1999 Denison Spring Programming Contest
Problem 1
Limit Cycles of xy (mod x+y)

For any two positive integers x and y define $F(x, y) = xy (\text{mod } x + y)$. Given two initial values x_0 and x_1 form a sequence using the recurrence

$$x_n = F(x_{n-1}, x_{n-2}).$$

For some initial values this eventually produces a constant sequence. For example, if $x_0 = x_1$ is odd, then $x_n = x_0$ for all n. Indeed most initial values lead to a fixed point but some lead to a limit cycle. The smallest limit cycle is 5,7,11. (Of course, then 7,11,5 and 11,5,7 are also limit cycles.) This problem is to read in the two initial values of the sequence, then determine if the sequence reaches a limit point or a limit cycle. If it reaches a limit cycle, print out the cycle.

There will be multiple test cases. Input for each test case will be two positive integers (both less than 1000), giving x_0 and x_1, separated by a space, on one line. If $x_0 = x_1 = 0$, input terminates. Each test case should generate one line of output. This line should be one of the two following forms:

`The sequence has a fixed point of n.`
`The sequence has a limit cycle of length n:` $a_1, a_2, \ldots, a_n$.

In the first example line, `n` is replaced by the integer that is the fixed point. In the second example line, `n` is the smallest cycle length and $a_1, a_2, \ldots, a_n$ is the limit cycle. (Any limit cycle of length `n` will do here.) It's the case you will always be able to determine either a fixed point or the limit cycle within the first 100 elements of the sequence, for the numbers we're dealing with.

EXAMPLE:

Input:	should generate output:
`3 5`	`The sequence has a limit cycle of length 3:  7, 11, 5.`
`304 384`	`The sequence has a limit cycle of length 4:  96, 304, 384, 464.`
`4 8`	`The sequence has a fixed point of 0.`
`0 0`	

Of course, the limit cycles shown here could be written in another order.

1999 Denison Spring Programming Contest
Problem 2
Parentheses

In how many ways can n pairs of parentheses be legally arranged? For example, 3 pairs of parentheses can be legally arranged in 5 different ways:

$$()()() \quad (())() \quad ()(()) \quad (()()) \quad ((())).$$

You will process multiple test cases. Input for each test case will consist of one positive integer n on a line. (You may assume n is no larger than 12.) Input of $n = 0$ indicates end of input. For each line of input should generate one line of output containing the sentence

`There are` c `balanced expressions with` n `pairs of parentheses.`

where c and n are replaced by the appropriate values.

EXAMPLE:

Input:	should generate output:
`3`	`There are 5 balanced expressions with 3 pairs of parentheses.`
`5`	`There are 42 balanced expressions with 5 pairs of parentheses.`
`0`	

1999 Denison Spring Programming Contest
Problem 3
Nimbers

The non-negative integers can be given a new addition that is both commutative and associative. Here are the new rules: If two powers of 2 of the same value are added, the result is 0. If the larger of the two numbers is a power of 2, the result is the same as usual addition. We call the non-negative integers with this new addition, *nimbers*.

For example,

$$5 + 6 = (4 + 1) + (4 + 2) = (4 + 4) + (2 + 1) = 0 + (2 + 1) = 3.$$

(Clarification: What is required here is to decompose the integer as a sum of powers of 2 and do so so that no power of 2 occurs more than once.)

There will be multiple test cases for this problem. Input for each test case will be two non-negative integers, n and m, separated by a single space, on one line. ($n = m = 0$ indicates end of input.) Here, n and m will be no greater than 1000. For each line of input, generate 5 lines of output. The first line should be the 4 results of adding n to $m, m + 1, m + 2$ and $m + 3$. The second line should be the 4 results of adding $n+1$ to $m, m+1, m+2$ and $m+3$. The third line should be the 4 results of adding $n+2$ to $m, m+1, m+2$ and $m + 3$. The fourth line should be the 4 results of adding $n + 3$ to $m, m + 1, m + 2$ and $m + 3$. Each number should be printed right-justified in a field width of 5. The fifth line should be blank. In other words, you are to produce a 4-by-4 chunk of the nimber addition table.

EXAMPLE:

Input:

```
10 14
903 37
15 0
0 0
```

should generate output:

```
    4    5   26   27
    5    4   27   26
    2    3   28   29
    3    2   29   28

  930  929  928  943
  941  942  943  928
  940  943  942  929
  943  940  941  930

   15   14   13   12
   16   17   18   19
   17   16   19   18
   18   19   16   17
```

1999 Denison Spring Programming Contest
Problem 4
Zig-Zag

A diamond pattern $\begin{array}{ccc} & a & \\ b & & c \\ & d & \end{array}$ is called *tight* if $b + c$ is one greater than $a + d$. Now consider the following pattern of non-negative integers:

$$\begin{array}{ccccccccccc}
 & 0 & & 0 & & 0 & & 0 & & 0 & \\
0 & & ? & & ? & & ? & & ? & & ? \\
 & 0 & & ? & & ? & & ? & & ? & \\
0 & & ? & & ? & & ? & & ? & & ? \\
 & 0 & & ? & & ? & & ? & & ? & \\
0 & & ? & & ? & & ? & & ? & & ? \\
 & 0 & & 0 & & 0 & & 0 & & 0 &
\end{array}$$

