

problem 7

Minelayer

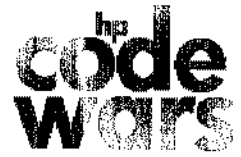
6 points

Introduction

One of the most frequently used office anti-productivity applications (you and I would call them “games”) is an application called Minesweeper. You’re helping out in the creation of a free-ware version of the game, and are tasked with setting up the playfield (a.k.a. minefield).

The playfield setup begins with mines randomly placed in a grid. Each square not containing a mine must be populated with a digit (1-9) representing the number of mines adjacent to it. Squares with no adjacent mines are filled with a period.

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



Sample Input

The input consists of a grid (30 columns by 15 rows of text), each square populated with either a mine “*” (asterisk) or a blank “.” (period).

```
.....*.....**
...**..*..*.....
.....**.....*...
.....**..*.....
.....*.....*...*
.....*.....*...
.....*.....*...
.....**.....*..
*.****.....*...*
.....*.*.*.....
.....*.....*...
.....*..**.....*..**
.....*.....***.*...
...*.***.....*...
.....**.....***.....*
```

Sample Output

The program must output a playfield, ready for a game of Minesweeper!

```
..1221111.111.....1*1....1**1
..1**11*1.1*1..1221111..112221
..12211122321112**1.....1*1...
.....1**212*2221111..11211.
.....1111222*211...1*1....2*2.
.....1*1...111.....1221...3*3.
.....111...1221.....1*1...3*3.
1223321....1**1.....1221..2*31
*2****1....12221212222*1..12*1
1223321.....1*2*2**211..1221
.....1111221.112122321.123*1.
.....1*11**21.....2*423**21.
..1121211234*1.....3***6*41..
..1*2*1..1*432.....3*8***2.11
..11211..12**1.....2***421.1*
```

Introduction

A traditional English riddle goes something like this:

As I was going to St. Ives
 I met a man with seven wives
 Each wife had seven sacks
 Each sack had seven cats
 Each cat had seven kits
 Kits, cats, sacks, wives
 How many were going to St. Ives?

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



The answer, of course, is one. Solving a riddle like this requires AI techniques beyond the scope of this contest. Instead, let's leave the riddle solving to the humans (for now) and write a program that can read text formatted like the St. Ives riddle and do some basic math with it.

Sample Input

Program input is five lines with the following format:

```
A MAN WITH NUMBER COMPANIONS
EACH COMPANION HAD NUMBER CONTAINERS
EACH CONTAINER HAD NUMBER OBJECTS
EACH OBJECT HAD NUMBER ITEMS
HOW MANY SOMETHINGS?
```

Valid numbers are two through thirteen, inclusive. The numbers may or may not be equal. Singular and plural will always be different by the final -S only, even if it contradicts English spelling rules (e.g., WIFES instead of WIVES). The "something" may be any companion, container, object, or item. For example:

```
A MAN HAD FIVE EMPLOYEES
EACH EMPLOYEE HAD NINE PROJECTS
EACH PROJECT HAD SEVEN DEADLINES
EACH DEADLINE HAD THIRTEEN IMPEDIMENTS
HOW MANY DEADLINES?
```

Sample Output

The program must print the number of "somethings" followed by the "something" noun in plural form.

315 DEADLINES

problem 9

Letter Scramble

8 points

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



Introduction

In a popular board game, players lay letter tiles on a grid to form words. Points are awarded for each letter in the word, where uncommonly used letters have a higher point value than commonly used letters. Additionally, certain squares in the grid multiply the value of letters or entire words played on them. There are four modifiers: double letter score, triple letter score, double word score, and triple word score. If a word is laid across letter and word modifiers, then the letter modifiers are applied first, then the word modifiers.

Write a program to calculate point values for words placed on the game grid.

Sample Input

The input will be divided into two sections. The first section is a grid where a square is represented by two characters separated by spaces. The size of the grid is given by a single integer preceding the grid itself. Normal squares are represented with [] characters. Modifier squares begin with a 2 or 3, followed by a W for word or L for letter. The next section provides a list of words to be placed onto the grid, starting with the number of words in the section. Each word is followed by the row and column of the first letter, then by a letter H or V to indicate if the word should be placed horizontally or vertically.

```
8
[] [] [] [] [] [] [] []
[] 3W [] 2L [] [] [] []
[] [] [] [] 3L [] 2W []
[] [] [] [] [] 2L [] []
[] [] 2W [] [] [] [] []
[] [] [] 3L [] 3W [] []
[] 2L [] [] [] [] [] []
[] [] [] [] [] [] [] []
3
CRAFT 2 2 H
ZOO 6 4 V
QUARK 4 6 V
```

A - 1	N - 1
B - 3	O - 1
C - 3	P - 3
D - 2	Q - 10
E - 1	R - 1
F - 4	S - 1
G - 2	T - 1
H - 4	U - 1
I - 1	V - 4
J - 8	W - 4
K - 5	X - 8
L - 1	Y - 4
M - 3	Z - 10

Point Values

Sample Output

The program must print each word followed by the point value for that word. The point value for a word must be calculated using the rules given above. The program should treat each word individually and not be bothered if the words happen to overlap, whether correctly or incorrectly.

```
CRAFT 33
ZOO 32
QUARK 84
```



Problem 10

Data in a Huff 6 points

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

Task Description

Compression algorithms are categorized according to whether they allow the exact original data to be reconstructed from the compressed data. Formats like MP3 and JPEG are "lossy", while formats like ZIP and PNG are "lossless". Not only can compressed data files be transmitted (across the internet or onto your MP3 player) faster than uncompressed data files, but they also require less storage space.

