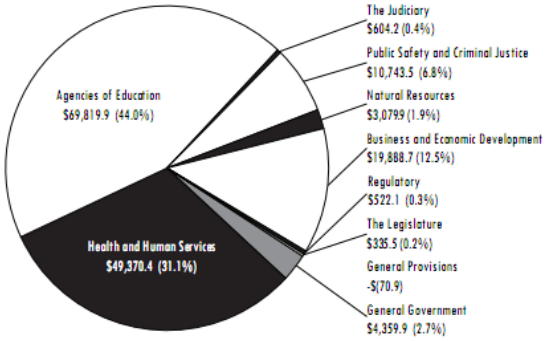
problem 1
Slash the Budget
2 points

JAVA program name must be prob01.java
C /C++ program name must be: prob01.exe

Introduction

You have recently been hired as a junior analyst for the 2nd Assistant Deputy to the Budget Director for the State of Texas. Huge budget shortfalls are expected for 2012, so efforts are underway to reduce the state budget to line up with expected revenues (yes, this is a fictional scenario). Your job is to write a program to accept values for expected state revenues and current budget estimates. Your program will output the projected budget shortfall for 2012.



Input

Your program should prompt for the current state budget estimate (an integer value, in billions) and expected state revenue (an integer value, in billions).

```
Enter projected 2012 state budget (in billions): 123
Enter projected 2012 state revenues (in billions): 105
```

Output

Your program should output the projected budget shortfall for the 2012 Texas state budget.

```
Estimated budget shortfall in 2012:  18 billion
```

problem 2
Divide by 11
3 points

JAVA program name must be prob02.java
C /C++ program name must be: prob02.exe

Introduction

In honor of 2011 – your task is to accept a positive integer as input and, using the below algorithm, determine if that integer is divisible by 11. This algorithm was originally devised in 1897 by Charles Dodgson, a.k.a. Lewis Carroll.

The algorithm:

> As long as the number under test still has at least two digits, form a new number by:
> - deleting the ones digit
> - subtracting the deleted digit from the remaining, shortened number
>
> If the remaining two-digit number is divisible by 11, then the original number is divisible by 11.

HINT: You may need to use a wide integer data type, such as "long" in Java.

Sample Input

```
161408196180
```

Sample Output

```
161408196180
16140819618
1614081953
161408192
16140817
1614074
161403
16137
1606
154
11

The number 161408196180 is divisible by 11.
```

problem 3
Provide Justification
4 points

JAVA program name must be prob03.java
C /C++ program name must be: prob03.exe

Introduction

In typesetting and page layout design, full justification is the alignment of text within a column so that the text aligns along both the left and right margins (see "Type alignment examples"). Your task is to provide full justification of section of text, given an output column width of 30 characters. Some rules:

- you must fit as many words on each line as possible, only pushing words down to the next line when necessary
- you cannot split up words, nor insert spaces into words
- you cannot concatenate words together – you must maintain at least one space between them
- excess spaces on a given line should be spread evenly between words on that line. If they cannot be spread exactly evenly, the larger "gaps" should appear to the left of the smaller ones
- you won't need to deal with punctuation or anything but a-z characters (upper and lower case)
- the last line in a fully justified text paragraph is simply left-aligned

Your program will read the input text until a terminating # appears. You will then output that text, full-justified to a 30-character column. All spaces will be replaced with the period (".") character. The # is ignored in the output.

Type alignment examples

Flush left / ragged right:

```
This has been a test of
the Emergency Broadcast
System. In the event of
an actual emergency,
panic would ensue.
```

Flush right / ragged left:

```
  This has been a test of
  the Emergency Broadcast
  System. In the event of
     an actual emergency,
       panic would ensue.
```

Full justification:

```
This has been a test of
the Emergency Broadcast
System. In the event of
an actual emergency,
panic would ensue.
```

Sample Input

```
If Java had true garbage collection most programs would
delete themselves upon execution#
```

Sample Output

```
If...Java...had..true..garbage
collection.most.programs.would
delete.....themselves.....upon
execution
```

problem 4
Reverse + Add = Palindrome
5 points

JAVA program name must be prob04.java
C /C++ program name must be: prob04.exe

Introduction

A simple way to generate numeric palindromes is the "reverse and add" method. Start with a positive integer, reverse the digits, and add it to the original number. If the result is not a palindrome (a sequence that is identical right-to-left and left-to-right), then repeat the process. For example, starting with 195 we add 591 (the reverse of the digits) to get 786, which is not a palindrome. Then we add 786 and 687 to get 1473, which is also not a palindrome. Then 1473 plus 3741 yields 5214, and 5214 plus 4125 yields 9339. The number 9339 is a palindrome, so we're done!

