perfect" solution, and some heavy work will be required)

* [Official Reddit API](http://www.reddit.com/dev/api)
* [Handson Reddit](https://github.com/timisbusy/handson-reddit)
* [Reddit JS Client](https://github.com/tommyvyo/reddit-js-client)

This is quite a big challenge to take on, so I'll be much more involved with responding to questions and comments. Good luck, and have fun!

Edit 1: Our awesome user [Skeeto](http://www.reddit.com/r/dailyprogrammer/comments/1dk7c7/05213_challenge_121_hard_medal_management/c9r4obj) has pointed out that a pure client-side implementation is not possible for security reasons; my bad! I've updated the rules to allow minimal server-side code or the choice of just making all of this a browser add-on.

Title: [03/22/13] Challenge #120 [Hard] Derpson Family Party
Text:
[](#HardIcon) *(Hard)*: Derpson Family Party
The Derpsons are having a party for all their relatives. It will be
the greatest party ever held, with hired musicians, a great cake and a
magical setting with two long tables at an old castle. The only
problem is that some of the guests can't stand each other, and cannot
be placed at the same table.

The Derpsons have created a list with pairs of enemies they know will
start a fight if put together. The list is rather long so it is your
mission to write a program to partition the guests into two tables.

Author: emilvikstrom
# Formal Inputs & Outputs
## Input Description
The input is a list of enemies for each guest (with empty lines for
guests without enemies). Each guest have a number which is equivalent
to the line number in the list.

It is a newline-separated file (text file or standard in). Each line is a
comma-separated (no space) list of positive integers. The first
line of the input is called 1, the second 2 and so on. This input can
be mapped to an array, *arr*, indexed from 1 to *n* (for *n* guests)

with lists of numbers. Each index of the array is a guest, and each
number of each list is another guest that he or she cannot be placed with.

If a number *e* appears in the list *arr[k]*, it means that *e* and *k*
are sworn enemies. The lists are *symmetric* so that *k* will also
appear in the list *arr[e]*.
## Output Description
A newline-separated list (on standard out or in a file) of guest
numbers to put at the first table, followed by an empty line and then
the guests to place at the second table. You may just return
the two lists if printing is non-trivial in your language of choice.

All guests must be placed at one of the two tables in such a way that
any two people at the same table *are not* enemies.

The tables do not need to be the same size. The lists do not need to
be sorted.

Additionally, if the problem is impossible to solve, just output
"No solution".
# Sample Inputs & Outputs
## Sample Input
```
2,4
1,3
2
1
```
## Sample Output
```
1
3

4
2
```
# Challenge Input
This is the input list of enemies amongst the Derpsons:
http://lajm.eu/emil/dailyprogrammer/derpsons (1.6 MiB)

Is there a possible seating?
## Challenge Input Solution
What is your answer? :-)

Title: [03/08/13] Challenge #120 [Hard] Bytelandian Exchange 3
Text:
[](#HardIcon) *(Hard)*: Bytelandian Exchange 3
Bytelandian Currency is coins with positive integers on them. (Don't worry about 0-valued coins because they're useless for this problem.) You have access to two peculiar money changing machines:

 * Machine 1 takes one coin of any value N. It pays back 3 coins of the values 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.
 * Machine 2 takes two coins at once, one of any value N, and one of any positive value. It returns a single coin of value N+1.

These two machines can be used together to get arbitrarily large amounts of money from a single coin, provided it's worth enough. If you can change an N-valued coin into an N-valued coin *and* a 1-valued coin, you can repeat the process to get arbitrarily many 1-valued coins. __What's the smallest N such that an N-valued coin can be changed into an N-valued coin and a 1-valued coin?__

For instance, it can be done with a coin worth 897. Here's how. Use Machine 1 to convert it into coins worth 448, 299, and 224. Through repeated applications of Machine 1, the 299-valued coin can be converted into 262 1-valued coins, and the 224-valued coin can be converted into 188 1-valued coins. At this point you have a 448-valued coin and 450 1-valued coins. By using Machine 2 449 times, you can make this into a 897-valued coin and a 1-valued coin. To summarize this strategy:

  * 897 -> 448 + 299 + 224 (Machine 1)
  * 299 + 224 -> 450x1 (repeated Machine 1)
  * 448 + 450x1 -> 897 + 1 (repeated Machine 2)

But of course, 897 is not the lowest coin value that lets you pull this trick off. Your task is to find the lowest such value.

[Here is a python script that will verify the steps of a correct solution (will not verify that it's optimal, though).](http://pastebin.com/JJuKJBLp)

Author: Cosmologicon
# Formal Inputs & Outputs
## Input Description
None
## Output Description
The lowest N such that an N-valued coin can be converted into an N-valued coin *and* a 1-valued coin.
# Sample Inputs & Outputs
## Sample Input
None
## Sample Output
None
# Challenge Input
None
## Challenge Input Solution
None
# Note
Please direct any questions about this challenge to /u/Cosmologicon

__Bonus:__ Now consider the case where Machine 1 is broken and will not give out any 1-valued coins (so for instance, putting a 5-valued coin into Machine 1 gives you a 2-valued coin and nothing else). In this case, what's the smallest N such that an N-valued coin can be converted into an N-valued coin *and* a 2-valued coin?

Title: [03/01/13] Challenge #119 [Hard] Polygon diagonals
Text:
[](#HardIcon) *(Hard)*: Polygon diagonals
In how many distinct ways can you divide a regular N-sided polygon into N-2 triangles using N-3 non-intersecting diagonals?

For instance, there are 3 ways to do divide a regular hexagon (6-sided polygon) into 4 triangles using 3 non-intersecting diagonals, as shown here: http://i.imgur.com/gEQHq.gif

A _diagonal_ of a regular polygon is a line joining two non-adjacent vertices. Two diagonals are _non-intersecting_ if they don't cross within the interior of the polygon. Two ways are _distinct_ if one cannot be rotated and/or reflected to become the other.

What's the largest N that your program can handle in a reasonable amount of time? Post the solution for N = 23 if you can get it.

Author: skeeto
# Formal Inputs & Outputs
## Input Description
Number of polygon sides N
## Output Description
Number of distinct ways to divide the N-gon into N-2 triangles using N-3 non-intersecting diagonals.
# Sample Inputs & Outputs
## Sample Input
6
## Sample Output

3
# Challenge Input
11
## Challenge Input Solution
228
# Note
None

Title: [01/25/13] Challenge #118 [Hard] Alphabetizing cipher
Text:
[](#HardIcon) *(Hard)*: Alphabetizing cipher
This challenge is an optimization problem. Your solution will be a string of the 26 letters of the alphabet in some order, such as:

```
    jfbqpwcvuamozhilgrxtkndesy
```

The string is a cipher. For this cipher, the letter `a` maps to `j`, the letter `b` maps to `f`, and so on. This cipher maps the word `bakery` to `fjmprs`. Notice that `fjmprs` is in alphabetical order. Your cipher's score is the number of words from the word list that it maps to a string in alphabetical order.

[The word list for this problem is here.](http://pastebin.com/9aFn1r27) It consists of the 7,260 six-letter words from the Enable word list that are made up of 6 different letters.

Since there are 60 words from the list that my example cipher maps to sorted strings, my score is 60. Can you do better? Post your solution, your score, and the program you used to generate it (if any).

Here's a python script that will evaluate your solution:

```
    abc = "abcdefghijklmnopqrstuvwxyz"
    words = open("enable-6.txt").read().splitlines()
    newabc = raw_input()
    assert len(newabc) == 26 and set(abc) == set(newabc)
    cipher = dict(zip(abc, newabc))
    for word in words:
      nword = "".join(map(cipher.get, word))
      if sorted(nword) == list(nword):
        print word, nword
```

Author: Cosmologicon
# Formal Inputs & Outputs
## Input Description
<Field to be removed>
## Output Description
<Field to be removed>
# Sample Inputs & Outputs
## Sample Input
<Field to be removed>
## Sample Output
<Field to be removed>
# Challenge Input
<Field to be removed>
## Challenge Input Solution
<Field to be removed>
# Note
None

Title: [01/18/13] Challenge #117 [Hard] Verify Your Language!
Text: # [](#HardIcon) *(Hard)*: Verify Your Language!
[Context-free grammar](http://en.wikipedia.org/wiki/Context-free_grammar) is a tool heavily used in programming language design, verification, compiling, and execution. It is, essentially, a formal language used to define a grammar (i.e. another language). CFG are "more powerful" (in that they can verify more complex languages) than other grammars, such as regular-expressions.

Programming language expressions are generally validated through CFGs. This is done by applying several products on an expression, subdividing the statement into known components, and finally into "terminal" components (i.e. parts of an expression that cannot be more subdivided). An example could be a CFG that only accepts addition expressions, such as "123 + 456". Such a CFG would have two rules that could be applied to verify if this expression was valid: A -> Int + Int, and Int -> '0123456789'Int | NULL

It is ** extremely important** that the reader understands CFG and the formal language associated with it - the above is simply a refresher / casual introduction to the subject.

Your goal is to write a program that accepts a CFG definition and a series of expressions, printing true or false for each expression if it is a valid expression of the given CFG.

Author: nint22
# Formal Inputs & Outputs
## Input Description
First, your program must accept an integer N, where there will be N products, one per line, right below this integer.

To keep things simple, products must be a single upper-case letter, such as "S". The letter "E" is reserved as the null-terminator. The equal-character "=" is reserved as the product definition operator. Finally, the pipe-character "|" is reserved for defining sets of possible products.

This syntax is similar to the "on-paper" definition, with the small difference of substituting "E" and "->" for the greek-letter and arrow symbols.

Assume that the grammar is valid; you do not have to error-check or handle "bad" CFG definitions.

Next, you program must accept an integer M, where there will be M expressions, one per line, that must be tested by the above-defined grammar.
## Output Description
For each expression M, print true or false, based on wether or not the expression is a valid expression of the given CFG.
# Sample Inputs & Outputs
## Sample Input
```
3
S = SS
S = (S)
S = ()
4
()
((()))
(()(()))
(()
```
## Sample Output
```
True
True
True
False
```
# Challenge Input
```
8
S = x
S = y
S = z
S = S + S
S = S - S
```

S = S * S
S = S / S
S = (S)
3
(x + y) * x - z * y / (x + x)
(xx - zz + x / z)
x + y * x - z * y / x + x
## Challenge Input Solution
True
False
False
# Note
Some grammars may be ambiguous! Make sure to research what that means, though it should not affect your solution - I mention this simply to give you a warning if you see odd parsing behavior while debugging.

Bonus: A short-hand method of having multiple products from one function-name is the "pipe operator", such as "S = x | y | z", instead of three separate "S = x", "S = y", "S = z". Support this notation system as a bonus.

Title: [01/11/13] Challenge #116 [Hard] Maximum Random Walk
Text:
[](#HardIcon) *(Hard)*: Maximum Random Walk
Consider the classic random walk: at each step, you have a 1/2 chance of taking a step to the left and a 1/2 chance of taking a step to the right. Your expected position after a period of time is zero; that is the average over many such random walks is that you end up where you started. A more interesting question is what is the expected rightmost position you will attain during the walk.

Author: thePersonCSC
# Formal Inputs & Outputs
## Input Description
The input consists of an integer n, which is the number of steps to take (1 <= n <= 1000). The final two are double precision floating-point values L and R which are the probabilities of taking a step left or right respectively at each step (0 <= L <= 1, 0 <= R <= 1, 0 <= L + R <= 1). Note: the probability of not taking a step would be 1-L-R.
## Output Description
A single double precision floating-point value which is the expected rightmost position you will obtain during the walk (to, at least, four decimal places).

# Sample Inputs & Outputs
## Sample Input
walk(1,.5,.5)
walk(4,.5,.5)
walk(10,.5,.4)
## Sample Output
walk(1,.5,.5) returns 0.5000
walk(4,.5,.5) returns 1.1875
walk(10,.5,.4) returns 1.4965
# Challenge Input
What is walk(1000,.5,.4)?
## Challenge Input Solution
(No solution provided by author)
# Note
* Have your code execute in less that 2 minutes with any input where n <= 1000

* I took this problem from the regional ACM ICPC of Greater New York.

Title: [1/2/2013] Challenge #115 [Difficult] Pack-o-Tron 5000
Text: **Description:**

Overdrive Mega-Corporation is coming out with a new and brilliant commercial electronic device that packs your bags for you! It is named "Pack-o-Tron 5000": an automated box-packing solution. Moving to a new home? Traveling overseas? Going on a business trip? No matter what, this device will help you pack your boxes optimally, reducing empty space and fitting more objects in less area!

As the lead engineer, you are tasked with designing the code that generates the "optimal placement" of items in a given area. Fortunately for you, the "Pack-o-Tron 5000" only works with very specific definitions of "items" and "areas". (Shh, don't tell the customers that, marketing is still working on it).

An "area" is an empty 2D grid, where the length and width are integers representing inches. As an example, a suitcase could be defined as a 36-inch by 48-inch grid. An "item" is a 2D object, whose length and width are integers representing inches. As an example, a tooth-brush item can be 1-inch in width, and 3-inches long.

Your goal is to place all given items in a given area as optimally as possible. "Optimally" is defined as having the smallest minimum-rectangle that spans your set of items.

Note that the "area" is defined as a grid where the origin is in the top-left and X+ grows to the right, with Y+ growing to the bottom. An item's origin is in the top-left, with X+ growing to the right, and Y+ growing to the bottom. A minimum-rectangle that spans your set of items is a rectangle on the grid that includes all items placed, and is either equal to or smaller than the given area. Your goal is to make this minimum-span as small as possible. You are allowed to place your objects in any position, as long as it is in the grid. You are also allowed to rotate your objects by 90 degrees.

[Here is an album of several examples of how objects are placed, and how they can be moved and rotated to minimized the minimum-span rectangle.](http://imgur.com/a/M3MNk)

Formal Inputs & Outputs:

Input Description:

You will first be given two integers: the width and height of your starting (empty) grid. After, you will be given another integer representing the number of following items you are to place in this grid. For each item, you will be given two integers: the width and height of the object. All of this is done through standard input. (i.e. console).

Output Description:

Once you have optimally placed the given items in the given grid such that it as the smallest minimum-span possible, you must print each item's x and y coordinate (as an integer), and the object's size (to show us if there has been any rotations).

Sample Inputs & Outputs:

[Take a look at the example images here.](http://imgur.com/a/M3MNk) For all images that have the two 1x3 items, you would be given the following sample input:

```
8 8
2
1 3
1 3
```

The most optimal result (the last image) is a 2x3 minimum-span, which can be described in the following:

```
0 0 1 3
1 0 1 3
```

For those who are keen observers, an equivalent solution is the same pair of objects, but mirrored on the diagonal axis:

    0 0 3 1
    0 1 3 1

Note:

This is essentially a clean version of the [Packing Problem](http://en.wikipedia.org/wiki/Packing_problem). Since the packing problem is in the computational complexity class of NP-hard, there is no known algorithm to run in P-time (...yet! Maybe there is! But that's the whole P-NP relationship problem, isn't it? :P). Instead, look into heuristic algorithm designs, and try to avoid brute-force solutions.

Title: [12/4/2012] Challenge #114 [Difficult] Longest word ladder
Text: What's the longest valid [word ladder](http://www.reddit.com/r/dailyprogrammer/comments/149kec/1242012_challenge_114_easy_word_ladder_steps/) you can make using [this list of 3,807 four-letter words](http://pastebin.com/zY4Xt7iB) without repeating any words? (Normally a word ladder would require you to take the shortest possible path between two words, but obviously you don't do that here.)

