# 1991 Denison Spring Programming Contest
## Problem 1
## Mastermind

In Mastermind, the player tries to guess the position and color of four hidden pegs (called the *target*). The colors come from the set { red, blue, green, yellow, black and white }. The player makes a series of guesses - he places 4 colored pegs in four holes – and is given a response telling him how many pegs were the correct color *and* correct position and how many were the correct color but the wrong position.

For instance, if the target was red, white, green, black and the guess was red, green, black, yellow, then the response would be 1 correct color and position (since red is in the correct position) and 2 correct color but wrong position (since both green and black appear but in the wrong position).

In this problem you will be given 3 guesses and the responses. You should output all possible targets that could generate these responses for all 3 of the guesses, or print a message saying that no pattern is possible.

In this version, there will be no duplicate colors in the guesses provided or the targets you generate.

The format of the input will be 4 characters – representing the four pegs, in order – followed by two integers, the first giving the number of pegs with correct color and correct position, and the second giving the number of pegs with correct color but wrong position. The following uppercase characters will be used for the colors: R-red, G-green, Y-yellow, B-blue, K-black and W-white.

Your output should be all targets (in any order) that could give these three responses – if there are some – or a message that states there is no target possible. Each legal target should appear on its own line.

---

Example

The following input:

```
RKWB 1 2
RYGW 1 2
RKGB 1 1
```

should produce the following output (but not necessarily in this order)

```
RWYK
RWBY
RWKY
YKRW
WYRB
```

# 1991 Denison Spring Programming Contest
## Problem 2
## Formatting with Hyphenation

In this problem you will print out a paragraph of text formatted so that the length of each line is as close as possible to the specified page width without exceeding the width. In order to do this, you may be required to hyphenate some words according to the following rules.

A word may be hyphenated only between two consonants, except when the second consonant is an h, or between a consonant and a following vowel. For our purposes, a, e, i, o and u will be considered vowels. All other letters are consonants. Words may contain other characters besides letters. When hyphenating a word, neither the front part nor the back part of the hyphenated word is allowed to have 2 or fewer letters. So, for instance, the word 'organize' may be hyphenated only as 'org-anize' or 'organ-ize'. A sufficiently long word may stretch to more than two lines when hyphenated.

Input will consist of an integer on the first line, indicating the width of each line in the paragraph, and a list of words, one per line (the word will be terminated by the end-of-line). The maximum length of a word is 50 characters. A word will contain no white space. The list of words is terminated by end-of-file. The output should be the text formatted so to fill out each line as fully as possible using the hyphenating rules given above.

---

Example

The following input (stored one word per line but shown differently here to save space):

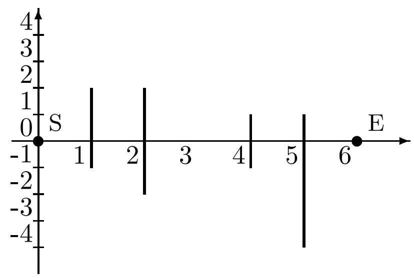
```
16
In many implementation schemes, code fragments are executed just before a reduction
takes
place.  Reductions provide a "hook" on which action consisting of arbitrary code
fragments can be hung.
```

should produce the following output:

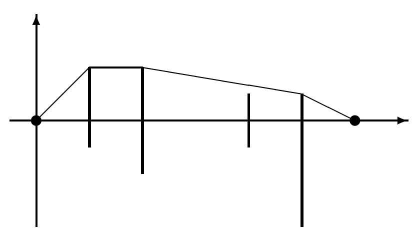
```
In many implem-
entation sch-
emes, code frag-
ments are exec-
uted just before
a reduction
takes place.
Reductions prov-
ide a "hook" on
which action
consisting of
arbitrary code
fragments can be
hung.
```

# Spring 1991 Denison Computer Science Contest
## Problem 3
## Line Barriers

In this problem you are to find the shortest path between two points located in the $x$-$y$ plane which have between 1 and 5 barriers between them. The starting point will always be located at the origin – point (0,0) – and the end point will lie at point (6,0) on the $x$ axis. The barriers will be vertical line segments that lie on the lines $x = 1$, $x = 2$, $\ldots$, $x = 5$, with at most one barrier on each line. For example, the figure below shows an example with 4 barriers (note: the $x$ and $y$ axis are not drawn to scale):




Your solution must be a minimum length path starting at point S, ending at point E and not crossing any of the barriers. The solution for this particular problem is shown below and has length 7.81.