Huffman coding is a lossless compression algorithm that encodes each input character into a variable-length bit string. Write a program to decode a bit string using the predefined Huffman encoding shown in Table 1.

Sample Input

The input will be a simple string of zeros and ones terminated by a single period.

```
10110100111010111100
11011101000111001110
10011100101111101001
1111110101011101000.
```

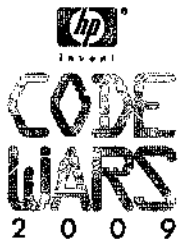
Sample Output

The program must print the decoded message to the console. In addition, the program must print the number of bits in the message and the number of bits in the compressed bit string. For this program we define each output character as an eight-bit byte. The numbers must be labeled correctly.

```
CODE WARS RULES!
message bits: 128
compressed bits: 79
```

' '	00
'.'	010010
'!'	01011101000
','	01011110
'2'	0101111100100
'0'	010111011
'1'	110111100
'2'	010111000
'3'	11011110111
'4'	010111110011
'5'	0101110101
'6'	11011110100
'7'	01011101001
'8'	1101110101
'9'	0101111100101
'A'	1000
'B'	110110
'C'	10110
'D'	11010
'E'	1111
'F'	01010
'G'	010110
'H'	10101
'I'	0110
'J'	01011111000
'K'	010111111
'L'	10100
'M'	01000
'N'	0111
'O'	1001
'P'	010011
'Q'	010111001
'R'	11100
'S'	11101
'T'	1100
'U'	101111
'V'	11011111
'W'	110110
'X'	0101111100
'Y'	101110
'Z'	11011110110

Table 1



Problem 11

Mass Intelligence

6 points

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

Task Description

As computers continue to increase in speed and memory capacity, new software will allow us humans to use our machines to solve problems in ways never before possible. The key to making this new software work will be its ability to mimic human intelligence. One important aspect of human intelligence is the manipulation of symbolic representations of knowledge for a field of study.

For example, in chemistry each element is represented by a one or two letter symbol: H for Hydrogen, C for Carbon, Na for Sodium. When several elements have been combined to form a compound, it is represented by a formula in which each element appears. If the compound is made of more than one of a certain element, then that element's symbol is followed by an integer that indicates how many of that element are in a single molecule of the compound.

Furthermore, each element has an atomic mass that indicates the relative mass of one atom of that element. When elements are combined to form a compound, its molecular mass is the sum of the atomic masses of all its constituent elements.

Write a program that can read a chemical formula and determine the molecular mass of the compound.

Sample Input

Each line of the input file is a chemical formula. The input ends with the word END.

O2
H2O
CH4
Na2SO4
C6H12O6
BeSO3
NaHCO3
END

**Table of Atomic
Masses**

H	1.008
He	4.003
Li	6.941
Be	9.012
B	10.81
C	12.01
N	14.01
O	16.00
F	19.00
Ne	20.18
Na	22.99
Mg	24.31
Al	26.98
Si	28.09
P	30.97
S	32.07
Cl	35.45
Ar	39.95

Sample Output

Your program must print the chemical formula, followed by the molecular mass of the compound described by the formula.

O2 32
H2O 18.016
CH4 16.042
Na2SO4 142.05
C6H12O6 180.156
BeSO3 89.082
NaHCO3 84.008



Problem 12

Histogram

6 points

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

Task Description

Write a program to count the number of times words appear in the input and print a histogram of the top five most frequent words in order of frequency.

Sample Input

The input will contain several lines of text. The input is terminated by a line with the string "###".

```
When in the Course of human events, it becomes necessary for one
people to dissolve the political bands which have connected them
with another, and to assume among the powers of the earth, the
separate and equal station to which the Laws of Nature and of
Nature's God entitle them, a decent respect to the opinions of
mankind requires that they should declare the causes which impel
them to the separation.
```

###

Sample Output

The program must display a histogram of the top five most frequent words in the input text. Lower-case letters should be converted to upper-case. Punctuation marks should be ignored. Each line of output must contain, in order, the histogram, the rank of a word, the word itself, and the number of occurrences. Words with the same frequency must be printed in alphabetical order. Please follow the format of the sample output below.

```
***** #1: THE - 9
***** #2: OF - 5
***** #3: TO - 5
*** #4: AND - 3
*** #5: THEM - 3
```



Problem 14

Anagram Finder

8 points

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

Task Description

An anagram is simply the rearrangement of the letters of a word or words to make another word or other words.

Write a program to look for anagram pairs from a list of supplied words. All anagrams found will come from the supplied list of words.

Input

A list of supplied words. Some of the words will have matching anagrams in the list; some will not. There will be no more than two corresponding anagrams per word list.

Output

A list of anagram word pairs.

Sample Input/Output

```
Type test_words.txt
"DANGER", "GARDEN", "WHO", "HOW", "DO", "ODD"
```

```
Java Anagrams < test_words.txt
WHO:HOW
DANGER:GARDEN
```

Note

Do not output word pairs more than once. For example, in the above example, "GARDEN : DANGER" would be a repeat.



Problem 15

Robert the Constructor

11 points

JAVA: program name must be prob15.java
C/C++ program name must be: prob15.exe

Task Description

Robert is a certified WCE (Word Construction Engineer) who builds words by deconstructing other words and recycling the parts. Robert receives orders for new words and BTO's them (Build To Order) using words from the community word recycling center. He breaks the words into substrings and combines these substrings to form new words. He only has room in his workshop to dismantle two words at a time

and he can only use one substring from each source word. These limitations mean that Robert spends a lot of time rummaging through recycled words to locate the proper substrings.

Write a program to help Robert find two words from a list so that he can build a new word.

