Uva 441 Lotto

In the German Lotto you have to select 6 numbers from the set {1,2,...,49}.

A popular strategy to play Lotto - although it doesn't increase your chance of winning - is to select a subset S containing *k* (*k*>6) of these 49 numbers, and then play several games with choosing numbers only from S.

For example, for *k*=8 and S = 1,2,3,5,8,13,21,34 there are 28 possible games: [1,2,3,5,8,13], [1,2,3,5,8,21], [1,2,3,5,8,34], [1,2,3,5,13,21], ..., [3,5,8,13,21,34].

Your job is to write a program that reads in the number k and the set S and then prints all possible games choosing numbers only from S.

Input Specification

The input file will contain one or more test cases.

Each test case consists of one line containing several integers separated from each other by spaces. The first integer on the line will be the number *k* (6 < *k* < 13). Then *k* integers, specifying the set S, will follow in ascending order.

Input will be terminated by a value of zero (0) for *k*.

Output Specification

For each test case, print all possible games, each game on one line.

The numbers of each game have to be sorted in ascending order and separated from each other by exactly one space. The games themselves have to be sorted lexicographically, that means sorted by the lowest number first, then by the second lowest and so on, as demonstrated in the sample output below.

The test cases have to be separated from each other by exactly one blank line. Do not put a blank line after the last test case.

Sample Input

```
7 1 2 3 4 5 6 7
8 1 2 3 5 8 13 21 34
0
```

Sample Output

```
1 2 3 4 5 6
1 2 3 4 5 7
1 2 3 4 6 7
1 2 3 5 6 7
1 2 4 5 6 7
1 3 4 5 6 7
2 3 4 5 6 7

1 2 3 5 8 13
1 2 3 5 8 21
1 2 3 5 8 34
1 2 3 5 13 21
1 2 3 5 13 34
1 2 3 5 21 34
1 2 3 8 13 21
1 2 3 8 13 34
1 2 3 8 21 34
1 2 3 13 21 34
1 2 5 8 13 21
1 2 5 8 13 34
1 2 5 8 21 34
1 2 5 13 21 34
1 2 8 13 21 34
1 3 5 8 13 21
1 3 5 8 13 34
1 3 5 8 21 34
1 3 5 13 21 34
1 3 8 13 21 34
1 5 8 13 21 34
2 3 5 8 13 21
2 3 5 8 13 34
2 3 5 8 21 34
2 3 5 13 21 34
2 3 8 13 21 34
2 5 8 13 21 34
3 5 8 13 21 34
```

In chess it is possible to place eight queens on the board so that no one queen can be taken by any other. Write a program that will determine all such possible arrangements for eight queens given the initial position of one of the queens.

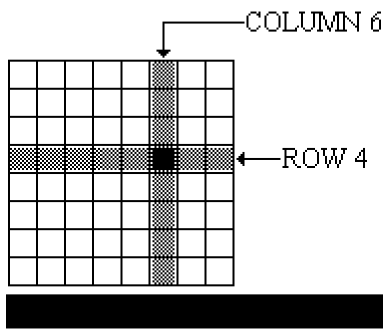
Do not attempt to write a program which evaluates every possible 8 configuration of 8 queens placed on the board. This would require 8^8 evaluations and would bring the system to its knees. There will be a reasonable run time constraint placed on your program.

Input

The first line of the input contains the number of datasets, and it's followed by a blank line. Each dataset will be two numbers separated by a blank. The numbers represent the square on which one of the eight queens must be positioned. A valid square will be represented; it will not be necessary to validate the input.

To standardize our notation, assume that the upper left-most corner of the board is position (1,1). Rows run horizontally and the top row is row 1. Columns are vertical and column 1 is the left-most column. Any reference to a square is by row then column; thus square (4,6) means row 4, column 6.

Each dataset is separated by a blank line.



Output

Output for each dataset will consist of a one-line-per-solution representation.

Each solution will be sequentially numbered $1 \ldots N$. Each solution will consist of 8 numbers. Each of the 8 numbers will be the ROW coordinate for that solution. The column coordinate will be indicated by the order in which the 8 numbers are printed. That is, the first number represents the ROW in which the queen is positioned in column 1; the second number represents the ROW in which the queen is positioned in column 2, and so on.

The sample input below produces 4 solutions. The full 8 × 8 representation of each solution is shown below.

