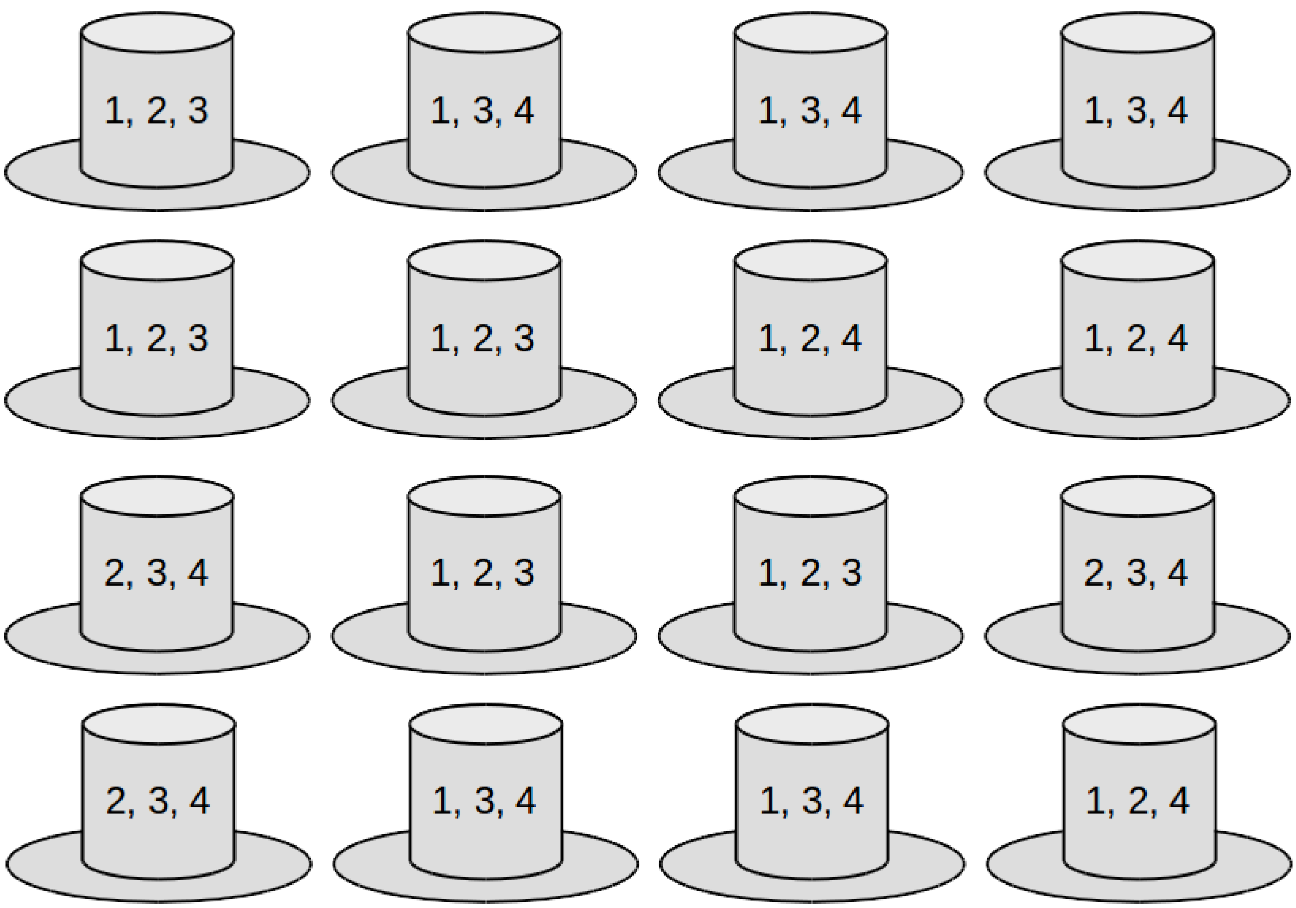
Magical Hats

Young Matt loves hats and, for that reason, plans to join a magician's guild. A magician needs to be a logical thinker to see past cheap illusions and discern what is truly magic. For this reason the magician's guild asks apprentices to solve a magical puzzle. The magicians will take a grid of **n** by **n** and place in each grid a number of dove eggs.