As a matter of fact, there are some rare numbers for which this process never yields a palindrome. For the purposes of this problem, you can assume that all input will yield a palindrome.

Sample Input

The input will be a single positive integer. Several examples are provided here.

```
195

304

628

570

259
```

Sample Output

The program must print the palindrome that results from applying the reverse and add method.

```
9339

707

5995

5115

2332
```

problem 5
Making Waves
6 points

JAVA program name must be prob05.java
C /C++ program name must be: prob05.exe

Introduction

Your task is to generate a "wave" from a string of characters and numbers. The wave is generated by elevating (line-1) a higher character, and degrading (line+1) a lower character. Equal characters are kept on the same line (no elevating or degrading done).

Input is made up of lower case characters and numbers only. Letters are considered higher than numbers.

Sample Input

```
1234567890qwertyuiopasdfghjklzxcvbnm
```

Sample Output

```
                          z
                         l x v n
                        k   c b m
                       j
                      h
                     g
                y   p s f
               t u o a d
             w r   i
          9 q e
        8 0
       7
      6
     5
    4
   3
  2
1
```

Sample Input

```
3141592653589793238462643383279502884197169399375105820974944592307816406286
```

Sample Output

```
              9 9   8 6 6
      9 6   8 7 3 3 4 2 4  8   9   88
 3 4 5 2 5 5     2       33 3 7 5 2  4 9   9 99 7
  1 1     3                 2   0    1 7 6 3  3 5   8              8 6
                                      1        1 5 2 9      9 3 7 1 4 6 8
                                               0   0 7 9  5 2 0     0 2 6
                                                    4 44              2
```

problem 6
Laser Target
7 points

JAVA program name must be prob06.java
C /C++ program name must be: prob06.exe

Introduction

Just how good a shot are you? The objective of this puzzle is to determine if a laser, shot from within a maze containing mirrors, hits its intended target. In this puzzle, a laser shot travels from its origin to the direction it is pointing. If a laser hits the wall, it stops. If a laser ray hits a mirror, it bounces 90 degrees to the direction the mirror points to. Mirrors are two sided (both sides are reflective). If a laser ray hits the laser emitter itself (><^v), it is treated as a wall.

Input

Two integers that determine the size of the maze. (i.e. `10 10` for a 10 by 10 matrix.)
Character symbols:

`#` - A solid wall
`X` - The target the laser has to hit
`/` or `\` - Mirrors pointing to a direction (depends on laser direction)
`V, ^, >, <` - The laser emitter (Determines the direction (down, up, right, left respectively)

There is only one laser and only one target. Walls inside rooms are possible.

Output

`True` if the laser will hit the target
`False` if the laser will miss the target

Sample Inputs/Outputs

Input	Input

`10 6`
`##########`
`#  / \  #`
`#  #`
`#  \  x#`
`# >  /  #`
`##########`
	`10 6`
	`##########`
	`#  v x  #`
	`# /  #`
	`#  /#`
	`#  \  #`
	`##########`

Output: `True`	Output: `False`

HINT: For debugging, it might be useful to print the path of the laser beam in the maze as a part of your output. The judges will not evaluate this path information.

problem 7
QWERTY Sort
7 points

JAVA program name must be prob07.java
C /C++ program name must be: prob07.exe

Introduction

You have been hired by a university researcher to write custom programs that assist in university research. From the first day you knew it would be a strange job. You have been told to write a program that can sort words in QWERTY order – that is, the order of the letter layout on a standard QWERTY keyboard. The full sequence is:

```
QWERTYUIOPASDFGHJKLZXCVBNM
```

Hey, the work may be bizarre but the pay is good.

Input

The input consists of a series of words, one per line. The last line contains only a period.

```
ARREST
SUBDIVISION
DISCONTENT
SUPERIOR
TOPOLOGY
DEBUNK
APPENDIX
SUBDUE
TRUNK
.
```

Output

The program must print the words in QWERTY sorted order, one per line.

