CODE WARS]I[PROGRAMMING COMPETITION

Greetings, Code Warrior!

Problem 1 Division II 1 Point

Problem Statement

Code Warriors live by a strict code (heh, heh) of rituals, one of which is the traditional Code Warrior greeting. Write a program to announce the names of your team members to the judges!

Program Input

There is no program input.

Program Output

The program should display on screen the name of your school *and* the names of each member of your team. It must be done according to the tradition as shown below.

Greetings, Code Warrior!
Aristophanes High School announces its team:
Blaise Pascal
Charles Babbage
Donald Knuth

CODE WARS]I[PROGRAMMING COMPETITION

Triangle Area

Problem 2 Division II 2 Points

Problem Statement

The area A of a triangle can be calculated from its base B and height H (*see figure 1*) according to the following formula:

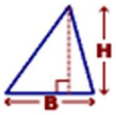


figure 1.

$$A = \frac{1}{2} * B * H$$

Write a program to read the base and height measurements of several triangles and print the area of each one.

Program Input

The program should read two real (floating point) numbers from the keyboard separated by one or more spaces. You may assume that the input numbers will be greater than zero. The program should use these values as the base and height of a triangle for the area calculation. The end of input is indicated by two zeros.

4 12
2.1 5.3
19.7 6.0
0 0

Program Output

The program should display on screen the area of the triangle as part of the complete sentence shown in the example output. The program should display at least 4 digits after the decimal, more is OK.

The area of the triangle is 24.0000
The area of the trianlge is 5.5650
The area of the trianlge is 59.1000

CODE WARS]I[PROGRAMMING COMPETITION

Data Merger

Problem 3 Division II 3 Points

Problem Statement

Merge data from 2 input files and output the results to a 3rd file.

Program Input

There will be 2 input files named PROG03A.IN and PROG03B.IN, and each file will contain the same number of lines of text. Merge the data such that the text from line 1 of file 1 gets written to the output file PROG03.OUT. Next, line 1 of file 2 gets appended to file PROG03.OUT. This repeats until all lines of text from both files are stored in the output file.

PROG03A.IN	PROG03B.IN
Line A1	Line B1
Line A2	Line B2
...	...

Program Output

The output file PROG03.OUT should contain the merged data from the 2 input files as follows.

Line A1
Line B1
Line A2
Line B2
...

CODE WARS]I[PROGRAMMING COMPETITION

Histogram

Problem 4 Division II 4 Points

Problem Statement

A histogram is a bar graph in which the area of each bar is proportional to an item's frequency of occurrence. Write a program to generate a histogram of the letters (A-Z) in a file.

Program Input

A collection of strings. The input file PROG04.IN will not contain any blank lines.

```
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG
THIS IS AN EXAMPLE OF HOW TO TEST YOUR
HISTOGRAM PROGRAM.  YOU
CAN USE THIS EXAMPLE.
```

Program Output

The program must display on screen a vertical histogram representing the occurrences of each letter of the alphabet. All letters will be uppercase (no need to check), and your program may ignore numbers, spaces, punctuation, etc. Lines will be no longer than 80 characters, and if a letter occurs more than 32 times the histogram should stop at 32.

```
                *
                *
     *          *
     *          *    *
 *   *          *   **
 *   *  *       *  ***
 *   *  **   * *  ****
 *   *  **   * ** ****
 *   * ***  ***** ****  **
 * * *****  ***** **** ***
**************************
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

CODE WARS]I[PROGRAMMING COMPETITION

Presentation

Problem 5 Division II 4 Points

Problem Statement

Our department has agreed to certain documentation conventions that should be used in our program code. "Super" Bob has been assigned to write the documentation tools, and he's currently working on a "boxed text" feature. The way it's supposed to work is this: given several lines of text data, center the text and enclose it within a box made of asterisks. *NOTE* Insert spaces and empty lines such that no characters from the text data are touching any asterisks.

Program Input

The input file will be named PROG05.IN and will contain several lines of text.

Title
Main Topic
By Sam Spoonwater

Program Output

Output the results to the output file PROG05.OUT. This file should contain the input data centered and surrounded by a box of asterisks, without any text touching the asterisks and only one space between the text and box.