When we figure out the values for all the ? to make this pattern tight, we get the following:

$$\begin{array}{ccccccccccc}
 & 0 & & 0 & & 0 & & 0 & & 0 & \\
0 & & 1 & & 3 & & 5 & & 4 & & 2 \\
 & 0 & & 3 & & 7 & & 8 & & 5 & \\
0 & & 1 & & 6 & & 9 & & 8 & & 3 \\
 & 0 & & 3 & & 7 & & 8 & & 5 & \\
0 & & 1 & & 3 & & 5 & & 4 & & 2 \\
 & 0 & & 0 & & 0 & & 0 & & 0 &
\end{array}$$

Of course, we could start the border given above with something besides all zeros. We will supply any collection of non-negative integers where each is no larger than 100.

In this problem you will be given the border and are to print out the resulting tight pattern. Specifically, there will be multiple test cases for input. Each test case will be 7 lines. The first line will be the 5 integers, separated by spaces, of the first row of the pattern: $a\ b\ c\ d\ e$. ($a = -1$ indicates end of input and there will be no more lines following that one.) The 2nd, 3rd, 4th, 5th and 6th lines will each contain a single integer. (Let's call them f, g, h, i and j, respectively.) The 7th line will contain the 5 integers, separated by spaces, of the last row of the pattern: $k\ l\ m\ n\ p$. No integer will be larger than 20. Again, this represents the starting pattern:

$$\begin{array}{ccccccccccc}
 & a & & b & & c & & d & & e & \\
f & & ? & & ? & & ? & & ? & & ? \\
 & g & & ? & & ? & & ? & & ? & \\
h & & ? & & ? & & ? & & ? & & ? \\
 & i & & ? & & ? & & ? & & ? & \\
j & & ? & & ? & & ? & & ? & & ? \\
 & k & & l & & m & & n & & p &
\end{array}$$

You are to fill in the values for all the ? and print the entire zig-zag pattern. Print each number right-justified in a field width of 4 and start the odd-numbered rows with 2 blanks, to offset them appropriately. One blank line should follow each pattern.

(Sample input and output on next page)

EXAMPLE:

Input:

```
 0 0 0 0 0
 0
 0
 0
 0
 0
 0 0 0 0 0
 2 3 1 4 5
 6
 0
 2
 1
 3
 10 2 3 6 1
 -1 0 0 0 0
```

should generate output:

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

      2   3   1   4   5
    6  -3   5  13   2  10
      0  -2  16  10   6
    2   0   8  12  13  11
      1   9   3  14  17
    3   9   3   4  17  2
     10   2   3   6   1
```

1999 Denison Spring Programming Contest
Problem 5
Crossword

Your internship at the local newspaper includes getting the daily crossword puzzle into shape. Among other things, you are given the blank grid on which the crossword puzzle is to be solved and you are to come up with the numbering. The grid is always square and has somewhere between 8 and 20 rows, inclusive. The squares that are blacked out are given and you need to number the squares that need numbering. The numbering system for a crossword puzzle works as follows: Numbering is consecutive integers starting with 1. The numbering proceeds across the first row, then the second row, etc. A square is numbered if it is either the start of a horizontal word or the start of a vertical word, or both. A word must be at least 2 characters long.

In addition to numbering the squares, you are to count how many *across* clues there are and how many *down* clues there are.

There will be multiple test cases for this problem. Each test case will start with an integer n on a line. If $n = 0$, input terminates. If $n > 0$, there will be n lines following, each consisting of a string of n zeros (0) and ones (1), indicating the squares in each row. The first such line gives the first row of squares, left-to-right, and so on. A 0 indicates an open square and a 1 indicates a blacked out square. The zeros and ones will have no spaces separating them. Each square will either be blacked out or part of a word; that is, there will be no isolated single open squares.

For each test case, you should produce the numbered crossword, in the format given below, followed by a blank line, followed by the line

`There are` a `across clues.`

followed by the line

`There are` d `down clues.`

followed by another blank line. Of course, a and d are to be filled in with the appropriate numbers.

The output of the crossword should be in the typical orientation (first square of the first row in the upper left-hand corner) with one row per line. Each square of the crossword should be represented in a field width of 3. If the square is to be numbered, then the number should be right-justified in the field. If the square is open but not numbered, the character `-` (hyphen) should appear right-justified. If the square is blacked out, `XXX` should appear in the field.

EXAMPLE:

Input:

```
10
0000100100
0000100100
1100000000
0001010011
0001000100
0010001000
1100101000
0000000011
0010010000
0010010000
0
```

should generate output:

```
  1  2  3  4XXX  5  6XXX  7  8
  9  -  -  -XXX 10  -XXX 11  -
XXXXXX 12  - 13  -  - 14  -  -
 15 16  -XXX  -XXX 17  -XXXXXX
 18  -  -XXX 19 20  -XXX 21 22
 23  -XXX 24  -  -XXX 25  -  -
XXXXXX 26  -XXX  -XXX 27  -  -
 28 29  -  - 30  - 31  -XXXXXX
 32  -XXX 33  -XXX 34  - 35 36
 37  -XXX 38  -XXX 39  -  -  -

There are 24 across clues.
There are 24 down clues.
```