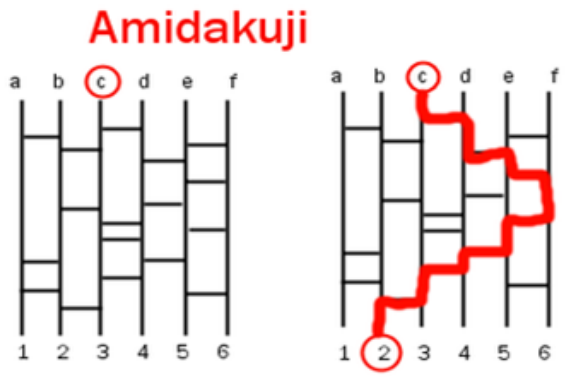
```
TRUNK
TOPOLOGY
ARREST
APPENDIX
SUPERIOR
SUBDUE
SUBDIVISION
DEBUNK
DISCONTENT
```

problem 8
Amidakuji
8 points

JAVA program name must be prob08.java
C /C++ program name must be: prob08.exe

Introduction

Amidakuji, also known as Ghost Leg, is a method of lottery designed to create random pairings between two sets of any number of things, as long as the number of elements in each set is the same.
It consists of vertical lines with horizontal lines connecting two adjacent vertical lines scattered randomly along their length; the horizontal lines are called "legs". The number of vertical lines equals the number of people playing, and at the bottom of each line there is an item - a thing that will be paired with a player. The general rule for playing this game is: choose a line on the top, and follow this line downwards. When a horizontal line is encountered, follow it to get to another vertical line and continue downwards. Repeat this procedure until reaching the end of the vertical line.



Note: The input will consist of '-' (minus signs) and '=' (equal signs). The minus sign is considered to be lower than the top bar of the equal sign, but higher than the bottom bar.

Input

The first line of the input file will specify the number of players (vertical lines) and number of rows of input.

```
6 4
|-|  |=|-|=|
|  |-|  |  |-|
|=|  |-|  |  |
|  |  |-|=|-|
```

Output

Output the letter number pairs for each line. The first line is lettered 'a' at the top and '1' at the bottom.

```
a : 6
b : 1
c : 3
d : 5
e : 4
f : 2
```

problem 9
Neotron
8 points

JAVA program name must be prob09.java
C /C++ program name must be: prob09.exe

Introduction

Help Neotron flight the lag and escape the grid-matrix. Neotron has learned about an unpublished grid-matrix API call that can attach tracers to the arena agents and track their location. Neotron is busy hacking the grid-matrix security to gain root access to the API, so he needs someone else to write the part of the program that retrieves the arena agent handles stored in the core memory disks of the arena agents' phones. The clue architect designed the core memory disks with a security protocol that uses an encryption key consisting of three English words. Of course the weakness of a word-based encryption key is its susceptibility to brute-force attack, so the clue architect implemented a real-time alert triggered by the use of an invalid encryption key. Early attempts by Neotron's allies to break into the core memory disks ended in disaster because the real-time alerts betrayed their activity and location before a valid key could be discovered. Neotron's success depends on your program producing a valid encryption key without resorting to brute-force attacks on the core memory disks themselves. Fortunately for you, Neotron's friend Corin infiltrated a grid-matrix control substation and used a thermal imager to capture infrared energy emissions of a keyboard before and after the entry of an encryption key. She used the thermal data to produce a list of letters used in the key and sent them to you in an encrypted email.
Write a program that can form three words using a list of letters and a dictionary of valid words.

Input

The first line of input is a sequence of letters with no spaces. These are the letters of the encryption key. The next several lines are dictionary words, one per line. The input ends with a period.

```
EERRTTTUIIIAAASSDFGGMMN
APPENDIX
DISCONTENT
FINGER
ENCRYPT
HELICOPTER
LOCATION
MAGISTRATE
NEOLOGISM
STADIUM
SUBVERT
TOPOLOGY
.
```

Output

The program must print the three words of the encryption key in alphabetic order.

```
FINGER MAGISTRATE STADIUM
```

problem 10
Egyptian Fractions*
9 points

JAVA program name must be prob10.java
C /C++ program name must be: prob10.exe

Introduction

The ancient Egyptians wrote fractions differently than we do today. They had hieroglyphic notations for reciprocals (or unit fractions) of the form 1/n, such as ½, ⅓, ¼, etc., but they had no way of writing fractions like 2/5, 3/4, or 5/8. Instead they wrote these fractions as the sum of distinct unit fractions.

For example, 2/5 could be written as 1/3 + 1/15, but not 1/5 + 1/5 because the fractions are not distinct.

Write a program that can convert Egyptian fractions into the standard fractional notation (numerator over denominator) that we use today.