[Here's a ladder I found of 1,709 words](http://pastebin.com/DgeX0CTC). What's the best you can do? Also post the code you used to generate it, of course.

Title: [11/20/2012] Challenge #113 [Difficult] Memory Allocation Insanity!
Text: **Description:**

Cyberdyne Systems Corporation is working on a powerful new processors, but due to a management snafu, the management has only allowed your code 512 Kilobytes (524288 Bytes) to implement your application's heap! For those unfamiliar with the [heap](http://en.wikipedia.org/wiki/Dynamic_memory_allocation#Dynamic_memory_allocation), it is an area of memory for processes where (the process) can allocate variable memory sizes on at run-time.

The problem here is that you can't use any pre-built code or libraries to serve your memory allocation needs in this tiny space, so you are now going to have to re-implement your own [malloc and free](http://en.wikipedia.org/wiki/C_dynamic_memory_allocation) functions!

Your goal is to implement two functions, regardless of language, named "malloc" and "free". Malloc takes a number of bytes (up to a maximum of 128 Kb at a time) and returns either a new address (array) that your process can use, or returns an invalid point (empty array) if there is not enough free space. This array must be continuous (i.e. a continuous block of 128 Kb). Free simply takes the given array and allows it to be reused by future malloc function calls.

Your code must only work in 512Kb, meaning that both the allocate memory AND the related data structures must reside in this 512Kb space. Your code is not part of this measurement. As an example, if you use a linked-list that requires one byte over-head for each allocated chunk, that means you must be able to contain this linked-list structure and the allocated spaces.

There are many methods to implement a heap structure; investigate either the Linux Slab allocator, or try to stick to the obvious solutions of linked lists. Don't forget to coalesce freed spaces over time! An example of this situations is when you have three blocks, left, middle, and right, where the left and right are unallocated, but the middle is allocated. Upon free-ing the middle block, your code should understand that there aren't three free blocks left, but instead one large unified block!

Formal Inputs & Outputs:

Input Description:

* `void* malloc(size_t ByteCount);` // Returns a pointer to available memory that is the "ByteCount" size in bytes
* `void free(void* Ptr);` // Frees a given block of memory on this heap

Output Description:

No formal output is required, but a helpful tool for you to develop is printing the memory map of the heap, useful for debugging.

Sample Inputs & Outputs:

* `void* PtrA = Malloc(131072);` // Allocating 128Kb; success
* `void* PtrB = Malloc(131072);` // Allocating 128Kb; success
* `void* PtrC = Malloc(131072);` // Allocating 128Kb; success
* `void* PtrD = Malloc(131072);` // Allocating 128Kb; fails, unlikely to return 128Kb since any implementation will require memory over-head, thus you will have less than 128Kb left on the heap before calling this function
* `free(PtrC);` // Free 128Kb; success
* `void* PtrD = Malloc(131072);` // Allocating 128Kb; success

Note

It is likely that you will have to implement this simulation / code in C or C++, simply because many high-level languages such as Java or Ruby will hide the necessary low-level memory controls required. You can still use these high-level languages, but you must be very strict about following the memory limitation rule.

Title: [11/14/2012] Challenge #112 [Difficult]What a Brainf***
Text: **Description:**

[BrainFuck](http://en.wikipedia.org/wiki/Brainfuck), is a [Turing-complete](http://en.wikipedia.org/wiki/Turing_completeness) (i.e. computationally-equivalent to modern programming languages), esoteric programming language. It mimics the concept of a [Turing machine](http://en.wikipedia.org/wiki/Turing_machine), a Turing-complete machine that can read, write, and move an infinite tape of data, through the use of the language's eight (that's right: 8!) operators.

An example is:

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

Which prints "Hello World!"

Your goal is to write a BrainFuck interpreter from scratch, and have it support both single-character output to standard-out and single-character input from standard-in.

Formal Inputs & Outputs:

Input Description:

String BFProgram - A string of a valid BrainFuck program.

Output Description:

Your function must execute and print out any data from the above given BrainFuck program.

Sample Inputs & Outputs:

See above

Notes:

This is a trivial programming challenge if you understand the basis of a Turing-machine. I strongly urge you to read all related Wikipedia articles to prepare you for this. A more significan't challenge would be

Title: [11/6/2012] Challenge #111 [Difficult] The Josephus Problem
Text: Flavius Josephus was a roman historian of Jewish origin. During the Jewish-Roman wars of the first century AD, he was in a cave with fellow soldiers, 41 men in all, surrounded by enemy Roman troops. They decided to commit suicide by standing in a ring and counting off each third man. Each man so designated was to commit suicide. (When the count came back around the ring, soldiers who had already committed suicide were skipped in the counting.) Josephus, not wanting to die, placed himself in position #31, which was the last position to be chosen.

In the general version of the problem, there are n soldiers numbered from 1 to n and each k-th soldier will be eliminated. The count starts from the first soldier. Write a function to find, given n and k, the number of the last survivor. For example, `josephus(41, 3) = 31`. Find `josephus(123456789101112, 13)`.

Title: [11/3/2012] Challenge #110 [Difficult] You can't handle the truth!
Text: **Description:**

[Truth Tables](http://en.wikipedia.org/wiki/Truth_table) are a simple table that demonstrates all possible results given a Boolean algebra function. An example Boolean algebra function would be "A or B", where there are four possible combinations, one of which is "A:false, B:false, Result: false"

Your goal is to write a Boolean algebra function truth-table generator for statements that are up to 4 variables (always A, B, C, or D) and for only the following operators: [not](http://en.wikipedia.org/wiki/Logical_NOT), [and](http://en.wikipedia.org/wiki/Logical_AND), [or](http://en.wikipedia.org/wiki/Logical_OR), [nand](http://en.wikipedia.org/wiki/Logical_NAND), and [nor](http://en.wikipedia.org/wiki/Logical_NOR).

Note that you must maintain order of operator correctness, though evaluate left-to-right if there are ambiguous statements.

Formal Inputs & Outputs:

Input Description:

String BoolFunction - A string of one or more variables (always A, B, C, or D) and keyboards (not, and, or, nand, nor). This string is guaranteed to be valid

Output Description:

Your application must print all possible combinations of states for all variables, with the last variable being "Result", which should the correct result if the given variables were set to the given values. An example row would be "A:false, B:false, Result: false"

Sample Inputs & Outputs:

Given "A and B", your program should print the following:

A:false, B:false, Result: false
A:true, B:false, Result: false
A:false, B:true, Result: false
A:true, B:true, Result: true

Notes:

To help with cycling through all boolean combinations, realize that when counting from 0 to 3 in binary, you generate a table of all combinations of 2 variables (00, 01, 10, 11). You can extrapolate this out to itterating through all table rows for a given variable count. [Challenge #105](http://www.reddit.com/r/dailyprogrammer/comments/11shtj/10202012_challenge_105_intermediate_boolean_logic/) has a very similar premise to this challenge.

Title: [10/30/2012] Challenge #109 [Difficult] Death Mountains
Text: **Description:**

You are a proud explorer, walking towards a range of mountains. These mountains, as they appear to you, are a series of isosceles triangles all clustered on the horizon. [Check out this example image](http://imgur.com/a/lyhMt), sketched by your awesome aid nint22 (smiling-mountain not important). Your goal, given the position of the base of these triangles, how tall they are, and their base-width, is to compute the overall unique area. Note that you should not count areas that have overlapping mountains - you only care about what you can see (i.e. only count the purple areas once in the [example image](http://imgur.com/a/lyhMt)).

Formal Inputs & Outputs:

Input Description:

Integer n - The number of triangles

Array of triangles T - An array of triangles, where each triangle has a position (float x), a base-length (float width), and a triangle-height (float height).

Output Description:

Print the area of the triangles you see (without measuring overlap more than once), accurate to the second decimal digit.

Sample Inputs & Outputs:

Todo... will have to solve this myself (which is pretty dang hard).

Notes:

It is critically important to NOT count overlapped triangle areas more than once. Again, only count the purple areas once in the [example image](http://imgur.com/a/lyhMt)..

Title: [10/25/2012] Challenge #109 [Difficult] (Steiner Inellipse)
Text: For the difficult challenge, you must find the [Steiner Inellipse](http://en.wikipedia.org/wiki/Steiner_inellipse) of a triangle. The Steiner Inellipse is the ellipse within a triangle that has the maximum area, without exceeding the bounds of the triangle.

For example: [here is an image of a typical inellipse.](http://upload.wikimedia.org/wikipedia/commons/thumb/0/0a/Steiner_Inellipse.svg/250px-Steiner_Inellipse.svg.png)

For your input, you will be given the three coordinates of a triangle. They are:

* Coord 1: (0,2)
* Coord 2: (0,7)
* Coord 3: (7,0)

For your output, you have two options: either use some sort of graphical implementation to draw the found ellipse, or output the equation of that elipse.

Any further information can be found [here](http://projecteuler.net/problem=385) at a similar problem on projecteuler.net.

Title: [10/25/2012] Challenge #107 [Difficult] (English Word Identifier)
Text: Given as input this [list of 20,000 twelve-character strings](http://pastebin.com/eJFn49vp), which contains 10,000 actual English words, and 10,000 random strings of characters, write a program that filters out the English words, with no false positives or negatives. [Your output must match this list exactly.](http://pastebin.com/trMz6nWQ)

Your program doesn't need to work on arbitrary lists of words, it can be custom tailored to this list. You could technically meet the challenge by simply including the solution in your program and outputting it, or by reading it from a file. But the point of the challenge is to make your program, combined with any data it inputs, as short/elegant as possible.

You will probably have an algorithm that does a pretty good job but requires some data to be coded in. That's fine, but the smaller the better.

This is not a "code golf" challenge: don't worry about the exact character count of your programs. It's more about how elegant your solution is. But for reference, I have a python solution that reads in no data and is about 1700 bytes.

Title: [10/23/2012] Challenge #106 [Difficult] (Mongolian Grill)
Text: One of my favorite places to eat is HuHot, a Mongolian Grill style restaurant. At HuHot, you take a bowl and visit any number of vegetable, meat, noodle, and sauce ingredient stations to build your own custom stir fry meal. Once you have your ingredients picked out, you get in line to have your meal cooked on an open grill before you. As would be expected, people spend a little time at each station in order to take ingredients. The cook time takes a few minutes, but the chefs can cook several meals on the grill at the same time.

Your task is to estimate the average time it takes to prepare a meal per customer. Customers will want dishes composed of 5 random ingredients. Assume it takes 10 seconds to take an ingredient from one of 20 stations and only 1 person can be at a station at once. Assume also that it takes 3 minutes for a meal to cook, but 8 meals can cook simultaneously. If the grill or a station is full, customers will wait in line until it is their turn.

Determine the average time to wait for the following:

- Only 1 customer
- 8 customers at the same time
- 25 customers at the same time
- 25 customers, staggered 1 minute apart
- 100 customers stampeding at the same time
- 100 customers, staggered 2 minutes apart

Extra Credit:

- Estimate the average time customers wait on the grill itself
- Estimate the average time customers wait on each of the 20 stations

Title: [10/20/2012] Challenge #104 [hard] (Image color analyzer)
Text: Sorry for taking the last challenge down but it seems it was already done. I had searched for "slide puzzle" -- what they're usually called -- but not "15 puzzle."

Anyway, new challenge! Make a program that analyzes the color of an image. What the majority colors are, specifically. Output if they fall into red, blue, green, etc. because saying "1 instance of <obscure color here>" takes of space, especially if that obscure color is a blue or red or green.

Example input:

`$ color_analyze image.jpg`

Example output:

`Pixels that are red: 15`
`Pixels that are blue: 227`
`Pixels that are green: 1,745`

Title: [10/18/2012] Challenge #104 [Hard] (Stack Attack)
Text: **Description:**

You are a devilish engineer designing a new programming language titled D++, which stands for a "Dijkstra++", named after your favorite computer scientist. You are currently working on the math-operations parsing component of your interpreter: though the language only supports [infix-notation](http://en.wikipedia.org/wiki/Infix_notation) (such as "1 + 2 * 3"), your interpreter internals require you to represent all math strings in [reverse polish notation](http://en.wikipedia.org/wiki/Reverse_Polish_notation) (for easier, stack-based, computing). Your goal is to simply take a guaranteed valid infix-notation string of math operations and generate (printing it out) an equivalent and valid reverse polish notation string.

Formal Inputs & Outputs:

Input Description:

string MathOperations - A valid string of infix math notation.

Output Description:

Print the converted RPN form of the given math operations string.

Sample Inputs & Outputs:

"1 + 2" should be printed as "1 2 +". "(1+2)*3" should be printed as "3 2 1 + *". "(6 (7 – 2)) / 3 + 9 * 4" should be printed as "6 7 2 - * 3 / 9 4 * +".

Notes:

Do not get trapped in overly-complex solutions; there are formal solutions, though many ways of solving for this problem. Check out the [Shunting Yard Algorithm](http://en.wikipedia.org/wiki/Shunting_yard_algorithm) for details on how to convert math-operation strings (a stack of tokens) from one notation system to another.

Title: [9/30/2012] Challenge #102 [difficult] (Pokémon types)
Text: Ah, who doesn't remember the endless hours wasted playing Pokémon games on a Game Boy during long car rides? I sure do. Pokémon had an interesting battle system, and one of the nice mechanics was the type system.

For this challenge, you'll be writing a function, `type_effect`, that takes two string arguments -- the offending move's name and the defending Pokémon's name -- and returns a multiplier like `2.0` or `0.25`.

Generally, you take the offending move's type, look up the multipliers for all the defending Pokémon's types in the type chart, and multiply them together. As an example, we'll run through the calculations for `type_effect("Ice Beam", "Dragonite")`.

(Optionally, use `enum`s instead of strings, like `type_effect(M_ICE_BEAM, P_DRAGONITE)`).

  - [Ice Beam](http://bulbapedia.bulbagarden.net/wiki/Ice_Beam_(move\)) is an Ice move.
  - [Dragonite](http://bulbapedia.bulbagarden.net/wiki/Dragonite_(Pok%C3%A9mon\)) has multiple types, Dragon and Flying.
  - According to the [type chart](http://bulbapedia.bulbagarden.net/wiki/Type_chart), Ice vs. Dragon has a 2.0× bonus, and Ice vs. Flying has a 2.0× bonus, too. Multiplying these together, you get 4.0×, so return `4.0`.

Obviously, this challenge is all about collecting the data you need yourself and manipulating it, so **don't steal each others' representations** of the Type array, Pokémon's types, etc!

Title: [9/27/2012] Challenge #101 [difficult] (Boolean Minimization)
Text: For difficult 101, I thought I'd do something with binary in it.

Write a program that reads in a file containing 2^n 0s and 1s as ascii characters. You will have to solve for N given the number of 0s and 1s in the file,
as it will not be given in the file. These 0s and 1s are to be interpreted as the outputs of a truth table in N variables.