Sample Input

Each line of the input contains an integer N followed by N words. The first word is the BTO word that Robert needs to build. The rest are the words in the box from the community recycling center. The input ends with a single zero.

```
7 PUTTER WRANGLER TERMINAL PURE INPUT FOREIGN PULMINARY
6 FETCH REFINE WATCHER FERN CHURN WHICH
9 MANUAL MENIAL ALTER MARE SALUTE HUMANS MARTIAL METAL SPIRITUALITY
0
```

Sample Output

For each BTO word requested, the program must print the BTO word followed by two words from the recycling box that can be used to form the BTO word. The words must be printed in the order in which their substrings appear in the BTO word. You may assume there is only one correct answer.

```
PUTTER INPUT TERMINAL
FETCH FERN WATCHER
MANUAL HUMANS SPIRITUALITY
```


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.

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

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.

6 points

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

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

Sample Input

31415926535897932384626433832795028841971693993751058209749445923078164062862

Sample Output

3 4 5 2 5 5 2 3 3 3 7 5 2 4 9 9 9 9 7 8 6
 1 1 3 2 0 1 7 6 3 3 5 1 5 2 9 9 3 7 1 4 6 8
 4 4 4 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
<pre> 10 6 ##### # / \ # # # # \ x# # > / # ##### </pre>	<pre> 10 6 ##### # v x # # / # # /# # \ # ##### </pre>
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

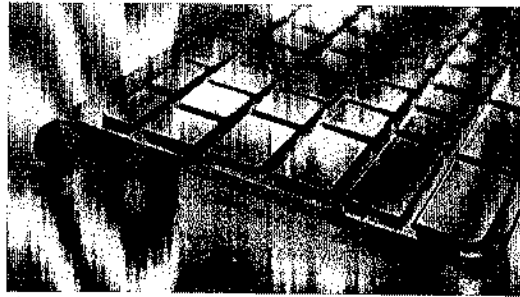
7 points

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

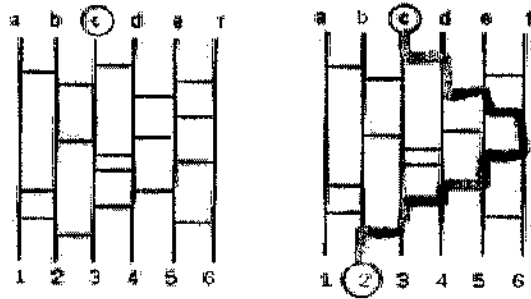


Introduction

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

Amidakuji



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

Neutron

8 points

Introduction

Help Neutron flight the lag and escape the grid-matrix. Neutron has learned about an unpublished grid-matrix API call that can attach tracers to the arena agents and track their location. Neutron 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 Neutron'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. Neutron'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, Neutron'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.

```
EERRTTTUIIIIAASSDFGGMN
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
```

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



problem 10

Egyptian Fractions*

9 points

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 $1/2$, $1/3$, $1/4$, 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
```

$$\frac{ansN}{denN} = \frac{ansN}{denN} + \frac{1}{den2}$$

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
```

a_n

$$\frac{a_1}{d_1} + \frac{a_2}{d_2}$$

$$= \frac{a_1 d_2 + a_2 d_1}{d_1 d_2}$$

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



problem 11

Fish Tales

9 points

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."

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



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 of 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 3

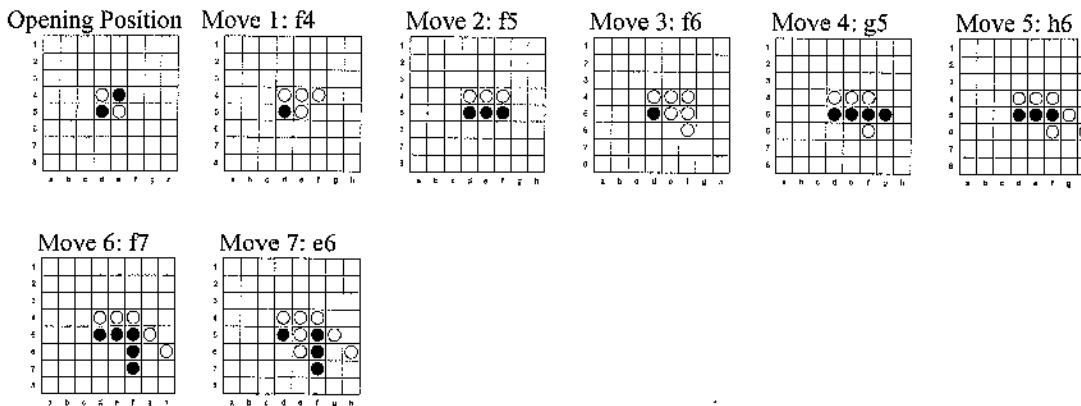
Reversi Moves

4 points



Introduction

Reversi is a game played between two players on an 8x8 grid. The opening position is shown below. White moves first. On White's turn, White must place a white piece on an empty square, such that there is at least one black piece between the new piece and another white piece, horizontally, vertically, or diagonally. After placement, all such black pieces (linearly between the new piece and the next white piece) are changed to white. The pieces must be adjacent in a single line, without empty squares. Black then plays similarly, converting one or more lines of white pieces. Play continues until either the board is filled or one player cannot make a legal move. An example of a game's first few moves is shown below.



Your program should analyze a sequence of moves for a Reversi game. The opening position is shown above. You do not need to check the validity of each move.

