**Filename: GOLDBACH**

The **Goldbach conjecture** is one of the oldest unsolved problems in number theory. It states that every even number larger than two can be expressed as a sum of two prime numbers. Write a program that reads a sequence of even positive integers. The first odd number terminates the input sequence. For each input number, the program finds and prints two prime numbers that sum to the input value. See the examples below for output formatting. If there is more than one possibility to write a number as a sum of primes, the program will print one of the possibilities of your choice. You can assume that the input is valid and not empty (there is at least one even number in the sequence before the terminating odd number and all even numbers in the input are larger than two). Also, there is no limit on the amount of even numbers that could be entered. The size of an input even number is only limited by the size of an integer in your programming environment.

***Example 1***

**Input:** 4

**Output:** 4 = 2 + 2

**Input:** 6

**Output:** 6 = 3 + 3

**Input:** 8

**Output:** 8 = 3 + 5

**Input:** 3

*This input terminates the program execution.*

***Explanation for the Output:*** Instead of 8 = 3 + 5 the program could print 8 = 5 + 3. Your program doesn’t need to print all possible combinations. The program should output one of the possibilities.

***Example 2:***

**Input:** 10

**Output:** 10 = 5 + 5

**Input:** 6

**Output:** 6 = 3 + 3

**Input:** 4

**Output:** 4 = 2 + 2

**Input:** 12

**Output:** 12 = 5 + 7

**Input:** 8

**Output:** 8 = 3 + 5

**Input:** 1

*This input terminates the program execution*

***Explanation for the Output:*** Instead of 10 = 5 + 5 the program could print 10 = 3 + 7 or 10 = 7 + 3.

And instead of 12 = 5 + 7 the program could print 12 = 7 + 5.

And instead of 8 = 3 + 5 the program could print 8 = 5 + 3.

Your program doesn’t need to print all possible combinations. The program should output one of the possibilities.

***Example 3:***

**Input:** 14

**Output:** 14 = 7 + 7

**Input:** 14

**Output:** 14 = 7 + 7

**Input:** 56

**Output:** 56 = 3 + 53

**Input:** 7

*This input terminates the program execution*

***Explanation for the Output:*** Instead of 14 = 7 + 7 the program could print 14 = 3 + 11 or 14 = 11 + 3.

And instead of 56 = 3 + 53 your program could print 56 = 53 + 3 or 56 = 13 + 43 or

56 = 43 + 13 or 56 = 19 + 37 or 56 = 37 + 19. Your program doesn’t need to print all possible combinations. The program should output one of the possibilities.

**Filename: ROMAN**

Roman numerals are an ancient numbering system.

**The Problem:** Add two roman numerals, expressing the result in roman numerals.

**The Input:** Several pairs of roman numerals. Only the capital letters I, V, X, L, C, D and M will be used. Each roman numeral will be on a separate line, starting in column one. The roman numerals and their sums are guaranteed to be within the range of 1 to 3999, inclusive. Input 0 to terminate input.

**The Output:** For each pair of roman numerals, print the sum, expressed in roman numerals. (Capital letters only, please.)

**Sample Input:**

III

VI

XXIII

XXVII

MCMLXXXIV

DXIV

MMMCM

XCIX

0

Sample Output:

IX

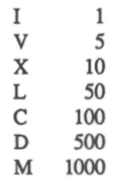
L

MMCDXCVIII

MMMCMXCIX

**Roman Numeral Reference**

Roman Numeral Symbols:



The roman numeral representation of a number is the shortest possible combination of the above symbols which add up to the number, arranged in descending order. 4's and 9's are handled specially, by subtraction. A smaller symbol is placed before a larger one, and is subtracted from it. For example, IV is four, and XC is ninety. The representation of each number is unique- there can only be one representation for any number.

***Filename: PIGLATN***

Pig latin is a simple code. To convert a word to pig latin, break it at the first vowel, put the leading consonants ***at*** the end, followed by 'ay.' If the first letter is a vowel, just put 'hay' at the end. For example, 'cat' becomes 'atcay', 'frog' becomes 'ogfray' and 'ant' becomes 'anthay'.

The Problem:  
Convert a paragraph into pig latin.

**The Input:**

A paragraph. Words willeither be separated either by a single space, ***or*** by a period and a single space. The letter 'Y' will never be the first vowel, so you won't have to handle **words** like psychology or xylophone. Capital letters **will** only appear **as** the first letter of a word. There **will** be no numbers, contractions, hyphenations, etc. - just words.

**The Output:**

The paragraph, converted into pig latin. Note that you must maintain proper leading capitalization (Cat becomes Atcay.)