Given this truth table, output a minimal boolean expression of the function in some form. (
[Hint1](http://en.wikipedia.org/wiki/Quine%E2%80%93McCluskey_algorithm),
[hint2](http://en.wikipedia.org/wiki/Karnaugh_map))

For example, one implementation could read in this input file

```
    0000010111111110
```

This is a 4-variable boolean function with the given truth table. The program could minimize the formula, and could output

```
    f(abcd)=ac'+ab'+bcd'
```

or

```
    f(0123)=02'+01'+123'
```

Title: [9/20/2012] Challenge #100 [difficult] (Min-Max of N-Dimensional Quadratic)
Text: A quadratic form in mathematics is any polynomial in N-Dimensions which has no exponent greater than 2. For example, the equations

```
    f(x)=x^2+1
    f(x,y,z)=xy+10x-2z^2
    f(x,y,z,w)=x^2+y^2+z^2-w^2
```

Are all quadratic forms in 1,3,and 4, dimensions, respectively. It is a part of linear algebra, that ANY quadratic in dimension N can be fully specified by an (N+1)x(N+1) symmetric matrix A of coefficients. The quadratic f(x) is then defined as

```
    f(v) = v' A v
```

(where x' denotes x transposed), and v is a homogenous vector of length N+1 with a 1 in the last position.

This would imply that any constant term is always in the lower left, the coefficients of the squared terms are along the diagonal, and the coefficients of the products are split half and half on the off-diagonal parts.

For example, to represent the quadratic

```
    f(y)=y^2+1,
```

We can use a 2-dimensional matrix

```
    A= [1 0;
       0 1]
    f(y)=[y 1]*[1 0;0 1][y;1]
    Doing out the matrix multiplication gives us the original f(y).
```

Another example, to represent the quadratic

```
    f(x,y,z)=xy+10x-2z^2+9
```

could be the 4x4 matrix A

```
    A=[0  .5  0 5;
       .5  0  0 0;
       0   0 -2 0;
       5   0  0 9;]
```

Every quadratic form has at least one point where the quadratic is an extrema: that is, where it is a global minimum or maximum or 'saddle point' in N dimensions.

Write a function that will take in a quadratic form defined as a (N+1)x(N+1) symmetric matrix, and output one of the extrema points of that quadratic. either the global min,max,or a saddle point.

Title: [9/17/2012] Challenge #99 [difficult] (Animated unemployment map of the United States)
Text: Improve the program from [today's intermediate problem](http://www.reddit.com/r/dailyprogrammer/comments/101mi5/9172012_challenge_99_intermediate_unemployment/) to make it be able to create not just still images of unemployment, but animations of how unemployment has changed over time, with the maps as the frames of animation. The output can either be in the form of animated gifs or a movie file.

Your program should be fully automated: it should take as inputs the start and end date, and produce the finished animation. As in the previous problem, any extra information on the map is optional, but for this problem it is strongly recommended (though, again, not required) that you list the date, so you can see when each map is represented.

Create an animation that shows how unemployment developed from 2000 to today. I imagine it would get quite interesting around 2007. Please show us the results you get, either by uploading the animations to imgur.com (if they're animated gifs) or to YouTube (if they're movie files)!

Title: [9/15/2012] Challenge #98 [difficult] (Reading digital displays)
Text: [Challenge #92 [easy]](http://www.reddit.com/r/dailyprogrammer/comments/ywlvf/8272012_challenge_92_easy_digital_number_display/) involved converting a number to a seven segment display representation (of a variable size) using +, -, and |. Assume the font looks like this:

```
    + +--+ +--+ +  + +--+ +--+ +--+ +--+ +--+ +--+
    |   |   | | | |   |      | | | | | | |
    |   |   | | | |   |      | | | | | | |
    + +--+ +--+ +--+ +--+ +--+    + +--+ +--+ +  +
    | |     |   |   | | |   | | |   | | |
    | |     |   |   | | |   | | |   | | |
    + +--+ +--+    + +--+ +--+    + +--+ +--+ +--+
```

Write a program that reads such a string and converts it back into a number. (You'll have to deduce the size yourself.) The output for the above text would be `1234567890`.

As a bonus, have your program be able to read a file containing characters of different sizes, like this:

```
    +-+ +  + +-+
      | |  | |
+-+ |  | +-+
  | +--+   |
+-+    | +-+
       |
       +
```

Title: [9/08/2012] Challenge #97 [difficult] (Markdown to HTML)
Text: [Markdown](http://daringfireball.net/projects/markdown/syntax) is the text markup system used by reddit in comments and text submissions. Write a script that converts a piece of Markdown into HTML.

Title: [9/06/2012] Challenge #96 [difficult] (Water Droplets)
Text: Your wet dog rand across your hardwood floor. It drops a series of water droplets randomly across the floor. The water droplets each land at particular positions on your infinite floor, and they each create a wet circle of a given radius across the floor.

What is the total area of wet? The input file will be a list of drop descriptions, one per line. Each drop description is three floating point numbers of the format x0 y0 radius, describing the center of the circle and the radius.

Estimate the area of wet accurate to 3 decimal places.

Example input file:

```
0.479477  -0.634017   0.137317
-0.568894  -0.450312  0.211238
-0.907263  -0.434144   0.668432
0.279875   0.309700  0.242502
-0.999968  -0.910107  0.455271
0.889064  -0.864342  1.292949
-0.701553   0.285499  0.321359
-0.947186   0.261604  0.028034
0.805749  -0.175108   0.688808
0.813269  -0.117034  0.340474
-0.630897  -0.659249  0.298656
-0.054129  -0.661273  0.270216
0.042748   0.469534  0.759090
0.079393  -0.803786   0.635903
-0.987166   0.561186   0.740386
-0.246960  -0.774309   1.035616
-0.189155  -0.244443  0.187699
0.683683  -0.569687   0.275045
-0.249028  -0.452500   0.713051
-0.070789  -0.898363   0.135069
```

Example output: (warning: flawed mod solution might be wrong)

```
Total wet area: 12.065
```

EDIT: I apologize, I generated the radii randomly and didn't notice some were negative. My solution simply takes the absolute value of the radius by default. (I'm using an r^2) so it didn't matter. I'm fixing the data now.

Title: [9/03/2012] Challenge #95 [difficult] (Overlapping rectangles)
Text: Let the four values (x, y, w, h) define a rectangle. Let the bottom left corner be located at (x, y) and let (w, h) be the width and height. Then the rectangles (0,0,4,3), (4,3,3,4) and (7,0,3,3) would look something like this:

```
    * * *
    * * *
    * * *
    * * *
* * * *     * * *
* * * *     * * *
* * * *     * * *
```

The total area of these three rectangles is simple to compute, it's just the sum of the areas of each individual rectangle: 4*3 + 3*4 + 3*3 = 12 + 12 + 9 = 33.

It gets more tricky when the rectangles start overlapping each other. For instance, lets look at the three rectangles (0,0,4,3), (2,1,3,4) and (4,4,3,3):

```
    * * *
    * * *
  * * + * *
  * * *
* * + + *
* * + + *
* * * *
```

You see that the rectangles overlap (the regions where they overlap is marked by a + instead of a *). So if we just calculate the sum of the areas like we did before, 12 + 12 + 9, we get too high a value, because we're counting the areas with the overlap twice. To get the correct answer, we have to subtract the areas where two rectangles intersect. The righ answer for the total area covered by the rectangles is then (12 + 12 + 9) - (4 + 1) = 28.

Things get even more hairy when three rectangles intersect. Take a look at (0,0,4,3), (2,1,3,4) and (3,0,3,3):

```
  * * *
  * * *
* * + x + *
* * + x + *
* * * + * *
```

Now the three rectangles all overlap at the points marked x (as before, the +'s mark where only two rectangles overlap). We do as before and start by calculating the sum of the areas of all three triangles: 12 + 12 + 9. Then we subtract the sum of all areas where two rectangles intersect, which is now 4 + 3 + 4. This is because rectangle 1 and 2 intersect in a region with the area 4, rectangles 1 and 3 intersect in an region with an area of 3 (this is th 1*3 "column" that includes the x's and the + right below them), and rectangles 2 and 3 intersect in a region with the area of 4. So far we've got (12 + 12 + 9) - (4 + 3 + 4).

However, by subtracting out all regions where two rectangles intersect, we've subtracted out the region where three rectangles intersect (i.e. the x's) three times. That is to say, we're not counting it at all. So we have to add that back in to get the right result. So the total area covered by the rectangles is, in fact,

A = (12 + 12 + 9) - (4 + 3 + 4) + (2) = 33 - 11 + 2 = 24

If you wish to verify that number, you can count the *'s, +'s and x's and you'll see that they add up to 24.

This sounds complicated, but the general rule is actually quite simple and elegant. If S1 is the sum of the areas of all rectangles, S2 is the sum of all regions where *two* rectangles intersect, S3 is the sum of all regions where *three* rectangles intersect, S4 is the sum of all regions where *four* rectangles intersect, and so on, the value of the total area covered is:

A = S1 - S2 + S3 - S4 + S5 - S6 + S7 - S8 + ...

This is known in mathematics as the [inclusion-exclusion principle](http://en.wikipedia.org/wiki/Inclusion_exclusion_principle), because you alternate including and excluding areas.

The values in my example correspond to:

```
S1 = 12 + 12 + 9
S2 = 4 + 3 + 4
S3 = 2
S4 = 0
S5 = 0
S6 = 0
...
```

With all variables above S3 equal to zero, so we don't need to count them.

Define a random number generator as follows:

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

Then define a function r(N) which returns rectangles (in the (x,y,w,h) for described here) like this:

```
r(N) = (s(4*N) mod 2000000, s(4*N + 1) mod 2000000, 1 + (s(4*N + 2) mod 99999), 1 + (s(4*N + 3) mod 99999))
```

That is,

```
r(0) = (s(0) mod 2000000, s(1) mod 2000000, 1 + (s(2) mod 99999), 1 + (s(3) mod 99999))
r(1) = (s(4) mod 2000000, s(5) mod 2000000, 1 + (s(6) mod 99999), 1 + (s(7) mod 99999))
```

In actual numbers, r(0) and r(1) is:

```
r(0) = (1456789, 880530, 94008, 74226)
r(1) = (1429729, 1957326, 87910, 3758)
```

Generate 2000 of these rectangles (i.e. r(0) through r(1999)) and calculate the total area they cover.

***

Bonus: Define r(N) like this instead:

```
r(N) = (s(4*N) mod 10000000, s(4*N + 1) mod 10000000, 1 + (s(4*N + 2) mod 99999), 1 + (s(4*N + 3) mod 99999))
```

The only thing that has changed is the modulus on the (x,y) values. Generate 50000 of those rectangles (i.e. r(0) through r(49999)) and calculate the total area that they cover.

***

EDIT: scaled up numbers to increase difficulty

Title: [9/01/2012] Challenge #94 [difficult] (Simple Lisp interpreter)
Text: Lisp is a family of programming languages, known for its extremely simple notation (called S-expressions) in which programs are defined as lists and code can be modified as data. Your task is to write an interpreter for a simple subset of Lisp.

Peter Norvig wrote a [popular article on how to write a simple Lisp interpreter](http://norvig.com/lispy.html) in only 90 lines of Python. You can choose to port his code to a language of your choice, or write one on your own, from scratch. Bonus points for solving today's easy challenge (or maybe even *this* challenge) in your own Lisp dialect!

Title: [8/30/2012] Challenge #93 [difficult] (15-puzzle)
Text: Write a program that can solve a standard ['15-puzzle'](http://en.wikipedia.org/wiki/Fifteen_puzzle).

The program should read in a hex string describing the puzzle state from left to right top to bottom, where F is a blank...for example,:

```
0FD1C3B648952A7E
```

would describe the puzzle

```
+----+----+----+----+
| 0  |    | 13 | 1  |
+----+----+----+----+
| 12 | 3  | 11 | 6  |
+----+----+----+----+
| 4  | 8  | 9  | 5  |
+----+----+----+----+
| 2  | 10 | 7  | 14 |
+----+----+----+----+
```

The program should output the final solution 0123456789ABCDEF, and ALSO output EACH intermediate board state as a string on the way to finding a solution.
Warning: I don't know if the above puzzle is actually solvable.

Title: [8/27/2012] Challenge #92 [difficult] (Bags and balls)
Text: Compute all the permutations of placing 9 balls into 4 bags such that each bag contains an odd number of balls.

Ball count is transient when placing bags within one another. For example, a bag containing 3 balls is placed inside a bag containing 2 balls. The inner bag contains 3 balls and the outer bag contains 5 balls.

Some example permutations:

```
((((9))))
(8(((1))))
(1)(1)((7))
```

Title: [8/24/2012] Challenge #91 [difficult] (Chess move validation)
Text: [Forsyth-Edwards notation](http://en.wikipedia.org/wiki/Forsyth%E2%80%93Edwards_Notation) is a is a notation used by chess players for describing a particular board position of a chess game. It contains information about the pieces, whose turn it is, who can castle, and how many turns have passed, among others. Write a program that reads a FEN file and two coordinates from input, like this:

```
    rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1
    e2 e4
```

Your program parses the FEN board, then determines whether moving the piece on coordinate 1 to coordinate 2 is a valid move, printing either `true` or `false`. As demonstrated here, it is:

```
/|a|b|c|d|e|f|g|h
:-:|:-:|:-:|:-:|:-:|:-:|:-:|:-:|:-:
**8**|♜|♞|♝|♛|♚|♝|♞|♜
**7**|♟|♟|♟|♟|♟|♟|♟|♟
**6**|·|·|·|·|·|·|·|·
**5**|·|·|·|·|·|·|·|·
**4**|·|·|·|·|♙|·|·|·
**3**|·|·|·|·|↑|·|·|·
**2**|♙|♙|♙|♙|◌|♙|♙|♙
**1**|♖|♘|♗|♕|♔|♗|♘|♖
```

Title: [8/22/2012] Challenge #90 [difficult] (Alien P=NP reductions)
Text: Ancient aliens have come down from earth and announced that, clearly, P=NP, and they've been trying to explain it to us for years. Something about it being encoded in the pyramids they helped us build...or...something.

In order to help human scientific progress, the aliens have begun distributing crystal alien technology nodes that are oracle solutions to NP-complete problems. Despite being very strange looking, they have USB cable ports and provide an API for all human computer languages and operating systems. This API provides a single
function "alien_knapsack()" which takes in an instance of the knapsack problem and immediately returns the solution (it uses alient time-travel technology or something).

Write a program demonstrating how you can solve your favorite NP-complete problem by calling "alien_knapsack" as a black-box. You can pick any problem you want and show how to reduce it to an instance of the knapsack problem. If you are finding your code difficult to test, then see the bonus problem.

BONUS: Write an implementation of the alien_knapsack() function in a human programming language. Obviously this does not have to be a polynomial-time solution.

Title: [8/20/2012] Challenge #89 [difficult] (Coloring the United States of America)
Text: On wikipedia, you can find [this lovely blank map](http://en.wikipedia.org/wiki/File:Blank_US_Map.svg) of the United States. What makes it so lovely? Well, first off all, it's in the [SVG format](http://en.wikipedia.org/wiki/File:Blank_US_Map.svg), which means that you can download it and edit it quite easily, even with a computer program you write yourself (SVG is nothing but XML, after all).

Second, the kind mapmaker has gone to the extra trouble of labelling all the states in the code with their proper abbreviation (so "AR" for "Arkansas" and "MT" for "Montana"). For instance, look at the source file for that image, and you'll see that the first path that is defined has the id "HI", so we know it represents Hawaii.

By parsing that file, noting the "id" attributes of the various "path" tags, you can then change the color of a specific state by changing the "style" attribute. For instance, if we change Hawaii's style attribute from

```
    fill:#d3d3d3;stroke:#ffffff;stroke-opacity:1;stroke-width:0.75;stroke-miterlimit:4;stroke-dasharray:none
```

to

```
    fill:#ff0000;stroke:#ffffff;stroke-opacity:1;stroke-width:0.75;stroke-miterlimit:4;stroke-dasharray:none
```

then Hawaii will stand out as bright red.