Sample Input

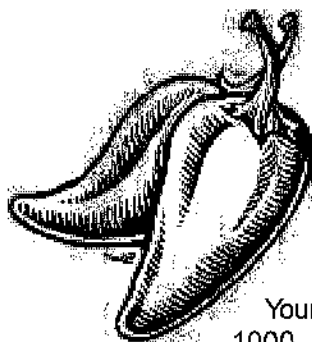
The input will be a sequence of coordinates representing the placement of pieces for each turn (white places first.) "END" terminates the sequence.

```
f4
f5
f6
g5
h6
f7
e6
END
```

Sample Output

Your program should output the state of the board to the screen after all moves are completed. Use 'B' for Black, 'W' for White, and '.' (period) for empty squares. Hint: printing the state of the board after each move may help you debug your program.

```
.....
.....
.....
...WWW..
...BWBW.
...WB.W
....B..
.....
```



Introduction

Peter's Popular Prime Pepper Plant provides packs of peppers in packages of 6, 11, or 13 peppers. The price to prepare each package is the same, regardless of size.

Your program should take as input an integer less than 1000. It should find the cheapest combination of packages to ship that number of peppers.



Sample Input

The input will consist of a single integer, representing the count of peppers.

Example 1

42

Example 2

55

Example 3

27

Example 4

88

Sample Output

The program should display the cheapest combination of packages to ship the count of peppers. If the same minimum number of packages can be obtained in two ways, choose the one that uses more of the size-13 packages.

Example 1

42 peppers can be packed most economically in:
1 package of 13
1 package of 11
3 packages of 6
5 total packages.

Example 2

55 peppers can be packed most economically in:
5 packages of 11
5 total packages.

Example 3

27 peppers cannot be packed.

Example 4

88 peppers can be packed most economically in:
5 packages of 13
1 package of 11
2 packages of 6
8 total packages.

problem 6
Heronian
Rectangles
5 points

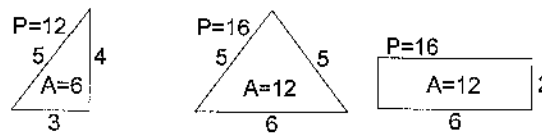
Introduction

Heron's Formula for the area of a triangle with sides of (a, b, c) is:

$$A = \sqrt{s(s-a)(s-b)(s-c)} \quad s = (a+b+c)/2$$

A Heronian Triangle is a triangle where all values above (A, a, b, c) are integers.

For CodeWars, define a Heronian Rectangle as a rectangle with integer sides with the same area and perimeter as a Heronian Triangle. For example, the smallest Heronian Triangle is (3, 4, 5), but there is no corresponding Heronian Rectangle. The Heronian Triangle (5, 5, 6) has a perimeter of 16 and an area of 12. The corresponding Heronian Rectangle has side-lengths of (2, 6).



Your program should find all pairs of Heronian Triangles and Rectangles, for a provided range of the longest side of the triangle, up to a maximum length of 500.

Sample Input

The input will be two integers representing the range for the longest side of the Heronian Triangle:

Example 1

6 13

Example 2

12 12

Example 3

490 500

Sample Output

Your program should output all triangle-rectangle pairs for the given range, one pair per line. For each pair, print the Triangle's side-lengths in increasing order, followed by the Rectangle's side-lengths in increasing order.

Example 1

(5, 5, 6) (2, 6)
(10, 10, 12) (4, 12)

Example 2

(10, 10, 12) (4, 12)

Example 3

(410, 410, 492) (164, 492)
(415, 415, 498) (166, 498)



problem 7

ASAI Skyline

6 points

Introduction

It's 1983 and you've just been hired by a videogame company (they're in "stealth mode", so you won't mention their name), and have been tasked with generating the backgrounds for their exciting new MS-DOS game. The game will use the state-of-the-art graphics, so you'll need to create some spiffy backgrounds to match the action on screen. The action takes place in the downtown area of a large Texas city - full of skyscrapers.

Because the game is dynamically building each "level" of the game, you can't just draw the backgrounds ahead of time, they must be generated on the fly. And to maintain the 3-D effects this hi-tech game requires, the buildings must be rendered in order from back to front.

Write a program that takes a list of building dimensions, and generate a skyline view of the city as output.

Sample Input

The program input consists of the number of buildings to render (maximum of 16), followed by a list of building dimensions. Each building dimension consists of: 1) the X-coordinate of the left edge of the building, 2) the width of the building, and 3) the height of the building.

```
4
4 4 8
10 10 4
13 5 6
2 4 4
```

Sample Output

The program will output the rendered skyline, using dash "-" for the roof of the building, a vertical bar "|" for each side, and hash "#" for the interior windows. The buildings are rendered back to front, such that buildings may be partially (or completely) hidden by later buildings. The output must also include a numbered X-axis wide enough to contain the image (continue printing to the next '0'). The maximum image size will be 60 columns by 20 rows, as we must leave room on the screen for the score and the command line!

```

      ----
      |##|
      |##|
      |##|
----#|  ----|###|
|##|##|  |##|###|##|
|##|##|  |##|###|##|
|##|##|  |##|###|##|
12345678901234567890
```



problem 10
A Weighty
Problem
5 points

Introduction

Five school children are weighed in pairs on a scale. The five children can be paired in ten different weights, or, ways. Given the ten paired weights, determine the weights of the individual children.

You may assume all the weights are integers.

HINT: Each child is weighed four times.



Input

Each line of input consists of ten integers. The input ends with ten zeros.

```
114 129 118 125 123 122 121 124 120 116
109 118 114 116 125 107 111 112 121 123
121 114 100 110 100 121 114 111 104 125
0 0 0 0 0 0 0 0 0 0
```

Output

The program must print the weight of each child in ascending order on a single line. Each child's weight must be an integer value.

```
56 58 60 64 65
52 55 57 59 66
45 55 55 59 66
```

Introduction

"An argument is a connected series of statements intended to establish a proposition."