```
**********************
*                    *
*       Title        *
*     Main Topic     *
* By Sam Spoonwater  *
*                    *
**********************
```

CODE WARS]I[PROGRAMMING COMPETITION

Find It

Problem 6 Division I / II 5 Points

Problem Statement

Programmers love to grep! Search an input data file using a search string and output only lines that contain that string. Searches should not be case sensitive; i.e., Put, put, and PUT should all be considered the same.

Program Input

The input file named PROG06.IN will contain several lines of text. Set up the program to receive a text input string via the keyboard, which will be used to search the input data file.

Test input.
Put text here.
Not here.
Run it on a computer.

Program Output

If the user were to enter 'put' as the input search text, the output file PROG06.OUT should contain only the lines of text that contain this string.

Test input.
Put text here.
Run it on a computer.

CODE WARS]I[PROGRAMMING COMPETITION

Pig Latin

Problem 7 Division I / II 6 Points

Problem Statement

Write a program to translate English text to Pig Latin. The rules for translating English to Pig Latin are:

1. If a word begins with a vowel, add the letters 'pay' to the end of the word. Thus, "open" becomes "openpay" and "art" becomes "artpay". For this program, vowels are a, e, i, o, u, y, and w.
2. If a word begins with one or more consecutive consonants, the entire consonant block is moved to the end of the word, and the letters 'ay' are added after it. Thus, "paper" becomes "aperpay", "stretch" becomes "etchstray", and "doesn't" becomes "oesn'tday".

Notice that any punctuation marks should remain unaffected by the translation. If a word in English is capitalized, the Pig Latin word should also be capitalized. Be sure to duplicate the word and line division during translation.

Program Input

The input file PROG07.IN will contain a series of lines of English words. The maximum length of a line is 80 characters. There will be no blank lines and no numeric digits in the file.

She brought fifteen small animals into the
courtyard, but she didn't know they were going
to run away and hide. We spent all afternoon
chasing and catching those animals!

Program Output

The program must write the Pig Latin translation of the input file to the output file PROG07.OUT, keeping the same number of words per line as the input file.

Eshay oughtbray ifteenfay allsmay animalspay intopay ethay
ourtyardcay, utbay eshay idn'tday owknay eythay werepay oinggay
otay unray andpay idehay. Wepay entspay allpay afternoonpay
asingchay andpay athcingcay osethay animalspay!

CODE WARS]I[PROGRAMMING COMPETITION

That's Amoré!

Problem 8 Division I / II 9 Points

Problem Statement

Loan payments are calculated according to a non-homogeneous linear recurrance relation. The balance owed on a loan is called *principal*. The charge for the borrowed money is called *interest*, and it is calculated as a percentage of the principal. Interest is typically published according to an annual rate, although it may be charged on a monthly basis.

If we let **P** be the initial loan principal, and let **R** be the annual interest rate, and let **N** be the number of months over which the loan is to be repaid, then we can calculate the monthly payment **M** according to the following formula:

$$M = \frac{\frac{R}{12} P \left(1 + \frac{R}{12}\right)^N}{\left(1 + \frac{R}{12}\right)^N - 1}$$

Each month when a loan payment is made, therefore, a portion of the payment is applied to interest while the remainder is applied to principal. If the principal balance in month *n* is **P***n*, then the interest amount levied that month is (**R**/12) * **P***n*. Banks and tax accountants are keenly interested (heh, heh) in knowing how much money each month is applied to each category.

Write a program to calculate loan payments and display an amortization schedule.

Program Input

The input file PROG08.IN will contain the initial principal amount, the annual interest rate, and the number of months of the loan period, in that order. Each number will be on a separate line as in the example below. There will be no blank lines.

The program does not need to validate the unit labels that appear after the numbers, but it should simply skip them. The dollar amount can range from $10000 to $99999. The interest rate may be within the range 6.0% to 18.0%. (Remember that 7.25% is a common representation of the number 7.25 / 100). The loan period may vary from 12 to 60 months.

12000.00 Dollars
7.25%
12 Months

Program Output

The program should write PROG08.OUT with three sections of information: 1) the monthly payment for the loan, 2) the amortization table, and 3) total interest and total amount paid. The amortization table must show, for each month, the amount of payment amount applied to interest and principal, plus the final balance. All dollar amounts should be rounded to cents*, and these amounts ABSOLUTELY MUST line up the decimal point in each column. Misaligned decimal points will cause the answer to be judged as incorrect!

* Tracking monetary rounding errors is a problem of its own, so for this problem we won't be concerned about rounding errors less than 3 cents.

for a loan of $12000.00, borrowed for 12 months at an annual interest rate of 7.25%, the monthly payment is $1039.70.

month	interest	principal	balance
1	72.50	967.20	11032.80
2	66.66	973.05	10059.75
3	60.78	978.93	9080.82
4	54.86	984.84	8095.98
5	48.91	990.79	7105.19
6	42.93	996.78	6108.41
7	36.90	1002.80	5105.61
8	30.85	1008.86	4096.75
9	24.75	1014.95	3081.80
10	18.62	1021.09	2060.71
11	12.45	1027.25	1033.46
12	6.24	1033.46	0.00

total interest paid: 476.45
total amount paid: 12476.45

CODE WARS]I[PROGRAMMING COMPETITION

Simple English Grammar

Problem 9 Division I / II 10 Points

Problem Statement

Many computer programs need to read and understand complex data stored or presented in a grammar. For instance, compilers parse through program source code, and some programs even have a limited natural language interface.

For this problem you will write a program to read a sequence of English words and indicate if they fit the following simple grammar model. (The square brackets indicate optional elements.)

[article] [adjective] noun [adverb] verb [[article] [adjective] noun]

Program Input

Your program must read two files. The first, ELEMENTS.TXT, contains a list of valid parts of speech: adjectives, adverbs, articles, nouns, and verbs, in that order. The file is divided into sections headed by the part of speech enclosed in square brackets. The program should allow up to 100 words for each section. If more than one form of a word is needed, each will be listed separately, such as: play, plays, and played.