Input

The input will consist of lines of integer numbers. The numbers on each line are the denominators of the unit fractions that need to be converted. Each line ends with the number zero, indicating the end of a single set of unit fractions. There will be no more than ten unit fractions per line and none of the numbers will exceed 100. The last line of input contains only the number zero.

```
3 15 0
2 6 0
3 5 15 0
2 4 20 0
2 8 0
2 8 88 0
35 0
12 51 68 0
63 99 0
2 6 21 77 0
6 9 18 16 24 48 96 0
0
```

Output

The program must add the unit fractions and simplify the sum into its most reduced form. The program must print each output on a separate line.

```
2/5
2/3
3/5
4/5
5/8
7/11
1/35
2/17
2/77
8/11
15/32
```

**and don't even get us started about Babylonian fractions…*

problem 11
Fish Tales
9 points

JAVA program name must be prob11.java
C /C++ program name must be: prob11.exe

Introduction

Several friends take a fishing trip every year. By tradition they have a contest to see who catches the heaviest fish. Each person pays for the junk food eaten on the trip based on their final standings, where the winner pays nothing and the loser pays the most. Write a program to determine the standings by using clues. By the way, in the tradition of fishing trips, every statement quoted here is a falsehood.

Marta: "Larry was first."
Woody: "I beat Sally."
Sally: "Marta beat Woody."
Larry: "Woody was second."

The answer for this example is Sally first, Larry second, Woody third, and Marta last.

Sample Input

The input consists of four lines of text. Each line begins with the name of one the friends followed by a colon and a statement. A statement can take one of two forms: "name1 BEAT name2" or "name1 WAS position." Notice that "name 1" could be an actual name or the word "I" and that "position" could be any of "FIRST", "SECOND", "THIRD", or "LAST". All four statements are false.

Example 1:
```
ASOK: CAROLINE WAS THIRD
LACHELL: SAM BEAT CAROLINE
SAM:  ASOK WAS LAST
CAROLINE: I BEAT LACHELL
```

Example 2:
```
NAVYA: I BEAT JING
JING: JAVIER BEAT NAVYA
JAVIER: NAVYA WAS SECOND
BRUCE: I BEAT JING
```

Sample Output

The program must print the four names on one line in order of their final standings. There will only be one possible solution.

Example 1: `LACHELL CAROLINE ASOK SAM`
Example 2: `JING BRUCE NAVYA JAVIER`

problem 12
Diffraction Grating
10 points

JAVA program name must be prob12.java
C /C++ program name must be: prob12.exe

Introduction

A diffraction grating is an optical component consisting of parallel elements at regular intervals. The grating is commonly made as a reflective material with grooves etched onto its surface (see diagram). It is the tool of choice for spectral analysis (separating the colors of incident light) because its high-resolution spectra show narrow bands of emission and absorption lines.

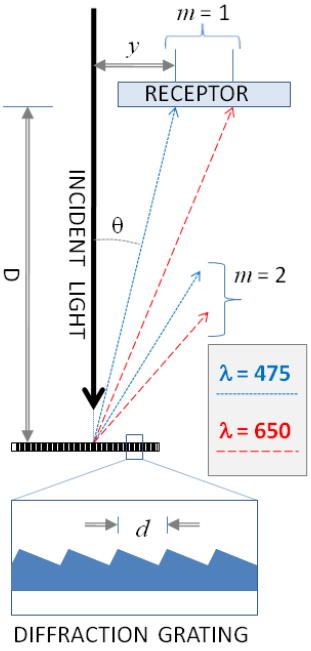
Interestingly, you can make a diffraction grating spectroscope with a CD or DVD, a small shipping box, a toilet paper tube, two razor blades, and some duct tape. It's good, clean, wholesome fun.

To make scientific measurements with your diffraction grating, you will need to be able to calculate the *y*-offset of the center-line for a particular wavelength of light λ given the slit separation *d* of the grating, the distance *D* of the receptor (like a camera or your eye) from the grating, and the spectral order *m*. The *y*-offset can be calculated using the following formulas:

$$d \sin\theta = m \lambda \qquad\qquad y = \frac{m \lambda D}{d \cos\theta}$$

For this program we will only consider the first spectral order, *m*=1, because it is the brightest. The value θ is the angle of diffraction based on the wavelength, spectral order, and slit separation.

The application of the formula seems simple, but there is a catch. Diffraction gratings are commonly referenced by grating density (the number of grooves per mm). However, the slit separation *d* is the "peak-to-peak" interval length of the grating and must be given to the formula in nanometers (nm). Recall that 1 mm = 1,000,000 nm.