"No it isn't."

"Yes it is! It's not just contradiction."

"Look, if I argue with you, I must take up a contrary position."

"Yes, but that's not just saying 'No it isn't'."

"Yes it is."

"No it isn't."

Write a program to reverse the negative of a sentence.



Input

The first line of input indicates the number of sentences to follow. Each line of input will contain a single sentence that will use the verb "is" and end in a period.

7

This is not an argument.

An argument is an intellectual process.

It is fair if you do not go.

The ferris wheel is not working.

A butterfly is beautiful, but litter is not.

A lady discerns that which is not elegant from that which is.

A lemur is a monkey and a grivet is a monkey but a chimp is not.

Output

For each sentence the program must print the sentence with the negative of the "is" inverted. That is, if the original sentence had the word "not" after the word "is", then the output should not. If the original sentence did not have the word "not" after the word "is", then the output sentence should print the word "not" immediately after the word "is". Is that clear?

This is an argument.

An argument is not an intellectual process.

It is not fair if you do not go.

The ferris wheel is working.

A butterfly is not beautiful, but litter is.

A lady discerns that which is elegant from that which is not.

A lemur is not a monkey and a grivet is not a monkey but a chimp is.

Introduction

The high school has a scrolling display that can advertise or display any message the administration wants to show. It is five pixels high, so the letters are drawn to fit in that space. The school newspaper has a webcam focused on the display and wants to translate the results into text so they can copy any important announcements.



Input

The input is a single banner string on 5 lines. It consists of letters, numbers and spaces. Each is 5 characters high and 3 characters wide. Letters and numbers are separated by 1 space. Words are separated by 5 spaces.

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

Output

The program must print the ASCII string the banner represents.

CODEWARS 2013 ROCKS

Characters seen on the banner:

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

### ### ## ## ##### ### # # # # # # # # # #
# # # # # # # # # # # # # # # # # # # # #
##### # # # # ##### ##### # # ##### # # #
# # # # # # # # # # # # # # # # # # # # #
# # ##### # # # # # # # # # # # # # # # #

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

For your convenience, the above characters can be found in the supplied file BannerCharacters.txt.

Introduction

Consider a keypad used to enter security codes to gain access to a locked door. The keypad layout looks like a typical phone number pad:

```
1 2 3
4 5 6
7 8 9
* 0 #
```



The user enters a code with four numeric digits and presses # to open the door.

Input

The input will consist of a 2D array of temperature readings from an infrared scanner. The data values are the surface temperatures of the twelve keys on the keypad. Temperature differentials are caused by heat transfer from a person who recently entered a key code.

Example input 1

```
72.1 75.0 75.1
72.0 72.1 72.2
72.0 72.2 77.9
72.1 72.2 75.2
```

Example input 2

```
77.2 77.1 76.9
77.0 83.5 77.0
77.1 77.3 77.1
77.0 79.4 79.1
```

The program will use this data to determine the four digits used in the key code. For each data set the ambient temperature and the differential temperature will vary. Also, a small amount of noise (slight variations in temperature) should be expected.

Output

The program must print a list of all possible permutations of the four digits, in ascending numeric order, without duplicates.

Example output 1

```
2399
2939
2993
3299
3929
3992
9239
9293
9329
9392
9923
9932
```

Example output 2

```
0555
5055
5505
5550
```




Introduction

A well-known shipping company has a distribution center in central Texas. Packages arrive at the truck bay by conveyor belt. There is room at the truck bay for two stacks of packages, which is where they are stacked before being loaded onto the truck. You will write a program to do the stacking.

When a package reaches the end of the conveyor it must be placed on the stack with a top surface area that most closely matches the package size. However, a package must not be placed on a stack if its dimensions exceed the length or width of the top of the stack. The floor space for each stack is 120 cm by 120 cm and no packages will be too large to fit in that space. A package should be rotated left or right (but not turned up on its side) to fit. Also, a package cannot be placed on a stack if it would cause the stack height to exceed 200 cm.

If a package cannot be placed on a stack when the other stack space is empty, then the package becomes the bottom of the other stack. A new stack may not be started if the package will fit on the existing stack. If a package cannot be placed on either stack, then the tallest stack is loaded onto the truck and the package becomes the bottom of a new stack. After all the packages are processed, any remaining stacks are loaded into the truck, in order from the tallest to the shortest.

Input

The first line of input indicates the number of packages N that need to be processed. Each line after that will have measurements in centimeters for a single package. Each package is tagged with a single letter and described by three integers: length, width, and height, in that order, in centimeters.

Example 1

```
13
A 59 109 49
B 84 76 58
C 91 54 42
D 43 75 38
E 35 45 25
F 76 89 40
G 26 29 16
H 102 84 46
I 75 75 75
J 71 71 51
K 101 82 42
L 78 88 58
M 77 87 57
```

Example 2

```
14
A 101 83 67
B 82 63 72
C 58 79 70
D 81 55 39
E 80 101 65
F 57 85 51
G 75 46 68
H 97 69 63
I 65 91 64
J 56 82 72
K 65 71 69
L 44 97 63
M 91 43 65
N 41 84 72
```

Output

The program must print one line for each stack that is loaded into the truck. The stacks must be printed in the order they are loaded. For each stack, the program must print the package tags from bottom to top.

Example 1

```
A C D E
B G
F I J
H K L
M
```

Example 2

```
A B D
C G
H I K
L M N
E F J
```



Introduction

In programming many problems can be solved by using parent-child relationships. This principle is used, for example, in the structure of HTML documents and the implementation of interface hierarchies. This approach is powerful because programs can apply rules to their data based on the notion of inheritance. Of course, the same data structure could also be used to track biological families, as in the recording and researching of people's family trees, or for selective breeding of food crops.