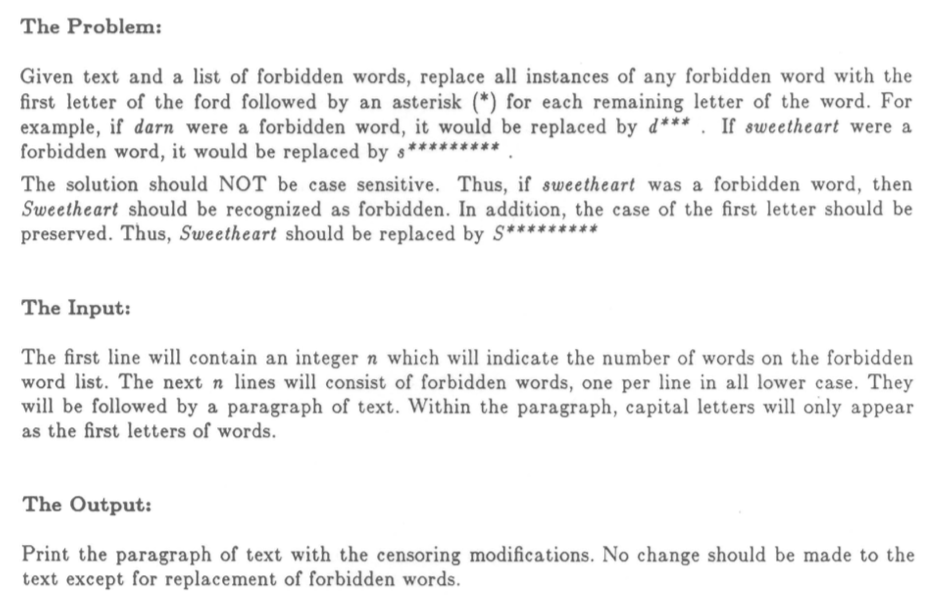
**Sample Input:**

This is a test of the Pig Latin program. These simple phrases should just about test all conditions. Good luck.

**Sample Output:**

Isthay ishay ahay esttay ofhay ethay Igpay Atinlay ograrnpray. Esethay implesay asesphray ouldshay ustjay abouthay esttay allhay onditionscay. Oodgay ucklay.

***Filename: CENSORED***

****

**Sample Input:**

4  
statements  
particularly  
substitutions  
substituted  
Equally frequent is the case where a certain sequence of statement appears in various places of the program in not exactly identical but highly similar form. Particularly important is the situation where the difference between the individual occurrences of the statements can be eliminated by systematic substitution's of identifiers or expressions. In this case, the statements to be abbreviated can be abstracted into a procedure schema. The entities still to be substituted for the individual occurrences  
are called procedure parameters.

**Sample Output:**

Equally frequent is the case where a certain sequence of statement appears in various places of the program in not exactly identical but highly similar form. P\*\*\*\*\*\*\*\*\*\*\* important is the situation where the difference between the individual occurrences of the s\*\*\*\*\*\*\*\*\* can be eliminated by systematic s\*\*\*\*\*\*\*\*\*\*\*\* of identifiers or expressions. In this case, the s\*\*\*\*\*\*\*\*\* to be abbreviated can be abstracted into a procedure schema. The entities still to be s\*\*\*\*\*\*\*\*\*\* for the individual occurrences

are called procedure parameters.

**Filename: SNAIL**

**The Problem:**

Given a two-dimensional matrix, your program is to print the matrix elements in a "snail-like" fashion as follows:

print the elements in the top row from left to right  
print the elements in the rightmost column from top to bottom print the elements in the bottom row from right to left print the elements in the left most column from bottom to top - print the elements in the second from top row from left to right ...

Note that each element should be printed only once.

**The Input:**

The input is divided into sets. Each input set starts with two integer values for m and n, the dimensions of a matrix (both values are between 1 and 25, inclusive, and the two values are separated by exactly one space). The next m input lines contain n values for matrix rows, one row per input line. Assume that each matrix element is a single non-space character and that there is exactly one space between matrix elements in the input. End of data is indicated by end of file.

**The Output:**

For each matrix, print the elements in a "snail-like" fashion. Leave no space between elements. Leave one blank line after the output for each matrix.

**Sample Input:**

3 4

|  |  |  |  |
| --- | --- | --- | --- |
| A | B | C | D |
| J | K | L | E |
| I | H | G | F |

**Sample Output**

ABCDEFGHI JKL

**Sample Input:**

56

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | F |
| R | S | T | U | V | G |
| Q | Z | Z | Z | W | H |
| P | Z | Z | Y | X | I |
| O | N | M | L | K | J |

**Sample Output**

ABCDEFGHIJKLMNOPQRSTUVWXYZZZZZ