Sample Input

The input will consist of three numbers: a wavelength λ of light in nm, a receptor distance D in cm, and the grating density in lines/mm (which must be converted to d). Two examples are shown below, but your program will read only one set of numbers on each run.

```
450 10 625

656 12.7 1350
```

Sample Output

The program must print the y-offset (which will be in cm) of the given frequency. The answer must match the judge data within +/- 1 digit of the fourth significant digit, i.e., if the correct answer is 4.069 then anything between 4.068 and 4.070 will be counted as correct.

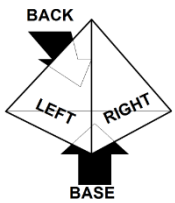
```
2.931

24.216
```

problem 13
Tetrahedron Face Painting
10 points

JAVA program name must be prob13.java
C /C++ program name must be: prob13.exe

Introduction

Humans can recognize other people's faces over a wide range of angles. We'll explore this idea by writing a program to recognize a color pattern on a tetrahedron from different angles. A tetrahedron is a three-dimensional solid with four faces where each face is a triangle. It looks similar to a pyramid, but a tetrahedron has a base and only three sides (see diagram). The tetrahedron can be rotated such that three of the faces shift positions and the fourth face remains in place. For example, you can rotate the tetrahedron on its left face so that the right side moves to the back, the back side to the base, and the base side to the right. Other rotations are also possible.



Let's paint each face of the tetrahedron a different color: red, green, blue, yellow, orange, purple, black, or white. If we represent these colors with the letters R, G, B, Y, O, P, K, and W, then we can describe any color combination as a sequence of four letters. We'll write the face colors in this order: left, right, back, and then base. Some color combinations could be considered identical to each other if they appeared the same after the tetrahedron was rotated in one or more directions. Thus the sequences RYKB and YRBK could be considered identical because one can be transformed into the other by rotation. On the other hand, the sequences GOWP and PWGO cannot be considered identical because there is no set of rotations that can transform one into the other.

Input

Each line of input contains two color sequences of four letters each. The last line of the input contains the two words QUIT EXIT.

```
RYKB YRBK
OBKW BKOW
BPWG GBWP
ORGK KRGO
YOPW POWY
GYPR GPYR
RBGY OPKW
QUIT EXIT
```

Sample Output

For each line of input the program must print a single line of output. It must repeat the two color sequences, followed by either the word SAME if the two sequences can be considered identical or the word NOT if they cannot be considered identical. The program must not print any output for the input line QUIT EXIT.

```
RYKB YRBK SAME
OBKW BKOW SAME
BPWG GBWP SAME
ORGK KRGO NOT
YOPW POWY SAME
GYPR GPYR NOT
RBGY OPKW NOT
```

problem 14
Dee Scramble
~~10~~ 11 points

JAVA program name must be prob14.java
C /C++ program name must be: prob14.exe

Introduction

Dee Scramble, singer and songwriter for the band Tilted Sister, is convinced that someone is secretly intercepting her emails and leaking information about her private life to the press. She has therefore written a program to scramble her emails. First, all spaces are converted to tilde "~" characters, which are otherwise unused. Then first character of the email remains in its position and the program shifts each subsequent character *M* positions to the right after the previous character. If a character is shifted beyond the length *N* of the email, then the position wraps back around the beginning of the email and continues counting the remaining shift positions. Dee randomly assigns the shift length *M* so that *M* > 2 and every character in the original email shifts to a unique position in the scrambled email.

For example, suppose Dee wrote the following email:

```
I'M HAVING DINNER TONIGHT WITH STEVE AT CHEZ BREU-VIE. -DEE
```

The scramble program would produce the following output for M = 3:

```
INC'IHMGE~HZHT~A~BVWRIIENTUGH-~~VDSIITENE.NV~EE-R~D~AETTEO~
```

If instead the scramble program selected M = 7 then the scrambled email would be:

```
I~VIGWZ'TEE~I~MO~.DTB~NA~IHRHIT-N~EAG~DNSUVHCEET-ITHEREVN~E
```

One weakness in Dee's scramble is that her email program automatically signs her emails with the string "-DEE" at the end. Using this knowledge someone could write an unscrambler. This weakness may turn out to be a good thing for Dee, who has unfortunately deleted the unscrambler program by accident and can no longer read her email. Write an unscrambler program for Dee that can unscramble her emails. But don't sell it to the gossip newspapers. After all, just because you're paranoid doesn't mean no one is watching you.