For this program, the relationship between two parents and one or more children is denoted by an expression like this: "A + B : C , D , E ." In this example A and B are the parents, while C, D, and E are the children.

There are two types of queries the program must be able to answer. The first is written like this: "A > D ?" which means, "is A an ancestor of D?" The second type of query is written like this: "O ^ V ?" which means, "do O and V share a common ancestor?"

Input

The first line of input is the number of parent-child expressions, followed by the expressions, one per line. Every expression will have two parents and at least one child. The next line is the number of queries, followed by the queries, one per line. Letters and symbols will be separated by a single space.

```
9
A + B : C , D , E .
F + G : H .
I + J : K , L .
C + H : M , N .
D + K : O .
L + E : P , Q , R , S .
N + Q : T , U , V .
O + S : W , X .
Y + H : Z .
7
K > O ?
F > W ?
B ^ U ?
O ^ V ?
A > Z ?
F > Z ?
X ^ Z ?
```

Output

The program must print each query, followed by the word TRUE or FALSE, indicating if the queried relationship exists. Note that a letter cannot be a parent/ancestor of itself.

```
K > O ? TRUE
F > W ? FALSE
B ^ U ? FALSE
O ^ V ? TRUE
A > Z ? FALSE
F > Z ? TRUE
X ^ Z ? FALSE
```

HP CODE WARS XVII

You hike up the steps into the bleachers to discover the next event. For some reason, all the seats in this section are numbered with only prime numbers. In the neighboring section the seats are all even numbers. The managing event coordinator explains how these numbers relate:

problem 9
Goldbach's
Conjecture
6 points

Goldbach's conjecture says that every positive even number greater than 2 is the sum of two prime numbers. This conjecture has never been proven in the general case, but it has been confirmed for numbers much larger than most programming environments' native data types.

Write a program to print the two prime numbers that sum to a given even integer. There will often be more than one answer, so print the solution with the minimum difference between the two prime numbers.

Input

Each line of input will be a positive, even integer greater than two, except the last line, which will be zero. The maximum input value will be 1000.

28
62
992
16
0

Output

The program must print the two prime numbers and the sum in equation form in ascending numeric order. Print only the solution with the minimum difference between the two prime numbers.

11 + 17 = 28
31 + 31 = 62
421 + 571 = 992
5 + 11 = 16

HP CODE WARS XVII

HP CODE WARS XVII

You dash to the closest event where the title appears to be in code:

DO TEE!ISCDH

The event coordinators explain how the title was created:

DECODE THIS! ... becomes ... DO TEE!ISCDH

problem 10
Decode
This!
6 points

There are 12 characters in both the original and encoded strings. The original first character is placed in the first encoded position. Each successive character is placed in an empty position based on the value of the last character:

Dxxxxxxxx. D=4, so the next character, E, is placed in the 4th empty position after D:

DxxxExxxxx. E=5, so the next character, C, is placed in the 5th empty position after E:

DxxxExxxxCxx. C=3, so the next character, O, is placed in the 3rd empty position after C, wrapping:

DOxxExxxxCxx. O=15, so place D in the 15th empty position after O, wrapping again.

DOxxExxxxCDx. D=4.

DOxxEExxxCDx. E=5.

DO xEExxxCDx. The space = 1.

DO TEExxxCDx. T=20.

DO TEExxxCDH. H=8.

DO TEExIxCDH. I=9.

DO TEExISCDH. S=19 (but there's only one spot left for the "I" anyway.)

DO TEE!ISCDH

Values:

A-Z = 1-26.

a-z = 1-26.

Any other characters (punctuation, numbers, spaces) have a value of 1.

Input

Each line will hold a number N, then one space, then N characters to decode. N will be at most 80. The last line will have a single 0.

```
12 Do Tee!iscdh
25 Yotei! mcgaeos'rued a drn
0
```

Output

Each line's decoded string on its own line.

```
Decode This!
You're a decoding master!
```

HP CODE WARS XVII

Summary

A "diamond in the rough" is someone or something having desirable qualities or potential but lacking experience, refinement or polish. When interviewing college graduates, technology companies usually make an effort to distinguish the diamonds in the rough from the lumps of coal.

Write a program that can generate diamond patterns of different sizes.

Input

Each line of input contains three positive integers: the size of the diamonds and the number of rows and columns of diamonds to be drawn. Due to the way the diamonds are drawn, the diamond size will always be an even integer. All values will be less than one hundred. The input ends with three zeros.

6	2	4
2	3	7
0	0	0

Output

The program must print diamonds in a rectangular grid using the slash and backslash characters for the bodies of the diamonds. A diamond of size two has exactly two slashes and two backslashes arranged in a diamond pattern. For diamonds larger than size two, the program should fill the spaces between the diamonds with hash characters to represent lumps of coal. The diamond grid must match the input rows and columns.

[illegible]

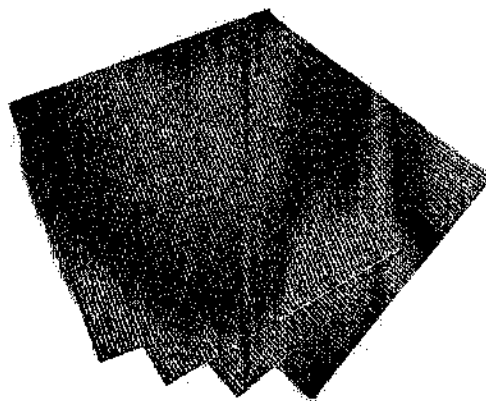
Problem 14 Squaring Rectangles

[11 points]

Summary

Our new company "Competitive Carpets", or C-squared for short, is all about customer comfort. We have an extra unique flair because we pride ourselves on conscientious covering of any room using only carpet cut in the shape of squares. In fact, we will finish a room with the fewest pieces of carpet needed.