- The eggs will be placed such that each spot on the grid will contain between 1 and **n** eggs inclusive.
- No two spots in the same row or the same column will contain the same number of eggs. In other words each row and column will contain each number of dove eggs (between 1 and **n**) exactly once.

A hat is place over each spot in the grid, hiding all the dove eggs. On each hat three numbers glow. One represents the number of eggs under the hat, the other two numbers are meant to misdirect those that are non-logical thinkers. In order for a magician to be accepted into the guild he must determine how many eggs are under each hat. Below is a possible hat setup.



Matt is testing his ability to solve the problem and has asked for a program to confirm his theories regarding different hat setups.

Problem

Given a grid representing a hat setup determine and print the unique arrangement of eggs that could lead to said hat configuration.

Input Specification

The first line of input contains a single positive integer, n ($4 \le n \le 16$), which represents the length of the square grid. The next line contains a list of $3n$ space separated, positive integers between 1 and n inclusive. In the j-th row each triple of integers, at indices $3i$-2, $3i$-1, and $3i$, for i in the range of 1 to n, represent the numbers on i-th hat of the j-th row.

Output Specification

Output a grid of n by n space separated, positive integers between 1 and n inclusive, which represents the correct number of dove eggs under each hat in the order the hats occurred in the input.

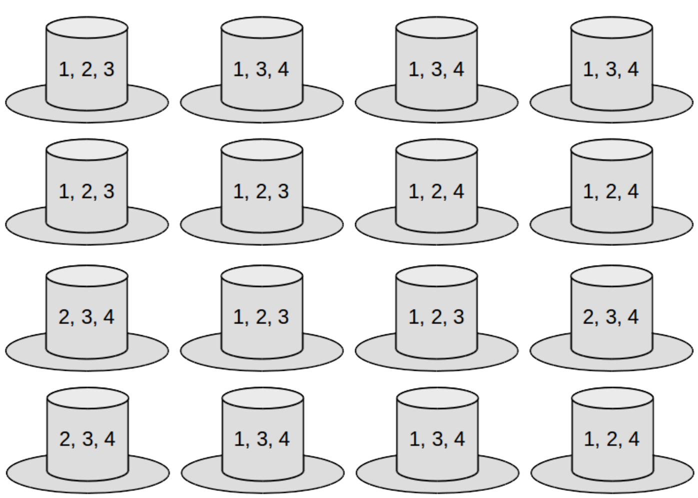
It is guaranteed that there exists **exactly** one solution.

Sample Input	Sample Output
4 1 2 3 1 3 4 1 3 4 1 3 4 1 2 3 1 2 3 1 2 4 1 2 4 2 3 4 1 2 3 1 2 3 2 3 4 2 3 4 1 3 4 1 3 4 1 2 4	2 4 3 1 1 3 2 4 4 2 1 3 3 1 4 2
5 3 4 2 5 4 1 1 5 3 5 2 3 1 2 4 5 4 3 4 1 5 2 1 4 3 5 2 1 2 3 5 1 3 3 4 2 1 5 3 4 2 1 5 4 2 2 5 1 2 1 3 4 5 2 3 1 4 3 5 4 1 2 4 5 2 3 2 3 4 5 1 4 3 1 5	2 5 1 3 4 5 1 4 2 3 3 4 5 1 2 1 3 2 4 5 4 2 3 5 1
9 8 5 6 7 5 6 7 5 6 9 4 8 3 7 2 2 3 9 3 1 9 5 7 8 2 6 9 7 2 9 8 1 2 3 7 2 1 4 9 7 4 6 5 3 1 3 2 9 3 4 8 1 6 7 6 7 8 7 2 3 2 4 7 7 6 4 7 1 4 1 8 9 8 1 3 8 3 5 5 9 4 1 3 6 3 5 2 7 1 6 3 4 5 9 6 8 6 8 7 4 8 9 5 2 8 4 7 6 9 8 3 1 4 8 4 8 1 8 9 7 9 6 5 4 2 7 2 8 3 6 2 9 9 6 2 4 1 5 5 2 6 8 9 4 9 8 3 1 8 2 7 6 1 3 7 5 1 3 4 5 3 7 8 1 5 1 6 3 9 5 7 3 9 2 6 9 3 1 6 3 4 1 5 8 3 6 8 1 7 9 2 3 2 6 7 8 2 7 1 9 3 3 4 7 1 3 2 4 6 1 2 1 3 1 5 4 7 6 9 9 6 4 3 2 8 3 2 9 7 4 5 6 1 4 6 5 2 1 3 8 5 3 1	8 6 5 4 3 9 1 7 2 9 8 3 1 6 5 2 4 7 7 2 4 6 1 8 3 5 9 1 3 6 5 8 7 9 2 4 3 4 1 7 5 2 8 9 6 4 5 9 8 2 6 7 1 3 5 1 7 2 9 3 4 6 8 2 7 8 9 4 1 6 3 5 6 9 2 3 7 4 5 8 1