Sample Input

The first line of the input file will specify the number of characters N in the scrambled email. The second line will contain the scrambled email message. The value of M is not provided.

```
59
INC'IHMGE~HZHT~A~BVWRIIENTUGH-~~VDSIITENE.NV~EE-R~D~AETTEO~
```

Sample Output

The program must print the unscrambled email.

```
I'M HAVING DINNER TONIGHT WITH STEVE AT CHEZ BREU-VIE. -DEE
```

problem 15
Where's My WYSIWYG?
11 points

JAVA program name must be prob11.java
C /C++ program name must be: prob11.exe

Introduction

Back before WYSIWYG editors there were only crude line editors.
Your task is to create an editor program that modifies text using three simple commands, and outputs the final result.

This editor only supports a few commands:

A <edit line number> <text>	Add a line at/after <line number> with <text>
D <edit line number>	Delete line <line number> If no line number is given, delete the first line in the file.
<edit line number> S /String1/String2/	Change String1 to String2 at line number. If no line number is given, change all occurrences of String1 to String2.

Input

You are given a text file. The input file will consist of the number of text lines to edit, the text to be edited, the number of edit commands, and a set of edit commands which you will use to transform the text.

```
5
My name is
Puddintane
John Smith.
Nice to
Meet you.
6
s/My name is/Are you/
s/Nice/It's Nice/
d 2
2 s/Smith./Smith?/
a 2 My name is Jane Smith.
a 3 Oops! It is Jane Brown.
```

Output

```
Are you
John Smith?
My name is Jane Smith.
Oops! It is Jane Brown.
It's nice to
meet you.
```

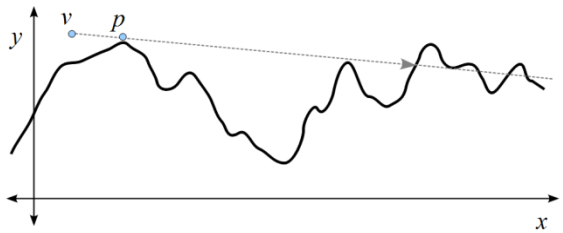
problem 16
Rendering Fractal Geometry
13 points

JAVA program name must be prob16.java
C /C++ program name must be: prob16.exe

Introduction

Fractal functions are used in computer graphics to generate synthetic terrain with a continuous, adaptive level of detail and yet virtually no data storage overhead. One challenge of displaying a fractal terrain is that it is not trivial to locate the intersection of the fractal with a ray-trace vector. A ray-trace vector describes a "line of sight" from a viewpoint v, which represents a person's eye, through a point p, which represents a pixel on a screen. The vector points out into the virtual world until it either intersects the terrain or passes a predefined clipping distance.

Fractal functions are quite interesting, but their workings can be difficult to describe. So instead of a three-dimensional fractal terrain we will use a simple two-dimensional terrain described by the trigonometric function $f(x) = a_1 \cos(b_1 x) + a_2 \sin(b_2 x)$. The algorithm used to render the trigonometric terrain will work just as well with a fractal terrain.



Write a program that can find the first intersection of a ray-trace vector with a terrain function. One method for doing this was developed by fractal pioneer F. Kenton Musgrave. His approach was to start at point p and search for intersections by incrementing x along the vector in very small steps. For this program's terrain function the step value Δx should be calculated by dividing π by ten times the greater of b_1 or b_2.

The program can determine if it has located an intersection by following this logic:

1. Calculate $s(x_i) = f(x_i) - g(x_i)$, where $g(x)$ is the equation of the line defined by v and p.
2. Compare $s(x_i)$ with $s(x_{i-1})$. If the signs (+/-) of the two values do not match, then the two functions intersect somewhere between x_i and x_{i-1}. The program must use a binary search or some other method to refine the exact x-value where $f(x)$ and $g(x)$ intersect.
3. Let $x_{i+1} = x_i + \Delta x$, and repeat until an intersection is found or the clipping distance is exceeded.