Our carpet squares all measure an integer number of feet on each edge, and we only accept jobs for rooms with both width (W) and length (L) an integer number of feet. Obviously if a room is perfectly square, we can finish the job with a single piece of carpet. And the maximum number of carpet squares we would ever need is $L \times W$ squares, each 1-foot on a side. But we can always find a smaller number.



Input

The input is one line with two integers, the Width and Length of the room. For this problem, the maximum for each is 25.

Example 1:

11 10

Example 2:

19 18

Output

On the first line, print "N squares can cover WxL" (replacing N, W, and L with their values). On the next line print the sizes of the squares in increasing order, separated by spaces. Finally, print a map of the room, using a different lowercase letter (a, b, ...) for each square of carpet used. (No problem will ever need more than 26 squares.) Your map may not match the sample output, but your minimum number of squares must.

Example 1:

6 squares can cover 11x10.

2 2 4 5 5 6

aaaaaabbbbb

aaaaaabbbbb

aaaaaabbbbb

aaaaaabbbbb

aaaaaabbbbb

aaaaaaccccc

ddddecccc

ddddecccc

dddffcccc

dddffcccc

Example 2:

7 squares can cover 19x18.

3 5 5 7 7 8 11

aaaaaaaaaabbbbbbb

aaaaaaaaaabbbbbbb

aaaaaaaaaabbbbbbb

aaaaaaaaaabbbbbbb

aaaaaaaaaabbbbbbb

aaaaaaaaaabbbbbbb

aaaaaaaaaabbbbbbb

aaaaaaaaaabbbbbbb

aaaaaaaaaaccddddd

aaaaaaaaaaccddddd

aaaaaaaaaaccddddd

eeeeeeffffffddddd

eeeeeeffffffddddd

eeeeeeffffffggggg

eeeeeeffffffggggg

eeeeeeffffffggggg

eeeeeeffffffggggg

eeeeeeffffffggggg

Problem 15 Analysis of Acronyms

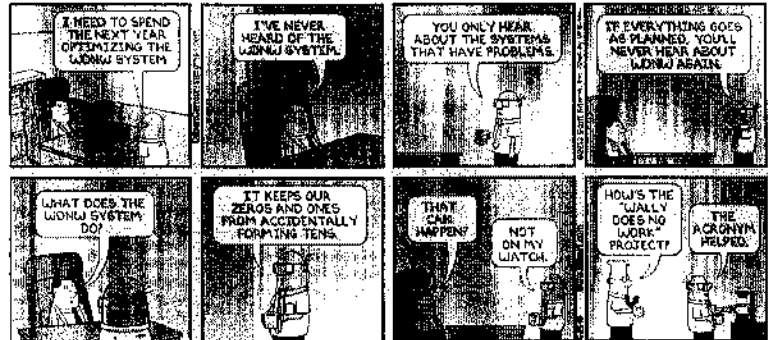
[11 points]

Summary

An acronym is an abbreviation made from the initial letters of the words in a phrase. For example, the National Aeronautics and Space Administration (NASA) uses acronyms for various projects and processes, like the Automated Retrieval and Tracking System (ARTS), Certificate of Flight Readiness (COFR), and Time Duration of Burn (TDB).

You may have noticed that sometimes (but not always) some initial letters are not used in the acronym (compare COFR with TDB). When all the words are represented, the acronym is complete. If one or more words is not represented, the acronym is partial.

The formation of acronyms can be even more interesting when more than one initial letter of a word is used, as in the Canadian space telescope called Microvariability and Oscillations of STars (MOST). For the purposes of this contest we'll only consider the first two letters of a word, even though humans are far more creative. When two initial letters from a word are used, the acronym is complex. When only initial letters are used the acronym is simple.



Write a program that can analyze a phrase and determine if a given word is an acronym, and if so, what type.

Input

The first line of input provides the number of phrases to be analyzed. Each line after consists of an acronym (candidate) followed by a phrase. The acronyms and phrases may be mixed (upper and lower) case. Words may be hyphenated.

7

SPBSG slowly pulsating B super-giants
GNU GNU is not Unix
COSTAR corrective optics space telescope axial replacement
MOST Microvariability and Oscillation of Stars
BOVINE Binary Obscure Voters Network
SoHo SOUTH OF HOUSTON STREET
BREAD Brian Edwards Analytical Design
ABJ A big Giant

Output

For each input line, the program must analyze the phrase and determine if the candidate is really an acronym for the phrase that follows, and if it is, what its properties are. The judges will expect the output to follow the patterns and wording shown in the examples below. The key words are complete vs. partial and simple vs. complex, or the use of the word not. Some acronyms could be parsed either simple or complex. In such cases the preferred analysis is simple.

SPBSG is a COMPLETE SIMPLE acronym for slowly pulsating B super-giants
GNU is a PARTIAL SIMPLE acronym for GNU is not Unix
COSTAR is a COMPLETE SIMPLE acronym for corrective optics space telescope axial replacement
MOST is a PARTIAL COMPLEX acronym for Microvariability and Oscillation of Stars
BOVINE is NOT an acronym for Binary Obscure Voters Network
SoHo is a PARTIAL COMPLEX acronym for SOUTH OF HOUSTON STREET
BREAD is a COMPLETE COMPLEX acronym for Brian Edwards Analytical Design
ABJ is NOT an acronym for A big Giant

Summary

In the early days of computing we had dot matrix printers that would output everything in a fixed width font. Those of us who were programming back then created all kinds of neat word art. Today you may have seen ASCII pictures or even ASCII animations. Let's create our own word art!