Your task today is to write a program that will read in that SVG file, then assign colors to all the different US states, such that no states that share a border has the same color. [Here is an example](http://i.imgur.com/QN3sG.png). You don't have to figure out what states border which other states from the SVG file, you can just put that as a table in your code, or use any other solution you can come up with.

If you finish, please upload your image, or a PNG of your image, to imgur, so the rest of us can see what it looks like!

Bonus: By the [4-color theorem](http://en.wikipedia.org/wiki/4-color_theorem) all maps like this can be colored using at most 4 colors, so that no two regions that share a border have the same color. Color the map using only four different colors.

NOTE: Look out for Michigan! Michigan is tricky.

***

Edit: to make it easier for everyone, [here's a list of what states borders other states](http://pastebin.com/uNJAEfgr). I compiled it myself, so I can't guarantee accuracy (though I'm fairly sure it's accurate, and it works fine in my program). To be clear, a line like

```
    ND <- MN, SD, MT
```

Means that North Dakota borders Minnesota, South Dakota and Montana.

Title: [8/13/2012] Challenge #88 [difficult] (ASCII art)

Text: Write a program that given an image file, produces an ASCII-art version of that image. Try it out on [Snoo](http://i.imgur.com/tJmB9.png) (note that the background is transparent, not white). There's no requirement that the ASCII-art be particularly good, it only needs to be good enough so that you recognize the original image in there.

Title: [8/10/2012] Challenge #87 [difficult] (Sokoban game)
Text: [Sokoban](http://en.wikipedia.org/wiki/Sokoban) is an old PC puzzle game that involves pushing boxes onto goal squares in a puzzling warehouse layout. Write your own simple Sokoban clone (using a GUI, or curses) that can read level files in [.xsb format](http://sokosolve.sourceforge.net/FileFormatXSB.html) from the command line and play them.

For extra credit, extend your program to include a level editor, allowing the user to draw his own levels and save them as .xsb files.

Title: [8/8/2012] Challenge #86 [difficult] (2-SAT)
Text: Boolean Satisfiability problems are problems where we wish to find solutions to boolean equations such as

```
    (x_1 or not x_3) and (x_2 or x_3) and (x_1 or not x_2) = true
```

These problems are notoriously difficult, and k-SAT where k (the number of variables in an or expression) is 3 or higher is known to be
NP-complete.

However, [2-SAT](http://en.wikipedia.org/wiki/2-satisfiability) instances (like the problem above) are NOT NP-complete (if P!=NP), and even have linear time solutions.

You can encode an instance of 2-SAT as a list of pairs of integers by letting the integer represent which variable is in the expression, with a negative integer representing the negation of that variable. For example, the problem above could be represented in list of pair of ints form as

    [(1,-3),(2,3),(1,-2)]

Write a function that can take in an instance of 2-SAT encoded as a list of pairs of integers and return a boolean for whether or not there are any true solutions to the formula.

Title: [8/3/2012] Challenge #85 [difficult] (Bitwise arithmetic)
Text: While basic arithmetic operations like addition, subtraction, multiplication, etc. seem 'natural' to us, computers think on a very different level: under the hood, computations work with [bitwise operations](http://en.wikipedia.org/wiki/Bitwise_operation), using operators like `~`, `&`, `|`, `^`, and `<<`. Your task is to implement functions for (integer) **addition**, **subtraction**, **negation**, **multiplication**, **division**, **modulo**, and **exponentiation**, without using any "high-level" operators like `+` and `*`. Other statements like "if" and "while", or recursion for functional languages, are fine.

As a hint, you could start with a helper function that increments a binary number, and work from there. Your functions should take signed integer arguments and return signed integer values, rounding down (e.g. `binary_div(14, 4) == 3`, not `3.5`).

Use your set of functions to implement [this function](http://i.imgur.com/ENkWO.png):

    f(a, b, c, d, e) = (a % e) * (b / (c - a) + exp(d * (-b), e))

What is the result of `f(50, -40, 300, 2, 3)`?

Title: [8/1/2012] Challenge #84 [difficult] (City-Block TSP)
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"

In this text adventure, there are N (<=10) 'interest points' on an infinite 2-D grid. The player (as described in the 'easy' challenge) can move one unit at a time on this grid towards one of these interest points.
The interest point they are told to move to is chosen at random. They also start at a random interest point.   It is important to note that the player cannot move diagonally in any sense, the player must move parallel to one of the axis.

Given a set of interest points specified as 2-D integer coordinates, what is the minimum number of steps a player could take to get to them all and win the game? (order doesn't matter).
What is the maximum number of steps? Assume the player heads optimally towards whichever interest point is randomly chosen until he gets it.

For reference, my solution produces a maximum of 1963 steps and a minimum of 617 steps on this input:

```
62 -67
4 -71
-86 71
-25 -53
-23 70
46 -34
-92 -62
-15 89
100 42
-4 43
```

EDIT: To clarify this a bit, what I mean is that you want to find the 'best possible game' and 'worst possible game'. So this min/max is over all possible starting locations. Also, you do not have to get back to the starting point.

Title: [7/30/2012] Challenge #83 [difficult] (Digits of the square-root of 2)
Text: The square-root of 2 is, as [Hippasus of Metapontum](http://en.wikipedia.org/wiki/Hippasus) discovered to his sorrow, irrational. Among other things, this means that its decimal expansion goes on forever and never repeats.

Here, for instance, [is the first 100000 digits](http://pastebin.com/tQ3NwP05) of the square-root of 2.

Except that it's not!

I, evil genius that I am, have changed exactly one of those 100000 digits to something else, so that it is slightly wrong. Write a program that finds what digit I changed, what I changed it from and what I changed it to.

Now, there are a number of places online where you can get a gigantic decimal expansion of sqrt(2), and the easiest way to solve this problem would be to simply load one of those files in as a string and compare it to this file, and the number would pop right out. But the point of this challenge is to try and do it with math, not the internet, so that solution is prohibited!

Title: [7/27/2012] Challenge #82 [difficult] (Bowling scores)
Text: Write a program that reads a series of space-separated bowling rolls from input, like this one:

```
10 7 3 7 2 9 1 10 10 10 2 3 6 4 7 3 3
```

Then calculates the scores for each frame according to the [scoring rules for ten-pin bowling](http://en.wikipedia.org/wiki/Ten-pin_bowling#Scoring). An example output for these rolls would be:

```
| 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10    |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-------|
| X   | 7 / | 7 2 | 9 / | X   | X   | X   | 2 3 | 6 / | 7 / 3 |
| 20  | 37  | 46  | 66  | 96  | 118 | 133 | 138 | 155 | 168   |
Total: 168
```

(You can format your output however you like. If you want, just outputting the scores for each frame (20, 37, 46...) is enough.)

Some other examples to test your program on:

```
10 10 10 10 10 10 10 10 10 10 10 10

| 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10    |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-------|
| X   | X   | X   | X   | X   | X   | X   | X   | X   | X X X |
| 30  | 60  | 90  | 120 | 150 | 180 | 210 | 240 | 270 | 300   |
Total: 300
```

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

| 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10    |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-------|
| X   | 9 / | 8 1 | 7 / | 5 2 | 8 1 | 4 / | 8 / | X   | 9 / 3 |
| 20  | 38  | 47  | 62  | 69  | 78  | 96  | 116 | 136 | 149   |
Total: 149
```

```
10 10 10 0 8 10 10 0 0 7 0 6 2 9 0

| 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10    |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-------|
| X   | X   | X   | 0 8 | X   | X   | 0 0 | 7 0 | 6 2 | 9 0   |
| 30  | 50  | 68  | 76  | 96  | 106 | 106 | 113 | 121 | 130   |
Total: 130
```

Title: [7/25/2012] Challenge #81 [difficult] (Matrix Exponential)
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 that can calculate the [Matrix Exponential](http://en.wikipedia.org/wiki/Matrix_exponential) for a 4x4 (or nxn) matrix. This function is extremely valuable for lots of different scientific areas.

There are [LOTS of ways to do it!](http://www.cs.cornell.edu/cv/researchpdf/19ways+.pdf)

For testing, here is a matrix.

```
 0.00000  -1.00000   3.00000   0.50000
 1.00000   0.00000   0.45000   0.10000
-3.00000  -0.45000   0.00000   0.40000
-0.50000  -0.10000  -0.40000   0.00000
```

And the resulting matrix exponential (as computed by GNU Octave)

```
-0.9276446  -0.2437849  -0.2285533   0.1667568
-0.2809791   0.7661246   0.5148905   0.2626626
-0.0150871   0.5946104  -0.7613132  -0.2580951
 0.2455577  -0.0077772  -0.3210194   0.9146516
```

Title: [7/23/2012] Challenge #80 [difficult] (Multi-word anagrams)
Text: In [today's easy problem](http://www.reddit.com/r/dailyprogrammer/comments/x0v3e/7232012_challenge_80_easy_anagrams/), we investigated anagrams that were single words. However, as is clear in the "I am Lord Voldemort" and "Tom Marvolo Riddle" example, anagrams can also be several words long.

Your difficult task today is to write a program that given a word will generate all multi-word anagrams of that word. Use [the same dictionary as in the easy problem](http://code.google.com/p/dotnetperls-controls/downloads/detail?name=enable1.txt).

So, for instance, the word "PARLIAMENT" has (by my count) ~~6636~~ 8438 multi-word anagrams using that dictionary. Examples include "MENIAL PRAT", "INEPT ALARM", "EAT NIL PRAM" (most of them will not make any sense) and "PARLIAMENT" itself. Note that in this problem, if the difference between two permutation is only word order, they count as the same anagram. So "INEPT ALARM" and "ALARM INEPT" should just count as one anagram.

Also, if there are single-word anagrams of the input, they should be counted in the total. For instance, in the 63 (again, by my count) multi-word anagrams of "MARBLES", the words "AMBLERS", "BLAMERS", "LAMBERS" and "RAMBLES" are included, as well as "MARBLES" itself (a few examples of multi-word anagrams for "MARBLES" are "ARM BELS", "REM LABS" and "ELM BARS").

How many multi-word anagrams is there for "CARPENTER" and "INHERITANCE"?

Title: [7/18/2012] Challenge #79 [difficult] (Remove C comments)
Text: In the C programming language, comments are written in two different ways:

* `/* ... */`: block notation, across multiple lines.
* `// ...`: a single-line comment until the end of the line.

Write a program that removes these comments from an input file, replacing them by **a single space character**, but also handles **strings** correctly. Strings are delimited by a `"` character, and `\"` is skipped over. For example:

```
  int /* comment */ foo() { }
→ int   foo() { }

  void/*blahblahblah*/bar() { for(;;) } // line comment
→ void bar() { for(;;) }

  { /*here*/ "but", "/*not here*/ \" /*or here*/" } // strings
→ {   "but", "/*not here*/ \" /*or here*/" }
```

Title: [7/18/2012] Challenge #78 [hard] (Maximum Forest Fire Fighting)
Text: You are an amateur computer scientist working at a mountain ranch for a summer retreat. This ranch
consists of very narrow trails winding up and down a small mountain. These trails form a series of interconnections
between various ranch buildings peppering the mountain. Because the trails are so narrow, only one person or vehicle can be occupying a given section
of trail at once. [Here is a map of the ranch.](http://i.imgur.com/ovGcc.jpg) ([original image](http://www.flickr.com/photos/acb/4290387040/sizes/l/) used with permission under cc-by-sa-nd)

All of a sudden a building at Sunset Peak on the far outskirts of the ranch catches fire from some honeymooners leaving the oven on! If the fire isn't put out quickly, a forest fire could start!

Although only base camp and lakeside station have access to running water, all the camps have an (for this purpose bottomless) [water buffalo](http://olive-drab.com/idphoto/id_photos_m149.php)
permenantly parked at each building to deposit water in.

A convoy of 8 fast humans travels 3.5 miles per hour and can carry 4 gallons of water for each person.

A truck can travel at an average speed of 10 miles per hour (country mountain roads) and can carry 100 gallons of water.

A train car can travel at an average speed of 30 miles per hour and can carry 250 gallons of water.

What is the maximum gallons of water per hour that can get to Sunset Peak to put out the fire?

EDIT: To save you guys some busywork, I made a datafile out of the image. Assigning each location to a numeric code like so:

0	Basecamp
1	Lakeside Lodge
2	Logging Camp
3	Radio Tower
4	RV Park
5	Creekfront Lodge
6	North Camp A
7	Mt. Greensparrow
8	Aviary
9	Greers Drop
10	North Camp B
11	Ranger Valley
12	Old Mine

13 Lookout Lodge
14 Sunset Peak

I then was able to list the bidirectional graph as a list of connections like "node1 node2 weight type"

```
1 2 8.6 R
0 3 8.7 R
4 5 11.3 R
11 12 7.7 R
12 14 7.7 R
0 1 0.9 F
2 3 1.1 F
3 5 5.1 F
4 8 7.5 F
9 12 7.9 F
7 11 5.6 F
6 11 6.4 F
11 13 3.9 F
5 6 2.0 F
0 4 8.3 T
3 5 5.2 T
4 9 7.6 T
8 9 4.8 T
7 8 5.2 T
6 7 5.5 T
6 10 7.1 T
10 13 2.4 T
10 11 3.4 T
11 14 7.9 T
13 14 4.5 T
```

This should save you some time implementing stuff.

HARD BONUS: Imagine now that you have a limit on people and trucks (trains are stuck to their track, so those are fine).
You only have 5 trucks and 50 people. A truck takes two people to drive (safety in pairs!), a train requires 3 people to operate, (engineer, coalman, radioman)
and people cannot legally travel on foot in groups of less than 2. How will you allocate people to get there fastest? You are allowed to have people do multiple things at different times, like
hike to a spot and then drive a truck if you want to make your answer super complicated.

Title: [7/16/2012] Challenge #77 [difficult] (Boggle solver)
Text: Write a program that is able to find all words in a [Boggle](http://en.wikipedia.org/wiki/Boggle) board. For a word list, you can [use this file](http://code.google.com/p/dotnetperls-controls/downloads/detail?name=enable1.txt).

How many words can you find in the following 10x10 Boggle board?