In this problem you will process many input sets. In each set the first line will contain the number of barriers, $n$. (If $n = 0$, then there will be no more lines and your program should terminate.) The next $n$ lines will each contain 3 integers $X\ \ Y_1\ \ Y_2$, where $X =$ the $x$ location of barrier, and $Y_1, Y_2 =$ the $y$ extents of the barrier. The barriers are given in order of increasing $X$, and for each barrier, $Y_1 < Y_2$. The maximum value of $n$ will be 5, and all $Y$ values will lie between $-10$ and 10. For each input set, print out one line containing the length of the minimum path rounded to the nearest hundredth.

---

Example

The following input:

```
4
1 -1 2
2 -2 2
4 -1 1
5 -4 1
2
2 0 10
3 -10 -1
0
```

should produce the following output (note: the first input set represents the figure above):

```
7.81
6.00
```

# 1991 Denison Spring Programming Contest
## Problem 4
## Arithmetic Expressions

In this problem you will be given 5 integers: a "target" integer (which will be positive) and 4 other integers. Your job is to insert the correct combination of arithmetic operators – taken from addition, subtraction, multiplication and integer division – between the 4 integers to get as close to the target integer as possible without exceeding it. You will then print your solution in the form illustrated below. Multiplication and division have precedence over addition and subtraction. You may assume that there will be at least one combination of the 4 integers that will be less than or equal to the target. In the case where two or more expressions yield the same "best" solution, print any one of them.

In this problem you will process many input sets. Each set will consist of two lines. The first line will contain the target integer. (If this is zero there will be no second line and your program should terminate.) The second line will contain 4 integers. You may not change the order of these integers: the order in which they appear in the input line is the order they must appear in your output.

---

Example

The following input: `40`

```
5 6 18 3
100
35 -1 40 2
20
3 20 0 3
```

`0` should produce the following output (note that best combinations may not be unique): `The closest`

```
combination to 40 is 5 * 6 + 18 / 3 = 36
The closest combination to 100 is 35 - -1 + 40 + 2 = 78
The closest combination to 20 is 3 + 20 + 0 - 3 = 20
```

# Spring 1991 Denison Computer Science Contest
# Problem 5
# Code Decryption

A substitution encryption scheme is one in which each letter of the alphabet is substituted by another, unique letter. For example, an '`a`' may be represented by an '`f`', a '`b`' by a '`u`', a '`c`' by an '`n`', etc. In this problem, you will read in a set of 6-letter words, and then the same words again after they have been encrypted. The order of the words will not necessarily be the same in both sets. Given these two sets, you are to determine the substitution scheme that was used to convert words from the first set into those in the second set.

In this problem you will process many input sets. In each set, the first line will contain the number of words in each set, $n$. (If $n = 0$, then there will be no more lines and your program should terminate.) The next $n$ lines will contain the set of un-encrypted words, one per line, and the next $n$ lines will contain the set of encrypted words, one per line. The maximum value for $n$ will be 10, and all words will be in lower case. For output, you should print out two lines: the first will contain the standard alphabet from '`a`' to '`z`', and the second will contain the substitution for each letter. In the absence of any information concerning the encryption of a specific letter (because it did not appear in any of the words in the first set), use an '`*`' for its substitution.

WARNING: Do *not* attempt to solve this problem by trying to generate all possible substitution schemes to see which one works. This approach takes far too much execution time to be worthwhile for this contest.

---

Example

The following input (stored one word per line but shown differently here to save space):

```
6
decamp fright jovial larynx squawk zombie
czjmvu glipwd ksopab nxwyjl qxtjpr rpzfhe
2
pascal enigma
odnedp ciyrtd
0
```

should produce the following output:

```
abcdefghijklmnopqrstuvwxyz
pyiglcmvjqbrwhxdszkuotaefn

abcdefghijklmnopqrstuvwxyz
d*e*c*r*y**pti*o**n*******
```