Input

The first line of input contains the four values a_1, b_1, a_2, and b_2, in that order. The next line contains the values x_v, y_v, x_p, and y_p, in that order. You may assume $x_p > x_v$. The last line indicates the predefined clipping distance, which is measured on the x-axis from x_v.

```
3.22 1.41 1.07 5.39
0.24 4.111 0.29 4.104
9
```

Output

The program must print the value of x where the ray-trace vector intersects the terrain function. This number must be accurate to three digits after the decimal point. If the ray-trace vector does not intersect the terrain function within the clipping distance, then the program must print the word NONE.

```
4.76442
```

problem 17
Node Graph Maze
15 points

JAVA program name must be prob17.java
C /C++ program name must be: prob17.exe

Introduction

In mathematics, a graph represents a set of objects where some pairs of objects are connected by links. Graphs have been adopted in computer science, where objects are often drawn as circles called nodes and the links are drawn as lines connecting the nodes. Graphs are useful for modeling many real-world problems such as cities connected by roads, finite state machines, or friendship networks among people. One reason graphs are useful is they make it easy to solve connectivity problems, which means finding if or how two nodes are related, even if they do not share a link. The problem is essentially the same as solving a maze where each link is a hallway and each node is an intersection.
Write a program that can solve a node graph maze.

Sample Input

The input is a series of node pairs, one pair per line, where each pair indicates a link between two nodes. All node names will consist of a single upper-case letter followed by a single digit. The last two lines of the input indicate the IN node and the EX node. There may be dozens of nodes in the input.

```
R2 D2
C3 P0
K9 X3
A3 B9
F1 E6
R2 I9
C1 X5
B9 Z2
T2 Z2
L1 B9
B7 W0
G8 A3
I9 T2
X3 T2
B4 X3
N1 L1
T2 E6
B7 Z2
Q5 B7
L1 X5
R9 E6
I9 C3
IN X5
EX I9
```

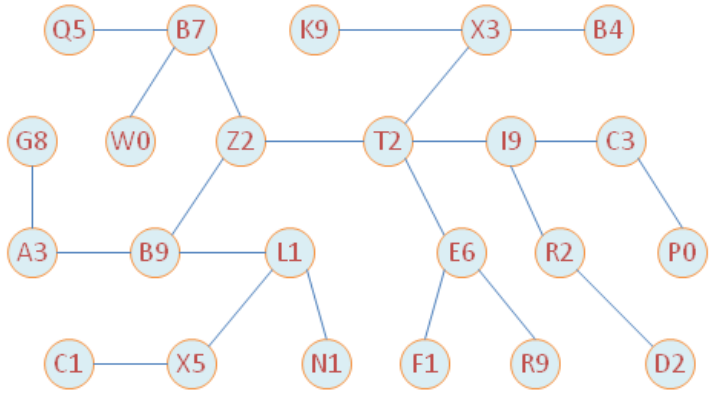


Figure 1. A node graph corresponding to the sample input

Sample Output

The program must find the path from the IN node to the EX node and print the nodes visited, in order. The input will not contain any cycles (or loops), meaning that there will only be one valid path connecting any two nodes.

```
X5 L1 B9 Z2 T2 I9
```

(*continued on page 2*)

```
S4 E9
T6 L1
N5 H9
L1 J9
F0 B4
S4 J9
L1 N5
B9 S9
W3 X5
E9 F1
F0 O9
K1 B2
B2 K8
L1 Y0
Y0 W3
P5 J9
U6 J9
W3 M4
B9 B4
B4 L0
B2 B4
K3 X5
B4 S4
I7 Y0
Y0 R5
S4 D3
K3 J4
IN D3
EX W3
```