Input

The first line of input contains the number of words that follow. Each following line will contain one word up to 10 characters long. All letters will be upper case.

```
3
TEST
SAMPLE
ART
```

Output

Print each given word horizontally once and multiple times vertically so that each letter in the horizontal word matches the position of that letter in the vertical words.

The horizontal word will be in the middle of the output. The first vertical word uses the first letter of the horizontal word. The last vertical word uses the last letter of the horizontal word.

Each parallelogram should be separated from the next by a blank line.

```




  T
 TE
TES
TEST
EST
ST
T

  S
 SA
SAM
SAMP
SAMPL
SAMPLE
AMPLE
MPLE
PLE
LE
E

  A
 AR
ART
RT
T
```


Summary

Consider a block of wood that has dimensions of L, W, D (all integers, in centimeters). Now paint that block on all faces and let it dry. Finally, cut the block into 1 cm cubes. Let's say the block is:

-  "PERFECT" if the number of cubes with paint is identical to the number of cubes without paint.
-  "MORE than Perfect" if there are more painted cubes than not painted.
-  "LESS than Perfect" if there are less painted cubes than not painted.

Write a program that, given the Length, Width, and Depth of a block, outputs the classification of the block.

Input

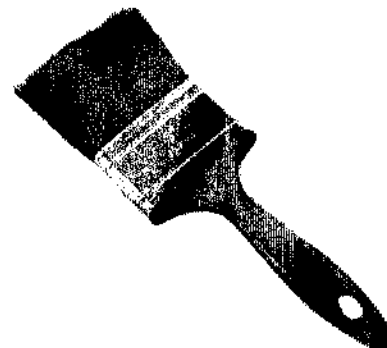
Each line of input has three integers: the Length, Width and Depth of the block. The input ends with three zeroes.

```
5 6 7
10 11 12
8 10 12
0 0 0
```

Output

The program must print the classification for each block. Capitalize only MORE, LESS, or PERFECT as appropriate.

```
A 5x6x7 block is MORE than Perfect.
A 10x11x12 block is LESS than Perfect.
A 8x10x12 block is PERFECT.
```



Problem 10 Tic-Tac-Toe

[6 points]

Summary

Write a program that will read in a Tic-Tac-Toe board configuration and determine if there was a winner. In our data representation the two players will be represented by X and O, respectively. An '=' indicates that the game ended before a move in this position was necessary.

The game of tic-tac-toe begins with an empty game board that looks like this:

```

|_|_|
|_|_|
|_|_|

```

The starting player then places an X into any board position:

```

X_|_|
|_|_|
|_|_|

```

The second player will place an O into another board position. The players then alternate moves until one player manages to place all X's or all O's along a row, column, or diagonal. Here are some example games. The first three board configurations show wins for O, X, and X respectively. The fourth board configuration shows a draw.

```

=XX  XO=  XOX  XOX
OOO  OX=  =OX  OOX
XOX  XOX  O=X  XXO

```

Input

Each line of input contains a complete board configuration of 9 characters. The first three characters represent the top row, the next three characters represent the middle row, and the last three characters represent the bottom row. The end of the input will consist of a board filled with all '='.

Note: The board configurations for this problem will never have a pair of winning directions.

```

=XXOOXOX
XO=OX=XOX
XOXOOXXO
=====

```

Output

The output should first print which player won followed by the board. The winning moves in the board should be marked by replacing the X's or O's with '\$'.

```

Player O won.
=XX
$$$
XOX

```

```

Player X won.
$O=
O$=
XO$

```

```

There was a tie.
XOX
OOX
XXO

```

Summary

Many modern computing devices have Location Awareness features. Phones, tracking systems, and self-driving vehicles have the ability to determine their current location. Often this is accomplished using a trilateration algorithm. If the device can receive signals from three sources whose locations are known, then it can determine its location from that data.

For this problem, you will write a trilateration algorithm (explained below) for an autonomous robot using signals from three towers positioned around a square arena. The arena is an integer grid with walls at the four lines defined by $x=100$, $y=100$, $x=-100$, and $y=-100$. The robot may be positioned anywhere within the arena. Tower 1 is located at (x,y) position (0,100), tower 2 at (-100,-100), and tower 3 at (100,-100).

Here's how the system works: the towers broadcast distinct signals that the robot can receive. The towers are all powered by one common battery. When the robot is near a tower, the signal strength is high, but the farther the robot is from the tower, the weaker the signal. The strength of a tower's signal is given by the following equation:

$$s = P / (d * d)$$

The variable P is the transmission power and d is the distance from the tower to the robot. When the battery is fully charged the signal is very strong. But over time, as the battery's energy is used, the signal power is reduced. So P is the same for each tower, but it changes over time. Also, the robot has no direct way to measure P . So it is not possible to make an exact calculation for the distance to a tower using the signal strength. You'll have to think of how to use all three signals to solve the problem. The only math you need to know is that the distance between two points (x_0, y_0) and (x_1, y_1) is given by this equation:

$$d = \sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2}$$

Use the signals from the three towers to determine the robot's location on the grid.

Input

Each line of input has three floating-point numbers separated by one or more spaces. These numbers are the signal strengths from towers 1, 2, and 3, in that order, for each location of the robot. The input ends with three zeros.

```
5.432 2.716 2.716
6.733 0.956 1.284
501.345 2.102 1.878
2.207 2.644 662.852
0 0 0
```

Output

For each input line, the program must print the exact integer x and y location of the robot.

```
0 0
21 35
-14 99
93 -90
```