```
T N L E P I A C N M
T R E H O C F E I W
S C E R E D A M E A
A O M O G O O F I E
G A C M H N L E R X
T D E R J T F O A S
I T A R T I N H N T
R N L Y I V S C R T
A X E P S S H A C F
```

I U I I I A I W T T

Title: [7/13/2012] Challenge #76 [difficult] (imgur album downloader)
Text: Write a script that takes an [imgur](http://imgur.com) album id and an output directory as command line arguments (e.g., `./script DeOSG ./images`), and saves all images from the album in the output directory as `DeOSG-1.jpg`, `DeOSG-2.jpg`, etc.

Hint: To retrieve the picture URLs, parse the HTML page at "`http://imgur.com/a/`**(ID)**`/layout/blog`".

Title: [7/12/2012] Challenge #75 [difficult] (C Preprocessor)
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.

For the difficult challenge, we are going to look at the C Pre-Processor. The C pre-processor is a program (implemented as a compilation pass in gcc, but it used to be a standalone program) that reads in text files that have been annotated with special 'commands'. It interprets these commands and uses them to output a transformed version of the text file with no annotations or comments. For some examples, look [here](http://www.cs.utah.edu/dept/old/texinfo/cpp/cpp.html#SEC2) and [here](http://en.wikipedia.org/wiki/C_preprocessor)

Your task is to implement as much of the C preprocessor as you can in as few lines as you can. Obviously getting an implementation fully-conformant with the full specification is very difficult, so try to get the most useful and obvious stuff as close as possible first. At least try to implement comments, #ifdef, #ifndef, #else, #elif, #endif, #define for constants, and #include . Don't kill yourself worrying about studying the spec for strange corner cases, just implement it as close as you can..this is for fun and learning, remember!

More complex stuff like #if and defined() and full macro with arguments support, and other things is a bonus.

As an example, consider this input file:

```
    #define DOGNAME Brian
    The following phrase is often used in web design:
    #define FRENCH
    #ifdef FRENCH //if french
    Le renard brun saute par-dessus le chien paresseux. Le nom du chien est DOGNAME.
    #else
    The brown fox jumped over the lazy dog.  The dog's name is DOGNAME.
    #endif
```

should output

```
    The following phrase is often used in web design:
    Le renard brun saute par-dessus le chien paresseux. Le nom du chien est Brian.
```

Title: [7/9/2012] Challenge #74 [difficult]
Text: Using the data-structure described in [today's intermediate problem](http://www.reddit.com/r/dailyprogrammer/comments/wa10w/792012_challenge_74_intermediate/), [follow Knuth's paper](http://arxiv.org/abs/cs/0011047) and make a complete implementation of Dancing Links that is able to solve exact cover problems.

Bonus: make a Sudoku solver using your implemented algorithm. If you need help converting Sudoku into an exact cover problem, see this [wikipedia article](http://en.wikipedia.org/wiki/Exact_cover#Sudoku).

Title: [7/6/2012] Challenge #73 [difficult]
Text: [Huffman coding](http://en.wikipedia.org/wiki/Huffman_coding) is a compression algorithm. Implementing it is a good occasion to work with queues, trees and bits.

Say we have a string of characters, and we want to transmit it over a network. To that end, we're gonna compress it.

The idea of the Huffman encoding is to replace each character by a bit sequence whose length depends on the frequency of occurrence of the character in the string: if a character occurs very often, we want to represent it by a very short bit sequence to avoid wasting space, but if appears only once or twice, it doesn't really matter if the bit sequence is long.

Exercise:

1. Write a function that takes a string and returns a Huffman tree, as described in the Wikipedia article.

2. Write an encoding function that takes a string and returns a sequence of bits that correspond to its Huffman encoding.

3. Write a decoding function that takes a sequence of bits and a Huffman tree, and reconstructs the original string.

Notice that you need the tree to decode a message. Bonus points if you figure out a way to encode the tree along with the bit sequence.

Also, don't let the gigantic introduction in the Wikipedia article discourage you, an algorithm is explained [here](http://en.wikipedia.org/wiki/Huffman_coding#Basic_technique). There's even a cute animation!

Title: [7/4/2012] Challenge #72 [difficult]
Text: The singular value decomposition (SVD) is an extremely useful algorithm in linear algebra. Some people have called it a 'master algorithm' because if you are able to perform the SVD you can perform almost every other useful linear algebraic operation.

Among other things, it is used to solve systems of linear equations, to find low-rank approximations, and to do least-squares estimation and machine learning.

The definition of the SVD is as follows:
A vector u and v are left and right singular vectors of a real matrix M, if and only if

Mv=su and M'u=sv

for some real scalar s. (M' denotes M transposed). The SVD of M is the set of ALL singular tuples (u,s,v) for which this is true.

Write a function that, given a 2x2 real matrix M, can output ONE of the singular tuples of M...that is find ONE of the left and right singular vectors from the Singular Value Decomposition (SVD) of M.

What makes this challenge VERY difficult is that you should not use a linear-algebra built-in function from a mathematical package to solve it..try to only use simple arithmetic operations such as matrix multiply, dot product,
squareroot, etc.

Since there are dozens of ways to solve it, here are 3 different hints about properties of the SVD corresponding to 3 different solutions I have written.

Imagine a function f(u,v)=u'Mv . If u,v, are singular vectors of M, then u,v are extrema of f and f(u,v)=s.

Imagine the matrix A=[0 M;M' 0]; If x is an eigenvector of A, then x*sqrt(2)=[u;v]

Since Mv=su and M'u=sv, then M'Mv=M'us=ssv, therefore M'Mv=ssv, so v is an eigenvector of M'M with eigenvalue s^2. By a similar argument, u is an eigenvector of MM' with eigenvalue s^2.

For some testing, the matrix [1 0;1 -1] has these two svd tuples according to GNU Octave: (u=[-0.52573;-0.85065],s=1.61803,v=[-0.85065;0.52573]) and (u=[-0.85065;0.52573],s=0.61803,v=[-0.52573,-0.85065])

EDIT: as Ttl points out, if you get the opposite sign from the test case on the vectors in a tuple it doesn't matter, the answer is still correct.

Title: [7/2/2012] Challenge #71 [difficult]
Text:

In 1987, mathematician John Conway invented one of the most curious programming languages ever, which he dubbed [FRACTRAN](http://en.wikipedia.org/wiki/Fractran).

A FRACTRAN program is simply a series of fractions, nothing more, nothing less. As input, the program takes a single integer. The program runs like this:

1. The integer is checked against each fraction in order. If the result of multiplying that integer with the fraction is another integer, you start over with the product generated by multiplying with that fraction.

2. If none of the fractions multiplied by the input integer results in another integer, the program finishes and returns the integer as the result.

Conway was able to show that despite the simplicity of this "programming language", it is in fact Turing-complete, meaning that any computation you can do in any other language, you can do in FRACTRAN.

The [wikipedia article for FRACTRAN](http://en.wikipedia.org/wiki/Fractran) explains very well how this works and how to write a program in FRACTRAN.

Your task is to first of all write a FRACTRAN interpreter that is able to run FRACTRAN programs (and remember that the numbers can very easily get very large, so 64-bit integers is not going to be enough, you need big numbers) and then to write a program in FRACTRAN. Here are a few suggestions of programs you could write, roughly ordered from least difficult to most difficult:

1. Implement the min(a,b) function. So for input 2^a * 3^b the code returns 5^min(a,b) where min(a,b) is the smallest number of a and b. Example: input 5832 should output 125 (2^3 * 3^6 -> 5^3)

2. Implement the max(a,b) function. So for input 2^a * 3^b the code returns 5^max(a,b) where max(a,b) is the largest number of a and b. Example: input 5832 should output 15625 (2^3 * 3^6 -> 5^6)

3. Write a program that takes an input a that is divisible by 3 and divides it by 3. So for input 2^a it returns 3^a/3 . Example: input 2097152 should output 2187 (2^21 -> 3^7). Note: this can be done in a single fraction.

4. Write a program that for an input n, returns the sum of all integers less than n. So if the input is 2^5, it should output 3^1+2+3+4 = 3^10. Example: input 32 should output 59049 (2^5 -> 3^10)

5. Write a program that generates the nth fibonacci number. So for input 2^n it should output 3^f(n) where f(n) is the nth fibonacci number. Example: input 128 should output 1594323 (2^7 -> 3^13).

Title: [6/29/2012] Challenge #70 [difficult]
Text: In today's challenge we will be touching the topics:

* [Strong Pseudoprime](http://mathworld.wolfram.com/StrongPseudoprime.html)

* [Fermat Pseudoprime](http://mathworld.wolfram.com/FermatPseudoprime.html)

* [Carmichael Number](http://mathworld.wolfram.com/CarmichaelNumber.html) (for bonus)

Fermat's primality test consists of choosing many numbers and checking if they are witnesses to the compositeness of the number being tested.

There are some composite numbers which pass Fermat's primality test for all possible witnesses; they are called Carmichael numbers

Because there exist numbers that fool Fermat's primality test for all bases, a strong pseudo-primality test is often used

Your tasks are

* to write two functions that test if a number is a Fermat pseudo-prime or a strong pseudo-prime to a given base

* two functions that test primality using the Fermat and strong pseudo-prime tests.

Bonus:

Write two functions that test if a number is a Carmichael number, and to identify all the Carmichael numbers less than a given input number by the user.

Title: [6/26/2012] Challenge #69 [difficult]
Text: The ADFGVX cipher described in [today's intermediate problem](http://www.reddit.com/r/dailyprogrammer/comments/vmbnb/6262012_challenge_69_intermediate/) turned out to be quite a challenge for the Allied powers, but it was finally cracked by the French cryptanalyst Georges Painvin. It is said that the work was so difficult and involved such complex techniques that it made him physically break down from the fatigue. His method required lots of similar messages encrypted during periods of high traffic.

Today we have an advantage that Painvin didn't have: massive computational power at our fingertips. Your difficult task for the day is to try and crack a message that have been encrypted using the ADFGVX cipher as defined in today's intermediate problem, without knowing either of the keys. To make the task a little bit easier, I'll give you a few hints regarding the message and how it's been encrypted:

1. The cleartext is in English
2. The substitution key is not any random permutation of the alphabet, it is simply a [caesar shift](http://en.wikipedia.org/wiki/Caesar_shift) of the alphabet. So, for instance, it could be 'BCDEFGHIKLMNOPQRSTUVWXYZ0123456789 A' (a caesar shift of 1) or 'CDEFGHIKLMNOPQRSTUVWXYZ0123456789 AB' (a caesar shift of 2), etc. etc.
3. The transposition key is an english word less than 13 characters long.

Those hints should be enough for you to be able to crack it, but if no one has succeeded in 24 hours, I'll give a few more hints. If you have an idea about how to solve it, but isn't quite sure, I encourage you to post your thoughts so others can see them and maybe develop on them.

Here is the ciphertext:

```
VGVFFDXFAAVVXGAFAVAAGFAVAFAGVGXXAAVFGVAXGGVVGGXFGVGDVGXGVAGGGDXDVFGFAFVGGVAGGFG
DGFAXGAGAXGVGAAGDGFXDXVAGVVAXGVGVGFVVAGXDVDGAGDGFXVGXAXGGXGVXGVVDGDGGXDADGGVGAV
AGGDXDXVGGGVGXGFXFXFGVXXAFGDGXDAXXAGVFVVAVVGGVDVXAXAFDGVDXVAAVXAFVFGVAXVFADDAVF
AFGVGVGFDVADVDVVXGXGGDVGXGVFGGAVADAFVFAGGFXXGFGVXFVFXAVFGFAFGVGFAVAAVFGFVAAVVVG
```

```
    DVAAXVGXDGDGGAGVDVGXDAXADGXGGADGFAFXGXVXVVVAFXXVFVVAAGVXXGGXADGAGFXFGFAGVFVVAFA
    DXDGGGDGXAFADGDAFVDGFVGAGGXGFXDXGGFXXAGADXXAVXFXFAAXGVVAGVFVVVGVFDDVGDFVGXFXDXF
    DXVGVGAVXDDFAFVFDGVFGFGDGGVGXGVFAFGFVGVXVGVGXFVVXGGDAXGFADAXGFVVVFDFAFAFAFVFVGG
    GDGGFGFVFXXVFADVFXXXFVVXDAFADGFVGGFGGAFXXXGVFGFGDXFAVAFVFXDGVVXXDXFGFXDVXVFXXGX
    GGAAXFAFGVXFXDAFGFGGGXADVGAGXVGFXFVVGDGDAGVGGGVFVGGXGGVFAGVDXFDXAXGGVDAFAFAFVAG
    GADXGXDVFXFVGGDAXXFAXXFVVGVVGGFXFXGXGVFADGVDXFXGFXXFVGVFGFAGGGVFVDVFVFGDVGGVAGX
    XVDGFAAVDGXGFXGVAGFGGGFGGVVGDXGAFXFGVDXAAVFAFVGDGVFADVGVFAVGAXFADGGXFXFGDAVGFXF
    AGGXXFVGXGAAVGVDAXGAGDXFGXVFVXVFGFVDXAGGXFAGAFXGAVVAADXVXGXXVGXDVXVFXVADXDVAXGF
    FVDGGVDGVXDGGDDXFVXGDVVGVGDGXAFXFGVAXVFVFXFXDVDVXVFVGGVADVXAGGAXFAGGXAXAFAGXGGX
    GVXDXGVFVFAVXVVFGVAFAFXFAGDXVGGVAXAGVFAFVVXFXGXFGFAVVFVDGGAFGVAGVXVDAGXGGGGDXFV
    FAFVFAXGDVGGFAFVFVFAFDXFVFXVVGADXFGVAFGGVFAAVXVDXFGVGFXFAGGGAGVFVGVGGDGXXDGVXGA
    DAVXGVDAFVVGGVGXDGFXGXDXGXGAFXFXVGDGDVFDGADADVFVXADGVAVXAADVGVDAGXAAFAVAGXGVGVF
    XGVDGDGFAGVFAXXDGVAGVGVXAFVFVFADXFADAXXDVGXFGDAXDDAXGGXGFAVGFDGXXAXVDAGVFGGXFGG
    XGAGXFVGGXAFVDAGXXXFXVXDXXAVXFXDAFVGVGGGADGGGGADAXGDVDGXVDDXVGAVAGVDVVAFXFVDAGX
    VGFXFXDGAVXAFVFAFVDVGGVXFVGVDXXGAGAVFVXVGXFXFXFADXGXFGVVVVFXDGFXFXDGDGGGAVFXFAA
    VGFADVFVDGXGGAGGFXVGFVVVFADXDVXGFVDVGVDVDGVXFXVVDGXAGGGGFVGGGVFXFVDAGGVVVAGXGGD
    ADAVADGDVDXXAGADGXXGGFVGADXFGFVDAGGDVFAGAFXGVGGVVVAFGXGXGGGFVGXGAGXAGDGAVFXDGXV
    GVVVGVGGVGVXDGDGXVXVDXDAVGDGDXGXGVDVVGFXFXXGDXFGDDDAVXDAAXAGXVFVVAGXDXGXGVFGFVA
    VXXFGGXFDVXAVDGDAGAGXXXFAXGXVDVFAGADDGXDGDGFVFXGGFAXDXGXVFGFAGGFVVXDGVGDGXGXGGV
    XGFVVXGVXXVVFVGGFVDGDXGAFGFXFVFAGGFAVADAVGGDDVFXFXGVFXGVXXGXGXGDGXAGXAFGGGGFVFV
    GAXADGDGVXVAFVFAGAFXGAGGDXXGDGGDGXVVFVDGFGGVAGGGDVXXFXDVVGFXVXVGDVVGFXVGGXFVFAX
    AFXVGDGXXDVGVGXGAAGGGDAFVFGGGFAGVVGVVXAXAFVDGGGFXGDXGVVVGGVFADGVAGADXDGFVGVGXGX
    DXAGFGGGGVGXFD
```

Good luck!

Title: [6/22/2012] Challenge #68 [difficult]
Text: Implement a program you can use to play the classic game of chess. White and black alternate inputting their moves using some form of [chess notation](http://en.wikipedia.org/wiki/Chess_notation). The computer checks if the moves are legal and if so, executes them. The program should be able to tell whenever a player is in check or check-mate. You can represent the chessboard in the terminal in ascii form.

Bonus: implement a simple AI that can play chess against you.

Title: [6/20/2012] Challenge #67 [difficult]
Text: Let the s(N) be a random number generator defined as follows (at this point, this should probably be anointed the Offical Random Number Generator of /r/dailyprogrammer):

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

Let Q(N) be an NxN two-dimensional matrix of the first N^2 values of this random number generator. That is, Q(5) would be:

```
    123456789   752880530   826085747  576968456   721429729
    173957326   1031077599  407299684  67656429    96549194
    1048156299  663035648   604085049  1017819398  325233271
    942914780   664359365   770319362  52838563    720059384
    472459921   662187582   163882767  987977812   394465693
```

Now, the task is to select exactly one element from each row and each column (so that no column or row has more than one element selected) in such a way that the sum of those elements is at a minimum. For instance, for Q(5) above, we would select the following elements (the selected elements marked with asterisks):

```
    *123456789* 752880530   826085747   576968456   721429729
```

```
   173957326   1031077599  407299684   67656429    *96549194*
   1048156299  *663035648* 604085049   1017819398  325233271
   942914780   664359365   770319362   *52838563*  720059384
   472459921   662187582   *163882767* 987977812   394465693
```

The sum of those elements is 123456789 + 663035648 + 163882767 + 52838563 + 96549194 = 1099762961 which is the smallest we can do with this matrix. Lets call this number M(X), i.e. M(X) is the smallest sum of elements selected from a square matrix X such that each row and each column has exactly one element selected. Then M(Q(5)) = 1099762961

Write a program that finds M(Q(20)).

Title: [6/18/2012] Challenge #66 [difficult]
Text: Today's difficult problem is similar to challenge #64's difficult problem

Baseball is very famous in the USA. Your task is write a program that retrieves the current statistic for a requested team. [THIS](http://www.baseball-reference.com/) site is to be used for the reference. You are also encouraged to retrieve some more information from the site .. just use your creativity! :D

Bonus: Stock prices can be retrieved from [this site](http://finance.yahoo.com/) ... your task is to retrieve the current price of a requested company.

Title: [6/15/2012] Challenge #65 [difficult]
Text: A magic square is a square of size NxN with the numbers 1 through n^2 put in so that all rows, all columns and both diagonals sum to the same number. For instance, this is a 3x3 magic square:

```
   8 1 6
   3 5 7
   4 9 2
```

As you can see all rows, all columns and both diagonals (8+5+2 and 4+5+6) sum to the same number, 15.

Write a program that draws a magic square of size 18x18.

Title: [6/13/2012] Challenge #64 [difficult]
Text: One of the sites where daily weather forecasts are shown is [here](http://weather.noaa.gov/pub/data/forecasts/.)

Get the forecast of any asked city and also try to be more innovative in your answers

Title: [6/11/2012] Challenge #64 [difficult]
Text: Find a way to sort the list in today's [intermediate problem](http://www.reddit.com/r/dailyprogrammer/comments/uw16v/6112012_challenge_63_intermediate/) using less than 19000 calls to reverse(N, A).

Title: [6/8/2012] Challenge #62 [difficult]
Text: Write a program to solve 9x9 Sudoku puzzles.

Title: [6/6/2012] Challenge #61 [difficult]
Text: As is well known, the decimal expansion of sqrt(N) when N is not a perfect square, goes on forever and does not repeat. For instance, sqrt(8) starts out 2.82842712... and never starts repeating itself. This is because when N is not a perfect square, [sqrt(N) is irrational](http://en.wikipedia.org/wiki/Infinite_descent#Irrationality_of_.E2.88.9Ak_if_it_is_not_an_integer) and [all numbers with repeating decimals are rational](http://en.wikipedia.org/wiki/Repeating_decimal#Every_repeating_or_terminating_decimal_is_a_rational_number).

However, if instead of using a decimal representation, you use a [continued fraction representation](http://en.wikipedia.org/wiki/Continued_fraction) of sqrt(N) when N is not a perfect square, then it will always have a repeating period. For instance, [this is the beginning of the continued fraction of sqrt(8)](http://i.imgur.com/WWIFJ.gif). The pattern of 1,4,1,4,1,4,... will repeat forever (the first integer, the 2, is not included in the period). A continued fraction with a period like this can be written as [a; [b,c,d,...]], where a is the first number outside of the fraction, and b, c, d, etc. are the period repeating inside the fraction. For example, sqrt(8) has a continued fraction representation of [2; [1,4]].

Here are some other continued fraction representations of square roots:

```
    sqrt(2) = [1; [2]]
    sqrt(13) = [3; [1, 1, 1, 1, 6]]
    sqrt(19) = [4; [2, 1, 3, 1, 2, 8]]
    sqrt(26) = [5; [10]]
```

Let Q(N) be the sum of the numbers in the period for the continued fraction representation of sqrt(N). So Q(19) = 2 + 1 + 3 + 1 + 2 + 8 = 17 and Q(26) = 10. When N is a perfect square, Q(N) is defined to be 0.

The sum of Q(N) for 1 &le; N &le; 100 is 1780.

What is the sum of Q(N) for 1 &le; N &le; 50000?

***

Bonus: If your code for solving this problem includes use of the sqrt() function, solve [today's intermediate problem](http://www.reddit.com/r/dailyprogrammer/comments/uo14v/662012_challenge_61_intermediate/) and use your own implementation of sqrt().

Title: [6/4/2012] Challenge #60 [difficult]
Text: The basic idea of RSA starts with two large prime numbers of equal bit-length, p and q; their product n becomes the modulus of the cryptosystem. The totient of n is computed as ϕ(pq) = (p−1) × (q−1). Then two keys are chosen, the encryption key e and the decryption key d, such that de ≡ 1 (mod ϕ(pq)) and gcd(e, ϕ(pq)) = 1. Then, given a message m, an integer on the range 0 < m <n, the message is encrypted by computing m^e (mod n) and the resulting cipher-text c is decrypted by computing c^d (mod n).

The standard definition of RSA cryptography is known as [PKCS #1](http://www.rsa.com/rsalabs/node.asp?id=2125). It provides a method for converting a text message to a number m suitable for encryption, and converting it back to the original text message. It is also possible to use RSA to provide non-forgeable signatures; the basic idea is that the sender encrypts a message hash with his decryption key, so the receiver can decrypt the message hash with the sender’s public key, which works because only the sender knows his private decryption key.

Your task is to write an RSA key generator and procedures to encrypt and decrypt messages using the RSA algorithm.

Here is the [RSA wiki](http://en.wikipedia.org/wiki/Rsa) to help you in understanding.

Title: [6/2/2012] Challenge #59 [difficult]
Text: Two strings A and B are said to have a *common substring* called C, if C is embedded somewhere in both A and B. For instance, "ble" is a common substring for "Dou**ble**, double, toil and trouble" and "Fire burn and cauldron bub**ble**" (because, as you can see, "ble" is part of both "Double" and "Bubble"). It is, however, not the longest common substring, the longest common substring is " and " (5 characters long for vs. 3 characters long for "ble").

Define two pseudo-random number generators, P(N) and Q(N), like this:

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

    Q(0) = 987654321
    Q(N) = (22695477 * Q(N-1) + 12345) mod 1073741824

Thus, they're basically the same except for having different seed values. Now, define SP(N) to be the first N values of P concatenated together and made into a string. Similarly, define SQ(N) as the first N values of Q concatenated together and made into a string. So, for instance:

    SP(4) = "123456789752880530826085747576968456"
    SQ(4) = "987654321858507998535614511204763124"

The longest common substring for SP(30) and SQ(30) is "65693".

Find the longest common substring of SP(200) and SQ(200)

***

BONUS: Find the longest common substring of SP(4000) and SQ(4000).

Title: [5/28/2012] Challenge #58 [difficult]
Text: A type of pseudo-random number generator is the so-called [lagged fibonacci generator](http://en.wikipedia.org/wiki/Lagged_fibonacci_generator), which has become somewhat popular because it is very simple to implement, can have an extremely long period, and produces high quality random numbers.

The idea is this: to calculate s(n) (i.e. the nth random number), you evaluate:

    s(n) = (s(n - a) + s(n - b)) mod M

For some positive constants a and b (it is thus similar to the fibonacci numbers, but it "lags" behind) and some modulus M. One popular choice for a and b is a = 24 and b = 55. Lets use those numbers and a modulus of 1073741824 (i.e. 2^30), and the generator becomes:

    s(n) = (s(n - 24) + s(n - 55)) mod 1073741824

In order for this formula to work, you need to initialize the values s(0),s(1),...,s(54), so that the recursion has somewhere to bottom out. Often, another random number generator is used to supply the inital values. Lets use the random number generator from the [intermediate challenge #53](http://www.reddit.com/r/dailyprogrammer/comments/tpxqc/5162012_challenge_53_intermediate/).

That is to say, for values s(0) through s(54), s is defined as follows:

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

But for values s(55) and above, s is defined as follows:

    s(n) = (s(n - 24) + s(n - 55)) mod 1073741824

Here are a few example values:

```
s(10)     = 1048156299
s(20)     = 472459921
s(55)     = 827614689
s(56)     = 530449927
s(100)    = 515277845
s(1000)   = 985063932
s(10000)  = 304605728
s(100000) = 434136346
```

Find s(10^18)

Title: [5/25/2012] Challenge #57 [difficult]
Text: Lets play some games shall we! .. for many of the next challenges i will be giving old classic games to be programmed.

Today your task is to implement [Hamurabi](http://atariarchives.org/basicgames/showpage.php?page=78). the link gives data required in implementing. Enjoy! :)

____

Edit: [Here](http://pastebin.com/LvsZHGTd) is the basic code for making things easier.

Title: [5/23/2012] Challenge #56 [difficult]
Text: Arguably the worlds most famous fractal is the so-called [Mandelbrot set](http://en.wikipedia.org/wiki/Mandelbrot_set), but even though many people have seen it, very few know the actual definition. Which is a shame, since the definition is actually quite simple.

The Mandelbrot set is the set of all complex numbers C such that the expression Z(n) = Z(n-1)^2 + C (with Z(0) = 0) remains bounded for all n, as n goes to infinity. The famous "squashed bug" image of the Mandelbrot set is simply all such points C plotted in the [complex plane](http://en.wikipedia.org/wiki/Complex_plane), i.e. a graph with the x-axis representing the real part of C and the y-axis the imaginary part of C.

Your task today is to create a program that will draw the Mandelbrot set. Your program does not need to include colors, it is fine to simply have black and white pixels indicating or not that pixel is part of the Mandelbrot set.

However you want to output it is fine. You can save it as an image file, draw it on the screen, make an HTML5 webpage, whatever you like. Hell, you can even draw a really low-def version of it on the terminal if you really want to. If you do save it as an image file, I encourage you to upload it to imgur so that the rest of us can see your creation. [Here](http://imgur.com/MQl8y) is my feeble attempt.

If you need more details on the Mandelbrot set, I highly encourge visiting the [wikipedia page](http://en.wikipedia.org/wiki/Mandelbrot_set), as it provides an excellent discussion of it.

Title: [5/21/2012] Challenge #55 [difficult]
Text: Write a program to implement the [Pollard's rho algorithm](http://en.wikipedia.org/wiki/Pollard's_rho_algorithm) using both, [Floyd’s cycle-finding algorithm](http://en.wikipedia.org/wiki/Cycle_detection) and [Brent’s cycle-finding algorithm](http://en.wikipedia.org/wiki/Cycle_detection#Brent.27s_algorithm) ..

Bonus: also compare the timings for the implementation of both the algorithms and come up stating whichever is the fastest.

Title: [5/19/2012] Challenge #54 [difficult]
Text: For this challenge, make a program that finds the [determinant of a square matrix](http://en.wikipedia.org/wiki/Determinant#Calculation). You do not need to use any exceedingly complex methods, using expansion by cofactors is entirely sufficient (if you have no idea what that is, the always helpful Khan Academy [here](http://www.khanacademy.org/math/linear-algebra/v/linear-algebra--3x3-determinant) to [help](http://www.khanacademy.org/math/linear-algebra/v/linear-algebra--nxn-determinant). Wikipedia [also](http://en.wikipedia.org/wiki/Laplace_expansion) has you [covered](http://en.wikipedia.org/wiki/Cofactor_(linear_algebra\))).

What is the determinant of the following matrix?

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

Note: the whole purpose of this challenge is to write the code for calculating the determinant yourself. Therefore, you are not allowed to use library functions that are designed to calculate things like this. The matrix should be represented by the native array construct of your language, not any special data type designed to operate on mathematical matrices. That means no NumPy, no using fancy functions in Mathematica or Matlab! You are allowed to use these to test whether or not your function works, but that's it.

Title: [5/16/2012] Challenge #53 [difficult]
Text: The set {2,3,5,7,11} has 32 subsets, and if you multiply the elements of each subset together, you get the "product" of that specific subset. So for instance, {2,5,7} is a subset of {2,3,5,7,11} and it has the product 70 (i.e. 2*5*7).

The subset of {2,3,5,7,11} with the largest product that *does not exceed 100* is {7,11}, with the product 77.

Given a set *s* and a number *v*, define A(s,v) as the subset of s with the largest product that does not exceed v. Also, define p(n) as the set of the first n primes (thus p(5) is equal to {2,3,5,7,11}). Here are some examples of A(p(n), v):

A(p(5), 100) = {7, 11}
A(p(7), 1000) = {5, 11, 17}
A(p(8), 2000) = {3, 5, 7, 19}
A(p(10), 10000000) = {2, 5, 7, 11, 19, 23, 29}

Find A(p(20), 10^18)

***

BONUS: Find A(p(40), 10^60)

***

NOTES: If it is more convienient, you are allowed to make your A(s,v) function output the product instead of the subset itself, so A(p(5), 100) = 77 instead of {7,11}. Watch out though, the numbers can get very big.

Title: [5/14/2012] Challenge #52 [difficult]
Text: Your task is to write functions that encrypt and decrypt using the [solitaire cipher](http://www.schneier.com/solitaire.html).

Title: [5/11/2012] Challenge #51 [difficult]
Text: Take a 7x7 grid of cells and remove the central cell (like a chessboard, but slightly smaller and with a hole in the middle), and it would look something [like this](http://i.imgur.com/UXtTA.png). The number of cells is 7*7 - 1 = 48 because we removed the central cell.

Now, lets start tiling this grid with dominoes. Each domino covers exactly two cells that are either horizontally or vertically next to each other, so if you are going to tile the whole
thing with dominoes, you would need 24 of them (48 over 2). [Here is an example](http://i.imgur.com/NmD8m.png) of the grid being perfectly tiled by dominoes. There are exactly 75272 ways you can use dominoes to tile a 7x7 grid with the central cell removed.

Find the last 8 digits of the number of ways you can use dominoes to tile a 15x15 grid with the central cell removed.

Note: rotations and reflections of tilings count as distinct tilings. I.e. if two tilings differ only by rotation or reflection, they are still considered to be different.

Title: [5/9/2012] Challenge #50 [difficult]
Text: T9 Spelling: The Latin alphabet contains 26 characters and telephones only have ten digits on the keypad. We would like to make it easier to write a message to your friend using a sequence of keypresses to indicate the desired characters. The letters are mapped onto the digits as 2=ABC, 3=DEF, 4=GHI, 5=JKL, 6=MNO, 7=PQRS, 8=TUV, 9=WXYZ. To insert the character B for instance, the program would press 22. In order to insert two characters in sequence from the same key, the user must pause before pressing the key a second time. The space character should be printed to indicate a pause. For example “2 2” indicates AA whereas “22” indicates B. Each message will consist of only lowercase characters a-z and space characters. Pressing zero emits a space. For instance, the message “hi” is encoded as “44 444”, “yes” is encoded as “999337777”, “foo bar” (note two spaces) is encoded as “333666 6660022 2777”, and “hello world” is encoded as “4433555 555666096667775553”.

This challenge has been taken from [Google Code Jam Qualification Round Africa 2010](http://code.google.com/codejam/contest/dashboard?c=351101#s=p2) ... Please use the link for clarifications. Thank You

Title: [5/7/2012] Challenge #49 [difficult]
Text: When you roll two regular six-sided dice, the total number of pips that can come up ranges from 2 (if both dice show 1) to 12 (if both dice show 6), but as all experienced gamblers know, some numbers are more likely than others. In fact, the most likely number to come up is 7, with a probability of 1/6. By contrast, the probability of 12 showing is only 1/36, so it is six times more likely that the dice will show 7 than it is that they will show 12.

The reason for this is of course that there are more ways that two dice can sum to 7. In fact, there are exactly six ways two dice can sum to 7: the first die can show 1 and the second 6, the first 2 and the second 5, the first 3 and the second 4, the first 4 and the second 3, the first 5 and the second 2, and finally the first die can show 6 and the second 1. Given that there are a total of 6*6 = 36 different ways the dice can land, this gives us the probability: 6/36 = 1/6. In contrast, there is only one way two dice can form 12, by throwing two sixes.

Define a function f(d, n) that gives the number of ways d six-sided dice can be thrown to show the number n. So, in the previous example, f(2,7) = 6. Here are a few other values of that function:

f(1,n) = 1 (for 1&le;n&le;6, 0 otherwise)

f(2,7) = 6
f(2,10) = 3
f(2,12) = 1
f(3,10) = 27
f(5,20) = 651
f(7,30) = 12117
f(10,50) = 85228

Find f(20, 100)

Note: the answer fits into a 64-bit integer
***

Bonus: Find f(1100, 5000) mod 10^7

Title: [5/4/2012] Challenge #48 [difficult]
Text: Inspired by the divide and conquer problem posted on 4/16/12, here's another problem that can be solved using divide and conquer. You are given a grid of vertices/nodes connected by adjacent nodes in a checkerboard manner (basically think of the intersection points of the grids on a piece of graphing paper) and each of these nodes is marked with a positive number. Assuming that these numbers are distinct, give an algorithm that can find a single local minimum.

Bonus: If there are n^2 nodes in our grid, give an O(n) time complexity algorithm that can find one/any local minimum.
In other words, if the construction of the grid takes some time c^* n^2, find an algorithm that locates any local minimum by looking at only some constant factor times the square root of the total number of nodes there are in the grid.

Hint: divide into quadrants; the recurrence T(n) = 1*T(n/4) + O(n) is (maybe somewhat counter-intuitively) in O(n).

Title: [5/2/2012] Challenge #47 [difficult]
Text: If you were to generate all permutations of the first three letters of the alphabet ("a", "b" and "c") and then sort them, you would get the following list of 6 permutations:

1. abc
2. acb
3. bac
4. bca
5. cab
6. cba

As you can see, the fourth permutation in a sorted list of all the permutations of "a", "b" and "c" is "bca".

Similarly, if we wanted the 30th permutation in a sorted list of all permutations of the first five letters of the alphabet (i.e. "abcde"), you get "baedc".

Define a function f(n,p) that generates the permutation number p in a sorted list of all permutations of the n first letters of the alphabet. So, for instance:

f(3, 4) = "bca"
f(5, 30) = "baedc"
f(7, 1000) = "bdcfega"
f(8, 20000) = "dhfebagc"

Find f(11, 20000000)

***
Bonus:

Find f(20, 10^18)

Title: [4/30/2012] Challenge #46 [difficult]
Text: The prime HP reached starting from a number , concatenating its prime factors, and repeating until a prime is reached. If you have doubts, refer the article [here](http://mathworld.wolfram.com/HomePrime.html)

write a function to calculate the HP of a given number.

Also write a function to compute the [Euclid-Mullin sequence](http://mathworld.wolfram.com/Euclid-MullinSequence.html).

Title: [4/27/2012] Challenge #45 [difficult]
Text: If you list all positive integers less than or equal to 20, you get this:

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

If you count the number of times the digit "1" appears in that list, you get 12. Define a function f(n) which gives the number of 1's needed to write out all numbers 1 to n in decimal notation.

Here is a couple of example values for f(n):

    f(20) = 12
    f(1234) = 689
    f(5123) = 2557
    f(70000) = 38000
    f(123321) = 93395
    f(3^35) = 90051450678399649

By the way, all numbers in this problem, inputs and outputs alike, fit into 64-bit integers.

Can you implement f(n) in a way that is faster than just listing the values less than n and counting the instances of 1?

What is f(5^20)?

***

Bonus: A curious thing happens when you try and calculate f(35199981). You get 35199981, the same number you put in. You can prove that the number of times f(n) is equal to n is finite. What is the sum of all n such that f(n) == n?

***

Since this problem is harder than most problems here, I'll give you two hints for solving it.

Hint for calculating f(n):

    The numbers f(10^n - 1) (i.e. f(9), f(99), f(999), f(9999), ...) all follow a very regular pattern that is
    trivial to evaluate. If you figure out this pattern (it's not very complicated), you should
    be able to figure out what (for instance) f(99999) is instantaneously.
    Once you have f(99999), what is f(100000)? What is f(200000)? What is f(299999)?

Hint for the bonus:

    There are 83 such numbers, and they are all less than 10^11

Good luck!

Title: [4/25/2012] Challenge #44 [difficult]
Text: Write a function that takes two arguments a and b, and finds all primes that are between a and a + b (specifically, find all primes p such that a &le; p &lt; a + b). So for instance, for a = 1234 and b = 100, it should return the following 15 primes:

1237, 1249, 1259, 1277, 1279, 1283, 1289, 1291, 1297, 1301, 1303, 1307, 1319, 1321, 1327

The sum of those primes are 19339. The number of primes for a = 9! and b = 8! is 3124 and the sum of those primes is 1196464560.

How many primes are there for a = 16! and b = 10!, and what is their sum?

***

Note 1: In this problem, n! refers to the [factorial](http://en.wikipedia.org/wiki/Factorial) of n, 1*2*3*...*(n-1)*n, so 9! is equal to 362880.

Note 2: All numbers and solutions in this problem fit into 64-bit integers.

Title: [4/24/2012] Challenge #43 [difficult]
Text: I wouldn't call this exactly a difficult question .. but it is a fun one :)

You are all familiar with the game [snake and ladders](http://en.wikipedia.org/wiki/Snakes_and_Ladders)

This is the [Board](http://imgur.com/6DyXQ) you are to refer.

Your task is to write programs that will answer the following questions

First, what is the minimum number of rolls required to reach space 100.

Second, for a single player, what is the average number of rolls required to reach space 100.

And third, for k players, what is the average number of rolls until one of the players reaches space 100 and wins the game.

Note: Space 100 must be reached by exact roll of the die; if the roll of the die would take the token past space 100, the token remains where it is and play passes to the next player. The winner of the game is the first token to reach space 100.

These are some of the questions from the paper "How Long Is a Game of Snakes and Ladders?" by S. C. Althoen, L. King and K. Schilling

Title: [4/23/2012] Challenge #42 [difficult]
Text: A *simple graph* is a graph that allows only one edge between two vertices and no edges that start end end in the same vertex (usually called "loops"). If a simple graph has V vertices, the maximum number of edges in that graph is V*(V-1)/2, which happens when every vertex is connected to every other vertex by an edge. This is called a *complete graph*.

A graph is said to be *connected* if you can get from any vertex to any other vertex by travelling on the edges. The minimum number of edges required for a graph of V vertices to be connected is V - 1 (a graph with the minimum number of vertices needed to be connected is called a *tree*).

To demonstrate the concept of connectedness, let's say that you have a simple graph of six vertices named 0,1,2,3,4 and 5, which has the following edges: (0,1), (2,3), (3,4) and (4,2). This graph is not connected, because there is no way to (for instance) get from vertex 1 to vertex 2, or from vertex 3 to vertex 5. In fact, if you drew the graph on a piece of paper, you'd see that it forms three "islands", with 0 and 1 on one island connected by the edge (0,1); 2,3 and 4 on one island connected in a triangle; and 5 on island of its own, connected to no other vertex.

Let's say that you had a simple graph with 50 vertices named 0,1,2,3,....,48,49 and the following 45 edges:

```
( 0,22),( 0,39),( 0,47),( 1, 5),( 2,38),( 2,39),( 2,44),( 4,23),( 4,33)
( 5,19),( 6,17),( 6,35),( 6,45),( 7,49),( 8,24),( 8,40),( 9,25),( 7,10)
(10,11),(10,21),(11,30),(12,32),(13,27),(14,33),(14,36),(15,49),(16,48)
(18,37),(18,45),(19,24),(19,40),(20,39),(21,34),(25,36),(26,27),(26,31)
(26,32),(28,48),(29,36),(29,41),(35,43),(38,42),(39,44),(43,46),(48,49)
```

This graph is obviously not connected, because there aren't enough edges to meet the minimum requirement (you would need at least 49). Write a program that prints how many islands of vertices there are, and what vertices there are on each island. In other words, print out some number of sets of vertices where in each individual set all vertices are connected to each other, but no two vertices in different sets are.

Bonus: For a graph of 1000000 (one million) vertices, the number of possible edges is 100000\*999999/2 = 499999500000 (almost five hundred billion), but obviously not all edges are needed for the graph to be fully connected. Create a graph of 1000000 vertices and no edges, and then one by one start adding edges to it, randomly selected from all possible edges, until you have a connected graph. How many edges did you add before the graph became connected? (note: this number will obviously be different every time you run the program)

Title: [4/19/2012] Challenge #41 [difficult]
Text: The application you will be writing will be a caching server. This application will be monitoring a directory containing lots and lots of files (so you probably should first write a program that randomly generates lots of files from 1 byte to 1 MB each first before proceeding). I/O should probably be network-based, but standard I/O (keyboard and screen) can suffice. Input to this caching server will be a file name matching a file found in the directory that the caching server is monitoring. When input is requested, the caching server will read the file contents into memory, cache the file contents, and write out the contents as a response to output. When the request is made again on the same file, just send back the data that was cached in memory.

The caching server will not need to cache contents of a file if a request hasn't been made after awhile, so set an expiration for cached data to free up memory.

The caching server can run out of memory, so when you cache data, make sure you have enough memory before caching the data.

After a long period of time, certain files may not be accessed and we want to preserve disk space. So after a long period of time, compress unused files into one zip file and remove the file from the monitored directory. When a request is made on that file, unzip the contents and put them back into the directory.

Title: [4/16/2012] Challenge #40 [difficult]
Text: Make a function that generates an array of 1,000 2-dimensional points, where both the x-coordinate and the y-coordinate are between 0.0 and 1.0. So (0.735, 0.167) and (0.456, 0.054) would be examples.
(Most computer languages have a simple random function that returns a double precision floating point number in this range, so you can use that to generate the coordinates. Python has random.random(), Java has Math.random(), Perl has rand(), etc.)

Create a program that finds the two points in this array that are closest to each other, and print that distance.
As a reminder, the distance between the two points (x1, y1) and (x2, y2) is sqrt((x1 - x2)^2 + (y1 - y2)^2).

Bonus 1: Do the same thing, but instead of using 1,000 points, use 1,000,000 points and see if you can create an algorithm that runs in a reasonable amount of time [edit: something like one minute or less].

Bonus 2: Do the same thing but for 3-dimensional points.

Title: [4/12/2012] Challenge #39 [difficult]
Text: Given a list of n words, create a program that can solve a word search. The word search would be a 2-dimensional array of characters varying in sizes.

BONUS: Give it the ability to solve [Snaking Puzzles](http://en.wikipedia.org/wiki/Word_search#Snaking_puzzles)

Title: [4/10/2012] Challenge #38 [difficult]
Text: Write a function that tests whether large numbers are prime or not, with extremely high certainty. There are several primality tests that can do this. Fairly simple ones include the [Fermat Test](http://en.wikipedia.org/wiki/Fermat_primality_test) and the even better [Miller-Rabin test.](http://en.wikipedia.org/wiki/Miller%E2%80%93Rabin_primality_test) The Wikipedia articles have pseudocode you can implement.

Use your function and a random number generator to post a 100-digit prime. You can [test your result at Wolfram|Alpha.](http://www.wolframalpha.com/input/?i=factor+341362016887055569382051876093823801537125554910414027673552798511083444632381620792969350674025567)

Title: [4/8/2012] Challenge #37 [difficult]
Text: Your task is to implement [Cornacchia's algorithm](http://en.wikipedia.org/wiki/Cornacchia's_algorithm) and use it to demonstrate that all primes of the form 4k+1 and less than 1000 can be written as the sum of two squares.

source: [programmingpraxis.com](http://programmingpraxis.com/2012/04/03/cornacchias-algorithm/)

Title: [4/5/2012] Challenge #36 [difficult]
Text: Let's play Lingo! Click [here](http://www.youtube.com/watch?feature=player_detailpage&v=HzWh_G0DiGw#t=113s) for an idea of how the game works. Now write a program that reads a random 5-letter word from a dictionary file and plays the game Lingo. If you're doing a text-based version, you can surround the correct letters at the correct location with [] and correct letters at the wrong location with ().

Title: [4/3/2012] Challenge #35 [difficult]
Text: The objective of this exercise is to maintain a list of Strings in memory that support undo and redo. Write a program that allows the user to add, edit, delete, **undo**, and **redo** entries in a list. You must be able to undo and redo everything you've done during the execution of this program. After each command is run, always print out the list (unless you're doing this in a UI). Before writing any code, first think about how to write add, edit, and remove with undo and redo in mind. If there are no submissions to this post, I'll reply with some hints.

Sample Run:

```
Enter command ('A'dd, 'E'dit, 'D'elete, 'U'ndo, 'R'edo): A

Enter text to add:  Venus
Venus

Enter command ('A'dd, 'E'dit, 'D'elete, 'U'ndo, 'R'edo): A

Enter text to add:  Mars
Venus
Mars

Enter command ('A'dd, 'E'dit, 'D'elete, 'U'ndo, 'R'edo): U
Venus
```

```
Enter command ('A'dd, 'E'dit, 'D'elete, 'U'ndo, 'R'edo): U

Enter command ('A'dd, 'E'dit, 'D'elete, 'U'ndo, 'R'edo): R
Venus

Enter command ('A'dd, 'E'dit, 'D'elete, 'U'ndo, 'R'edo): R
Venus
Mars

Enter command ('A'dd, 'E'dit, 'D'elete, 'U'ndo, 'R'edo): A

Enter text to add:  Saturn
Venus
Mars
Saturn

Enter command ('A'dd, 'E'dit, 'D'elete, 'U'ndo, 'R'edo): E

Enter index to edit:  1

Enter text to edit:  Earth
Venus
Earth
Saturn

Enter command ('A'dd, 'E'dit, 'D'elete, 'U'ndo, 'R'edo): U
Venus
Mars
Saturn


Enter command ('A'dd, 'E'dit, 'D'elete, 'U'ndo, 'R'edo): R
Venus
Earth
Saturn


Enter command ('A'dd, 'E'dit, 'D'elete, 'U'ndo, 'R'edo): D

Enter index to delete:  2
Venus
Earth


Enter command ('A'dd, 'E'dit, 'D'elete, 'U'ndo, 'R'edo): U
Venus
Earth
Saturn


Enter command ('A'dd, 'E'dit, 'D'elete, 'U'ndo, 'R'edo): R
Venus
Earth
```

Title: [3/31/2012] Challenge #34 [difficult]
Text: Inspired by the restaurant I ate at the other day. This is the puzzle: You have a wooden triangle, roughly equilateral with 5 rows of holes. The top row has one hole, and the bottom has 5, increasing by one hole with each successive row.

One hole of the triangle is empty and the rest are filled with golf tees. To make a move, you jump a golf tee over another adjacent golf tee into a hole immediately beyond it, removing that second golf tee from the game. Your goal is to find a solution set of jumps such that all of the golf tees but one are removed from the board. The notation of such a solution is at your discretion.

Bonus: Investigate if the choice of initial empty hole influences the solvability of the problem, and if so, what is the maximum number of pegs that can be removed given each starting hole.

Title: [3/30/2012] Challenge #33 [difficult]
Text: Travelling Salesman problem.

There are N cities numbered from 0..N-1. A salesman is located at city 0. He wishes to visit all cities exactly once and return back to city 0. There are K toll booths. Each toll booth has a certain range of functioning. The parameters for toll k are given as x_k and y_k. If the salesman travels from city i to j, he has to pay 1 dollar toll fee to each toll p having x_p >= i and y_p <= j. Calculate the cheapest way for the salesman to complete his tour.

Input :

The first line contains T the number of test cases. T test cases follow. The first line of each test case contains two space seperated integers N and K. Each of the next K lines contains 2 integers, the ith line containing x_i and y_i (0 <= x_i,y_i < N). A blank line seperates two test cases.

Output :

Output T lines, one for each test case, containing the required answer.

Edit: Also since some people have problems, here are two solutions .. just see it for reference!
[link1](http://pastebin.com/j7aMRj8M) [link2](http://pastebin.com/FgVSZYYF)

Title: [3/28/2012] Challenge #32 [difficult]
Text: A quine is a computer program which takes no input and produces a copy of its own source code as its only output which, in turn, compiles and print out itself

[Hint](http://en.wikipedia.org/wiki/Quine_(computing))

Title: [3/27/2012] Challenge #31 [difficult]
Text: In this challenge, given an array of integers, the goal is to efficiently find the subarray that has the greatest value when all of its elements are summed together. Note that because some elements of the array may be negative, the problem is not solved by simply picking the start and end elements of the array to be the subarrray, and summing the entire array.
For example, given the array

{1, 2, -5, 4, -3, 2}

The maximum sum of a subarray is 4. It is possible for the subarray to be zero elements in length (if every element of the array were negative).

Try to come up with an efficient algorithm!

Title: [3/26/2012] Challenge #30 [difficult]
Text: The Fibonacci numbers are defined recursively as

f(0) = 0
f(1) = 1
f(n) = f(n-2) + f(n-1)

Find the last eight digits of f(10^18).

If you have some computer science and/or discrete math training, this is not very difficult, but if you don't it can be really tricky. You can't just write a for-loop to calculate Fibonacci numbers one by one (and you certainly can't simply implement the recursive definition directly). That for-loop would have to run a quintillion times, and by the time it's finished, the sun will probably have exploded. You have to be more clever than that.

I should add that the identity you need to solve this problem is on the Wikipedia page for Fibonacci numbers. Using that identity and another algorithm solves the problem instantly (my Python program gives the answer in less than 0.1 seconds).

Title: [3/22/2012] Challenge #29 [difficult]
Text: Draw a line... except it must be your implementation of a line using only the ability to draw a point. Think implementing a line is too easy? Try it :). You can output the result in ASCII text if you'd like instead of using a graphics library. A successful implementation will be able to draw [this](http://imgur.com/a8LuR). Only being able to draw horizontal, vertical, and diagonal lines is not enough, and the lines can't contain any holes. Also, if you're drawing a line (I'll use drawLine(x1, y1, x2, y2) as an example) using the following call: drawLine(100, 10, 200, 300), then the following call must draw the same line: drawLine(200, 300, 100, 10).

Title: [3/20/2012] Challenge #28 [difficult]
Text: The idea is simple. Use the [pastebin](http://pastebin.com/login.php?ref=L2FwaQ==) API (wrappers should not be allowed) in the most creative way to create a cool command line tool. A simple example that's very easy to implement in most modern programming languages is a program that posts to pastebin the contents of a given file. A few ideas for extra features:

* The ability to post a whole directory to pastebin with one command.
* The option to post only a part of a file
* Tweeting the link to twitter when posting
* Language recognition for the filename
* A history of recent pastes with their links
* Automatic pasting every few minutes (or after a file changes) for backup

Title: [3/17/2012] Challenge #27 [difficult]
Text: Write a program that will perform date/time addition. Input can be interactive using standard input or command line. 3 parameters are required: a whole number, a unit of time using the following values: year, month, week, day, hour, minute, second, and a date and time (you choose the date format). The result will be the new date and time.

Example (using ISO 8601 date/time format without time zone designator):

```
Enter a date and time (YYYY-MM-DDThh:mm:ss):  2012-03-17T09:00:00
Enter value to add:  10
Enter unit of time (year, month, week, day, hour, minute, second):   minute
New date and time is 2012-03-17T09:10:00


Enter a date and time (YYYY-MM-DDThh:mm:ss):  2012-11-29T15:00:00
Enter value to add:  5
Enter unit of time (year, month, week, day, hour, minute, second):   day
New date and time is 2012-12-04T15:00:00


Enter a date and time (YYYY-MM-DDThh:mm:ss):  2012-06-01T09:00:00
Enter value to add:  -2
Enter unit of time (year, month, week, day, hour, minute, second):   week
New date and time is 2012-05-18T09:00:00
```

Title: [3/16/2012] Challenge #26 [difficult]
Text: Create a piece of code that downloads an entire album from imgur. It should support multiple arguments. e.g.

```
   imgurDownloader http://imgur.com/a/0NZBe http://imgur.com/a/OKduw
```

The url might get a bit redundant for large batch's, so consider leaving out the link, so one just needs to enter 0NZBe OKduw ect. You can use a third party librarys.

Tip: every single image link in an album is listed in the source code.

Bonus points: You can enter the directory you would like to save the files but it's optional. Extra bonus points: if you can change all the file-names into something readable, that's also customizable. For example if I wanted all my images called wallpapers_001, wallpapers_002, ect. I would just add wallpapers_# as an argument.

Title: [3/15/2012] Challenge #25 [difficult]
Text: Write a program that places N queens on an NxN chessboard such that no two queens are on the same row, column, or diagonal, and no queen is on either of the two major diagonals (corner to corner). Get a solution for as large a value of N as you can.

* [Sample valid board for N = 8](http://pastebin.com/5JYt1XND)
* [Sample valid board for N = 48](http://pastebin.com/nwgdf8rk)

Title: [3/13/2012] Challenge #23 [difficult]
Text: Sort a given set of strings based on a unique collating sequence for each position in a string. Given N collating sequences, to sort strings of length greater than N, sequence i mod N is used at character position i.

For example, consider the three collating sequences:
collating sequence 0 is: ASCII-order-ignore-case
collating sequence 1 is: reverse-ASCII-order
collating sequence 2 is: a-z 0-9 ASCII-order A-Z

In this example the strings

The Cat in the Hat
the Rain in Spain
The RAIN in Spain
Beavis and Butthead

Note that the last ordering says lower case comes before digits; and digits before everything not upper case; and upper case follows all.

The allowable notations for collating sequences are:

ASCII-order
ASCII-order-ignore-case
reverse-ASCII-order
reverse-ASCII-order-ignore-case
a-z
A-Z
0-9

These can occur in any order without repetition.

Input will be in the form:

N
description of collating sequence 1
..
..
description of collating sequence N
line 1
line 2
..
..
line unknown number

So for the given example, the input would look like:

3
ASCII-order-ignore-case
reverse-ASCII-order
a-z 0-9 ASCII-order A-Z
The Cat in the Hat
the Rain in Spain
The RAIN in Spain
Beavis and Butthead

Title: [3/10/2012] Challenge #22 [difficult]
Text: For todays challenge, write a maze generator.

For extra credit, write a second program which can solve the maze.

Title: [3/9/2012] Challenge #21 [difficult]
Text: We'd like to write a list of people, ordered so that no one appears in the list before anyone he or she is less smart than.

The input will be a list of pairs of names, one pair per line, where the first element in a pair names a person smarter than the person named by the second element of the pair. That is, each input line looks like:

smarter-person : less-smart-person

For example:

Einstein : Feynmann
Feynmann : Gell-Mann
Gell-Mann : Thorne

Einstein : Lorentz
Lorentz : Planck
Hilbert : Noether
Poincare : Noether

There is no limit to the number of lines of input. Your output should be a list of all the distinct input names, without duplicates, one per line, ordered as described above. For example, given the input shown above, one valid output would be:

Einstein
Feynmann
Gell-Mann
Thorne
Lorentz
Planck
Hilbert
Poincare
Noether

Note that the "smarter than" relation supplied as input will not, in general, specify a total order that would allow us to write out the desired list in strictly decreasing order of smartness. For example, the following output is also valid for the input shown above:

Hilbert
Einstein
Feynmann
Gell-Mann
Poincare
Thorne
Lorentz
Planck
Noether

Title: [3/8/2012] Challenge #20 [difficult]
Text: create a program that will remind you to stop procrastinating every two hours with a pop up message! :)

This program has the potential of helping many people :D

Title: [3/7/2012] Challenge #19 [difficult]
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).

Write a program that will build and output a word index for The Adventures of Sherlock Holmes. Assume one page contains 40 lines of text as formatted from Project Gutenberg's site. There are common words like "the", "a", "it" that will probably appear on almost every page, so do not display words that occur more than 100 times.

Example Output: the word "abhorrent" appears once on page 1, and the word "medical" appears on multiple pages, so the output for this word would look like:

abhorrent: 1

medical: 34, 97, 98, 130, 160

Exclude the Project Gutenberg header and footer, book title, story titles, and chapters.

Title: [3/5/2012] Challenge #18 [difficult]
Text: Write a program that draws a [square spiral](http://10binary.deviantart.com/art/square-spiral-203786602). You can print out this spiral in ASCII text, but using a graphics library would produce a more pleasant output.

Bonus: Now draw a normal spiral. Some samples of spirals can be found [here](http://images.google.com/search?tbm=isch&hl=en&source=hp&biw=1920&bih=987&q=spiral&gbv=2&oq=spiral&aq=f&aqi=g10&aql=&gs_sm=3&gs_upl=2312l3239l0l3992l6l6l0l1l1l0l104l429l4.1l5l0).

Title: [3/4/2012] Challenge #17 [difficult]
Text: build a tic tac toe game with opponent AI

bonus points for cooler AI implementations (depth/breadth first search, neural network, etc)

Title: [2/27/2012] Challenge #16 [difficult]
Text: You all know about [O'Neill's algorithm](http://www.cs.hmc.edu/~oneill/papers/Sieve-JFP.pdf)

write a program such that you compute primes for a given input by the user using it.

edit: if anyone has any suggestions for the subreddit, kindly post it in the feedback thread posted a day before. It will be easier to assess. Thank you.

Title: [2/24/2012] Challenge #15 [difficult]
Text: Write a pair of programs that communicate with one another through socket connections. AKA a client-server connection.

Your server should be an echo server that simply echoes any information it receives back to the client.

For bonus points, your server should take a special command that will echo the subsequent information in reverse.

Title: [2/23/2012] Challenge #14 [difficult]
Text: Write a program that will generate a random array/collection of 1 million integers, then sort them using a multi-threaded algorithm.

Your program should take the number of threads through standard input.

Bonus points if you can find the most efficient number of threads for your program.

Title: [2/21/2012] Challenge #13 [difficult]
Text: Create a rock-paper-scissors program, however, there should be no user input. the computer should play against itself. Make the program keep score, and for extra credit, give the option to "weigh" the chances, so one AI will one more often.

Title: [2/20/2012] Challenge #12 [difficult]
Text: Write a program which will take string inputs "A", "B", "C", "D", "E", "F", and "G", and make the corresponding notes, in any method of your choosing.

Title: [2/19/2012] Challenge #11 [difficult]
Text: Create a program which prints out a table with the month's calendar in it, when the month and year is given as input.

Extra points for highlighting the current day and providing links to next and previous months.

Happy coding :)

Title: [2/18/2012] Challenge #10 [difficult]
Text: Your task is to implement the interactive game of [hangman](http://en.wikipedia.org/wiki/Hangman_\(game\))

bonus point for making the game unique. be more creative!

Title: [2/17/2012] Challenge #9 [difficult]
Text: The U.S government has commissioned you to catch the terrorists!

There is a mathematical pyramid with the following pattern:

1

11

21

1211

111221

312211

you must write a program to calculate up to the 40th line of this pyramid. If you don't, the terrorists win!

Title: [2/16/2012] Challenge #8 [difficult]
Text: Write a program that will take coordinates, and tell you the corresponding number in pascals triangle. For example:

Input: 1, 1

output:1

---

input: 4, 2

output: 3

---

input: 1, 19

output: error/nonexistent/whatever

---

the format should be "line number, integer number"

for extra credit, add a function to simply print the triangle, for the extra credit to count, it must print at least 15 lines.

Title: [2/15/2012] Challenge #7 [difficult]
Text: This challenge will focus on creating a bot that can log into Reddit!

Write a program that can log into a working Reddit account.

Since this challenge is vague, bonus points are awarded for a few different things:

* If the bot can navigate the site and view posts

* If the bot can make posts

* If the bot can make statistics from the front page/any subreddit. These statistics include time on front page, number of comments, upvotes, downvotes, etc.

The more functionality in your bot, the better.

Title: [2/14/2012] Challenge #6 [difficult]
Text: create a AI that will play [NIM](http://en.wikipedia.org/wiki/Nim)

Title: [2/13/2012] Challenge #5 [difficult]
Text: Arrr, me mateys! Yer' challenge fer' today be a tough one. It be gettin awfully borein' on the high seas, so yer' job be to create a pirate based fightin' game! This game oughter' be turn based, and you oughter' be able to pick yer attacks every turn. The best game'll be winnin' some custom flair, and all the rest o' ya will be walkin the plank!

Title: [2/12/2012] Challange #4 [difficult]
Text: today, your challenge is to create a program that will take a series of numbers (5, 3, 15), and find how those numbers can add, subtract, multiply, or divide in various ways to relate to eachother. This string of numbers should result in 5 * 3 = 15, or 15 /3 = 5, or 15/5 = 3. When you are done, test your numbers with the following strings:

4, 2, 8

6, 2, 12

6, 2, 3

9, 12, 108

4, 16, 64

For extra credit, have the program list all possible combinations.

for even more extra credit, allow the program to deal with strings of greater than three numbers. For example, an input of (3, 5, 5, 3) would be 3 * 5 = 15, 15/5 = 3. When you are finished, test them with the following strings.

2, 4, 6, 3

1, 1, 2, 3

4, 4, 3, 4

8, 4, 3, 6

9, 3, 1, 7

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

For this challenge, you need to write a program that will take the scrambled words from this post, and compare them against [THIS WORD LIST](http://pastebin.com/jSD873gL) to unscramble them. For bonus points, sort the words by length when you are finished. Post your programs and/or subroutines!

Here are your words to de-scramble:

`mkeart`
`sleewa`
`edcudls`
`iragoge`
`usrlsle`
`nalraoci`
`nsdeuto`
`amrhat`
`inknsy`
`iferkna`

Title: [difficult] challenge #2
Text: Your mission is to create a stopwatch program. this program should have start, stop, and lap options, and it should write out to a file to be viewed later.

Title: [difficult] Challenge #1
we all know the classic "guessing game" with higher or lower prompts. lets do a role reversal; you create a program that will guess numbers between 1-100, and respond appropriately based on whether users say that the number is too high or too low. Try to make a program that can guess your number based on user input and great code!