```
[ADJECTIVES]
fast
red
heavy
[ADVERBS]
slowly
...
```

Secondly, each line of the file PROG09.IN contains a word sequence separated by one or more spaces with no punctuation. The file will not contain blank lines.

```
the contest quickly progresses
choosey artists choose gif
this sequence no verb
```

Program Output

For each line in the input file the program should analyze the word sequence according to the grammar pattern given above. All word compares should be case INSENSITIVE (so 'play' matches 'PLAY'). If the word sequence matches the pattern, the program should write the word 'valid' to the output file PROG09.OUT. If not, it should write 'INVALID'. The program must write one line for each line in the input file.

```
valid
valid
INVALID
```

CODE WARS]I[PROGRAMMING COMPETITION

Undo Arithmetic

Problem 10 Division I / II 11 Points

Problem Statement

Many commercial programs offer an "undo" feature that allows a user to revert back to a state prior to an operation. Write a simple arithmetic calculator with undo capabilities.

Program Input

Each line of the input file PROG10.IN follows one of these formats:

number
operator number
UNDO

The valid operators are +, -, *, and /. The number will be in %f format (no scientific notation). Operators will be separated from numbers by one or more spaces. The program should allow at least 16 levels of UNDO.

```
7.24
* 94.18
+ 13.6
/ -28.935
+ 0.0122
UNDO
UNDO
/ -29.835
+ 0.0122
UNDO
+ 0.0112
12.7
+ 3.1
UNDO
UNDO
```

Program Output

The program should write the input line to the output file PROG10.OUT along with the current value in the calculator, as shown below:

```
7.24          7.24
* 94.18       681.8632
+ 13.6        695.4632
/ -28.935     -24.03536201832
+ 0.0122      -24.02316201832
UNDO          -24.03536201832
UNDO          695.4632
/ -29.835     -23.31031339031
+ 0.0122      -23.29811339031
UNDO          -23.31031339031
+ 0.0112      -23.29911339031
12.7          12.7
+ 3.1         15.8
UNDO          12.7
UNDO          -23.29911339031
```

CODE WARS]I[PROGRAMMING COMPETITION

Symbolic Artificial Intelligence (AI)

Problem 11 Division I 12 Points

Problem Statement

You have been given the job of writing a simple symbolic AI processor which must read some facts and correctly answer some questions. Fortunately, the facts and questions relate to specific individuals, who belong to groups which act on other groups. The grammar is very simple: there are two types of statements and one type of question, and all group nouns are always plural.

Program Input

Each line of PROG11.IN contains a statement or a question. The end of the input is indicated by a line whose first character is a #. Each line has one of the following forms:

- A statement of the form "<noun1>s <verb> <noun2>s." This means members of group <noun1> do <verb> to members of group <noun2>.
- A statement of the form "<id> is a <noun>." This means that the particular individual <id> belongs to group <noun>.
- A question of the form "Does <id1> <verb> <id2>?" This asks whether <id1> does the action <verb> to <id2>.

Lines are up to 80 characters long. Words are up to 40 characters long and may consist of upper and lower case letters, although searches should be case insensitive. There may be multiple spaces between words, but there are never spaces between words and the punctuation. Allow for up to 100 facts in the input.

Dogs chase cats.
Dogs like kids.
Fido is a dog.
Fluffles is a cat.
Lydia is a kid.
Does Fido chase Fluffles?
Does Fluffles chase Fido?
Does Fido like Lydia?
#

Program Output

For each question, the program should (1) echo the question to PROG11.OUT, and (2) print 'Yes' if the "facts" imply that the question is true, and 'No.' otherwise.

Does Fido chase Fluffles? Yes.
Does Fluffles chase Fido? No.
Does Fido like Lydia? Yes.

source: 1995 ACS Australian Programming Competition.

CODE WARS]I[PROGRAMMING COMPETITION

Handprinting Sleuth

Problem 12 Division I 12 Points

Problem Statement

Character recognition - Given 8x8 bitmaps made up of 8 bytes of data, determine if the letter 'A' is present anywhere within the bitmap. There will be only one character within the 8x8 bitmap, the character will be upright, and the character 'A' (if present) will exactly match the predefined 5x6 character 'A'.

Predefined 'A'

00100
01010
10001
11111
10001
10001

Program Input

The input file will be named PROG12.IN and will contain several sets of 8 numbers. Each number will range from 0-255 and will represent a binary byte of data representing a row of 8 bits in an 8x8 bit array.

32,80,136,248,136,136,0,0
255,0,136,136,248,0,163,49
0,8,20,34,62,34,34,0

Program Output

Output the results to the output file PROG12.OUT. This file will simply contain a response indicating whether each 8x8 bit array contained the letter 'A'.

Yes
No
Yes

CODE WARS]I[PROGRAMMING COMPETITION

Maze Surfing

Problem 13 Division I 14 Points

Problem Statement

You dirty rat. You're caught in a malicious lab experiment, dropped into a maze. Your job is to find the cheese and thereby to prove the superior intelligence of the various rodent species.

Program Input

The input file PROG13.IN describes a single maze. The first line of the file is an integer, N, indicating the number of rows and columns in the maze, constrained by: 8 < N < 19. The next N lines each contain N characters separated by spaces. Walls are represented by '#', hallways by '.', the start position by 'M', and the cheese by 'C'. For example:

```
10
# # # # # # # # # #
# . # M . . # . . #
# . # # . # # # . #
# . . . . . . . . #
# # # # . # . # # #
# . . . . # . # C #
# # # # # # . # . #
# . # . . . . # . #
# . . . # . . . . #
# # # # # # # # # #
```

Program Output

The program should write the maze to PROG13.OUT, drawing the shortest path from M to C with a series of '+' characters. You may assume that (1) there *will be* a valid path from M to C, and (2) there will *only be one* path from M to C (i.e., no loops). The path is not permitted to contain any diagonal movements. The solution for the above input file is:

```
# # # # # # # # # #
# . # M + . # . . #
# . # # + # # # . #
# . . . + + + . . #
# # # # . # + # # #
# . . . . # + # C #
# # # # # # + # + #
# . # . . . + # + #
# . . . # . + + + #
# # # # # # # # # #
```

CODE WARS]I[PROGRAMMING COMPETITION

Four Word Cross Word

Problem 14 Division I 17 Points

Problem Statement

Crossword puzzles and word search puzzles have been a popular pastime for many people throughout the years. Write a program that can arrange four words in a rectangular crossword pattern.

Program Input

The input file PROG14.IN will contain four words, each on a separate line. For example:

economics
speech
chemistry
communications

Program Output

The program should write a rectangular crossword to the output file PROG14.OUT. Each word must be connected to two other words in such a way as to form a rectangular region as shown in the example below. All words must read either top to bottom or left to right. They may be arranged and connected in any order, as long as the output forms a rectangle in the center. Diagonal words are not permitted.

If the program is unable to find a solution to the input, it should write the message NO SOLUTION FOUND.

```
  e
  chemistry
  o    p
  n    e
  o    e
communications
  i    h
  c
  s
```

CODE WARS]I[PROGRAMMING COMPETITION

Traveling Chess Salesman

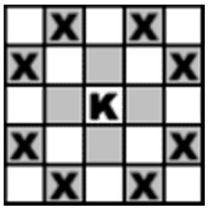
Problem 15 Division I 19 Points

Problem Statement

A wee time ago, there once lived a Scottish chess salesman who used to offer a free chess set to any potential customer who could move the Knight piece around the board in the minimum number of moves and land on every square. There were three simple rules. One, the Knight had to start in the Queen's Knight position. Two, no other pieces were on the board. And three, each square could be landed on multiple times, but each move had to count. Of course, the customer would have to agree to buy a chess set if they could not match the salesman's best effort of 63 moves. Legend has it that this chess salesman used to chant the following Lyric before the challenge began:

In days of old when Knights were bold,
And the game was invented,
The Knight was alone when he went off to course,
And had to be contented.

Without any clues, in how little moves,
Could a Knight cover her Lord's green?
You've got to make her cover every acre,
And be certain of where's She's been.



Knight **K** can move to any space denoted by an **X**

Program Input

The program should model an 8x8 chessboard and a single Knight starting in the top row. The Knight may move horizontally two spaces and vertically one space, or vertically two spaces and horizontally one space, as shown in the diagram above. PROG15.IN contains a single number (1-8) indicating the column of the Knight's starting position.

7

Program Output

The program must write to PROG15.OUT an 8x8 matrix of numbers 1-64 indicating the jumps the Knight makes to cover the entire chessboard. The numbers indicate the Knight's jump (or move) sequence along the path, and they must line up in 8 rows and 8 columns. The example shows the first 13 moves for one possible solution:

```
09 -- 03 -- 07 -- 01 --
04 -- 08 -- 02 -- -- --
-- 10 -- 06 -- 12 -- --
-- 05 -- 11 -- -- -- 13
-- -- -- -- -- -- -- --
-- -- -- -- -- -- -- --
-- -- -- -- -- -- -- --
-- -- -- -- -- -- -- --
```