Explanation

Case 1

```
3
1 2 3 1 3 4 1 3 4 1 3 4     (1, 2, or 3) (1, 3, or 4) (1, 3, or 4) (1, 3, or 4)
1 2 3 1 2 3 1 2 4 1 2 4  => (1, 2, or 3) (1, 2, or 3) (1, 2, or 4) (1, 2, or 4)
2 3 4 1 2 3 1 2 3 2 3 4     (2, 3, or 4) (1, 2, or 3) (1, 2, or 3) (2, 3, or 4)
2 3 4 1 3 4 1 3 4 1 2 4     (2, 3, or 4) (1, 3, or 4) (1, 3, or 4) (1, 2, or 4)
```



We can see that in order for the first row to have a 2 egg hat, the top left hat must have exactly 2 eggs.

```
2???
????
????
????
```

This forces the 2nd hat on the left column to be 1 egg since none of the other hats in that column can have 1 egg.

```
2???
1???
????
????
```

This forces the (2, 2) egg to be a 3 since no other hat in that row has a 3 left.

```
2???
13??
????
????
```

The 3rd row 2nd column hat must contain 2 eggs.

```
2???
13??
?2??
????
```

The 3rd row 3rd column must contain 1 egg

```
2???
13??
?21?
????
```

The next part is tricky. We know that the 1st hat in the last row must be a 3 or 4, and we know that the 3rd hat in the last row must be a 3 or 4. This is because both of the columns already contain a 1. This means that the other two hats in the row must have a 1 or 2. Using this information we know that the second hat in the last row is a 1, and the last had in the last row is a 2.

```
2???
13??
?21?
?1?2
```

This forces the top right hat to be a 1, and the 3rd hat in the 2nd row to be a 2.

```
2??1
132?
?21?
?1?2
```

Due to row completion we know that the 2nd row's empty spot is a 4 and the 2nd column's empty spot is also a 4

```
24?1
1324
?21?
?1?2
```

Row completion in the first row gives us a 3.

```
2431
1324
?21?
?1?2
```

Column completion in the 3rd column gives a 4 and 4th column gives us a 3.

```
2431
1324
?213
?142
```

The last two spots are forced by row completion.

```
2431
1324
4213
3142
```

The last two spots are forced by row completion.

<u>Solution Thoughts</u>

Use the sudoku base code on webcourses.

Grading Details

Read/Write from/to standard input/output – 10 points

Good comments, whitespace, and variable names – 15 points

No extra input output (e.g. input prompts, “Please enter the number of words”) – 10 points

write a validity check function – 15 points

Your program will be tested on 10 test cases – 5 points each

No points will be awarded to programs that do not compile using gcc -std=gnu11 (gnu “eleven”).

For this problem there is no limitation on what can be used.

Any case that causes your program to return a non-zero error return code will be treated as completely wrong. Additionally any case that takes longer than the maximum allowed time (the max of {5 times my solution, 5 seconds}) will also be treated as wrong.

For this problem you must use backtracking. Without backtracking you will earn at most 50 points.

No partial credit will be awarded for an incorrect case.

Follow the output format as specified for full points.