A generalization of the eight queen puzzle is the n queen puzzle in which the number n can be an arbitrary value. There are two common representations for the n queen puzzle: matrix and permutation. In the matrix representation, 0 as empty and 1 as queen for each cell. As for the representation of permutation, the index is the x coordinate and the value of the element is the y coordinate for the eight queens. For example, a four-queen matrix can be:

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

and its corresponding permutation is:

```
0 2 3 1
```

If any two queens will not reside in the same row and column, then an attack only occurs when the difference of x coordinate equals to the difference of y coordinate. Thus, in the above example we have one attack.
A possible improvement with respect to the number of attacks can be found through systematically searching for all possible swaps for two queens. Such swap can be made on the permutation by simply swap the two values with different indices. Suppose the two indices are i and j, then we can go through all possible pairs of i and j by looping i and then looping j in increasing order.
Your goal is to write a program to read the matrix form of an n queen puzzle, show the original attacks, and make one improvement for the game. You have to define two classes: one to store the n queen puzzle, and one to solve the n queen puzzle; that is, the solver contains an object of n queen puzzle as its private data member.
The class to store n queen puzzle should overload the stream extraction operator to input the matrix to the class object, overload the stream insertion operator to output the permutation and number of attacks, and provide a method to calculate the number of attacks. Also provide a default constructor, copy constructor, assignment operator, and destructor for the n queen puzzle class. Remember to use dynamic memory allocation in the class according to the input of number of queens.
The solver class should provide a constructor with one argument as an object of an n queen puzzle, overload the stream insertion operator to output the permutation and number of attacks, and provide a method to improve current n queen puzzle. (The constructor needs to initialize the class data member of n queen puzzle object by the overloaded copy constructor.)
Requirement: Provide classes satisfying the above requirements. Prepare your classes with appropriate constructors and overloaded operators and encapsulate the methods. Separate your program in files of three kinds: the class header file (.h), the class source code file (.cpp), and the file containing main function (.cpp).
Prohibited: Use C-style input/output.

Input

Each case contains an integer n, representing the number of queens, and an n-by-n matrix. The input ends with -1.

Output

For each case, use the matrix to construct the class object, and output the corresponding permutation and number of attacks. The permutation numbers are separated by space and end with a colon, followed by a space, the number of attacks, and an end line stream manipulator. Then make an improvement as mentioned above, and output the corresponding permutation and number of attacks again. Separate each case with an end line stream manipulator.

Sample Input

```
9
0 0 0 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 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 1 0
1 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0
0 0 1 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0
-1
```

Sample Output

```
8 4 6 3 7 0 5 2 1: 6 Attacks
8 4 6 3 0 7 5 2 1: 2 Attacks
```