DO NOT SUBMIT THE BOARD MATRICES AS PART OF YOUR SOLUTION!

```
   SOLUTION 1            SOLUTION 2            SOLUTION 3            SOLUTION 4

1 0 0 0 0 0 0 0       1 0 0 0 0 0 0 0       1 0 0 0 0 0 0 0       1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0       0 0 0 0 0 0 1 0       0 0 0 0 0 1 0 0       0 0 0 0 1 0 0 0
0 0 0 0 1 0 0 0       0 0 0 1 0 0 0 0       0 0 0 0 0 0 0 1       0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 1       0 0 0 0 0 1 0 0       0 0 1 0 0 0 0 0       0 0 0 0 0 1 0 0
0 1 0 0 0 0 0 0       0 0 0 0 0 0 0 1       0 0 0 0 0 0 1 0       0 0 1 0 0 0 0 0
0 0 0 1 0 0 0 0       0 1 0 0 0 0 0 0       0 0 0 1 0 0 0 0       0 0 0 0 0 0 1 0
0 0 0 0 0 1 0 0       0 0 0 0 1 0 0 0       0 1 0 0 0 0 0 0       0 1 0 0 0 0 0 0
0 0 1 0 0 0 0 0       0 0 1 0 0 0 0 0       0 0 0 0 1 0 0 0       0 0 0 1 0 0 0 0
```

Submit only the one-line, 8 digit representation of each solution as described earlier. Solution #1 below indicates that there is a queen at Row 1, Column 1; Row 5, Column 2; Row 8, Column 3; Row 6, Column 4; Row 3,Column 5; ... Row 4, Column 8.

Include the two lines of column headings as shown below in the sample output and print the solutions in lexicographical order.

Print a blank line between datasets.

Sample Input

```
1

1 1
```

Sample Output

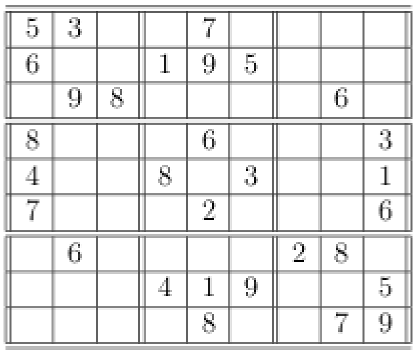
```
SOLN        COLUMN
 #       1 2 3 4 5 6 7 8

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

d060: [2005] A - Sudoku Solver

內容：

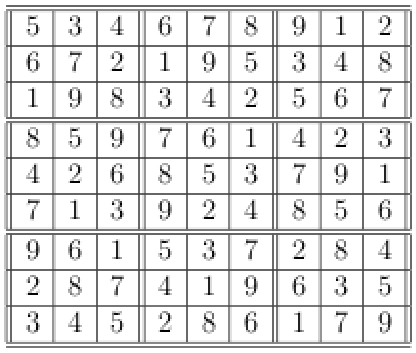
Sudoku is a popular puzzle demanding logic and patience. The aim of the puzzle is to enter a numeral from 1 through 9 in each cell of a 9 × 9 grid made up of 3 × 3 subgrids, starting with various numerals given in some cells(called given). Each row, column and subgrid must contain only one instance of each numeral. See Figure 1 and Figure 2 for an example and its solution.

Write a program that output the correct solutions to the puzzles in the input file. You can solve these puzzles with or without the help of computer



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

Figure 1: An example of a Sudoku puzzle



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

Figure 2: The solution

輸入說明：

There is no need to read in the input file during your program execution as the input file is already provided to you. There are 20 Sudoku puzzles in the input file with the following format: Each row contains 9 consecutive numerals, 0 indicates that the cell has to be filled, and other numerals indicate that the cell is given that numeral. Each puzzle consists of 9 rows, there are 180 rows in total.

輸出說明：

Each row contains 9 consecutive numerals. There are 180 rows in total. And row 1 to 9 is the answer to the first puzzle and row 9i – 8 to row 9i is the answer to the ith puzzle.

範例輸入：

```
530070000
600195000
098000060
800060003
400803001
700020006
060000280
000419005
000080079
```

範例輸出 ：

```
534678912
672195348
198342567
859761423
426853791
713924856
961537284
287419635
345286179
```

提示 ：

出處 ：

NCPC 2005