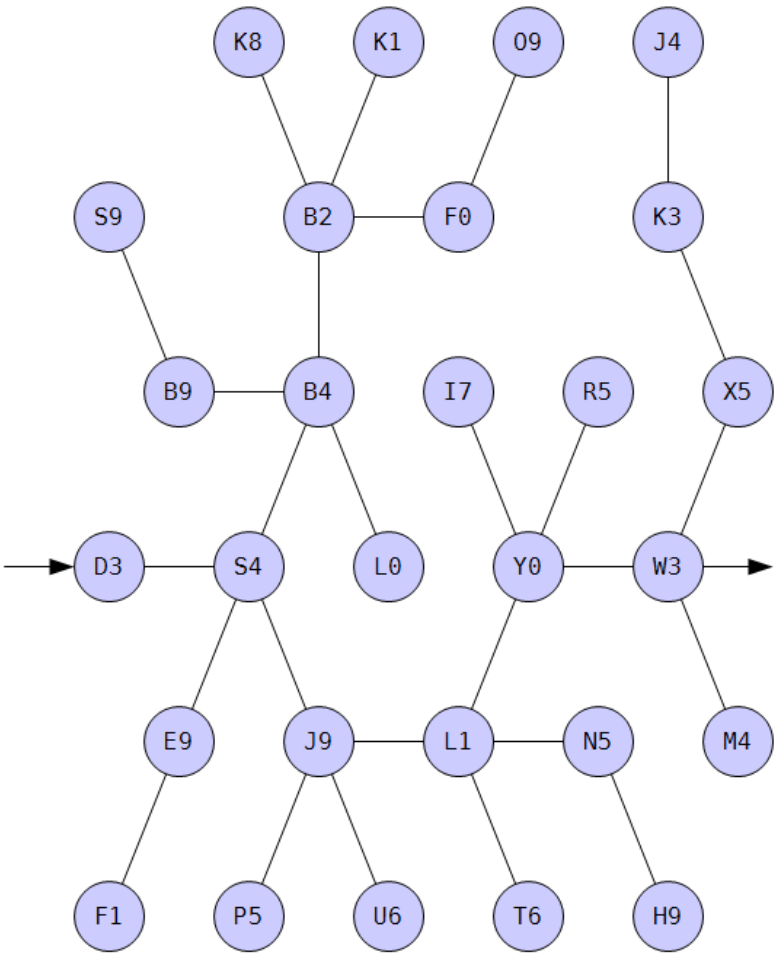


Figure 2. A node graph corresponding to the second sample input data set.

problem 18
Power Gaming Calculator
20 points

JAVA program name must be prob18.java
C /C++ program name must be: prob18.exe

Introduction

A Power-Gamer plays a role-playing game with the goal of maximizing the power of the player's character, often as quickly as possible. Power leveling requires knowing what quests award the most experience points per unit of time. This analysis is not trivial because quest experience point awards depend on a variety of factors.

For this program a character begins the game at level one with zero experience points. The character level automatically increases by one for every thousand experience points earned. Each quest has a difficulty level and base experience award value. Characters may not complete quests above their level. The experience value of a quest is reduced if the character level exceeds the quest level. This reduction is 10% times the level difference. The experience value is also reduced by 10% each time the quest is repeated. Experience point reductions are cumulative but may not exceed 100%. Fractional reduction values should be truncated. For example, a 40% reduction of a quest valued at 347 experience points results in a quest value of 209 points, a reduction of 138. Write a program that can determine the optimal set of quests to complete for a character to reach a level goal in the shortest amount of play time.

Input

The input will consist of a series of quests, one per line. The first line of the input will indicate the number of quests. Each quest has a name, a level, an experience point value, and the average play time in minutes. Two example data sets are shown below, but the program will only read a single data set each time it is run.

```
12
TrollTrouble 1 285 19.4
MistyMountains 1 375 23.1
Mirkwood 1 347 24.8
OldForest 2 338 23.9
Weathertop 2 351 18.3
MinesOfMoria 2 418 36.2
EntMoot 3 214 22.6
HelmsDeep 3 429 29.7
DeadMarshes 3 248 23.5
ShelobsLair 4 292 19.9
GatesOfMordor 4 412 22.4
MountDoom 4 301 16.8
```

```
14
Wardrobe 1 304 21.4
TurkishDelight 1 321 22.3
WhiteCastle 1 377 24.0
RescueTrumpkin 2 294 18.1
CairParavel 2 336 23.8
LoneIslands 2 475 26.6
DragonCave 3 296 25.8
Harfang 3 260 25.1
Underland 3 272 26.5
EscapeToTashbaan 3 284 27.4
DefenseOfArchenland 4 348 24.7
BetweenTheWorlds 4 226 16.2
TalkingTrees 4 235 18.4
Oathbreakers 4 296 20.3
```

Output

The program must determine an optimized sequence of quests to let a character achieve level five in the shortest amount of play time. The output must show the total play time and experience points acquired, followed by the quests completed, in order, one per line. The exact sequence may vary slightly without affecting the result. The number of minutes of play time must be equal to or less than the judge's solution. The program must print a solution in less than a minute. Can your program find better solutions than the examples shown below?

```
255.9 4001
MistyMountains
TrollTrouble
Mirkwood
Weathertop
Weathertop
Weathertop
Weathertop
HelmsDeep
HelmsDeep
GatesOfMordor
MountDoom
MountDoom
```

```
272.8 4000
WhiteCastle
TurkishDelight
Wardrobe
LoneIslands
LoneIslands
LoneIslands
RescueTrumpkin
RescueTrumpkin
CairParavel
DefenseOfArchenland
Oathbreakers
Oathbreakers
```