Carnival

Each of Peter's N friends (numbered from 1 to N) bought exactly one carnival costume in order to wear it at this year's carnival parties. There are C different kinds of costumes, numbered from 1 to C. Some of Peter's friends, however, might have bought the same kind of costume. Peter would like to know which of his friends bought the same costume. For this purpose, he organizes some parties, to each of which he invites some of his friends. Peter knows that on the morning after each party he will not be able to recall which costumes he will have seen the night before, but only how many different kinds of costumes he will have seen at the party. Peter wonders if he can nevertheless choose the guests of each party such that he will know in the end, which of his friends had the same kind of costume. Help Peter!

Interaction

Your program must interact with the grader via standard input and output.

First, your program must read a line that contains the number N of friends.

Then, your program should interact with the grader as follows: To specify a party, it outputs a single line; this line contains the number k of people to invite ($1 \leq k \leq N$), followed by a list of the numbers of people to invite to this party (see example). Don't forget to flush the output (for example, using `fflush(stdout);` or `cout << endl;`)! Afterwards, your program must read the answer: one line with the number of different kinds of costumes that were worn by his friends at this party.

As soon as your program has determined which friend bought which costume, it should output this result in one line. This line starts with the number 0, followed by N space separated integers $c_1, \ldots, c_N$ ($1 \leq c_i \leq C$ for all i): c_i specifies the kind of costume that friend number i has bought. The numbering of the kinds of costumes does not matter; it is only necessary that identical costumes have identical numbers and that different costumes have different numbers and that the costumes all have numbers between 1 and C.

Immediately afterwards, your program must terminate (with `return 0;`).

Constraints and grading

We always have $2 \leq N \leq 150$.

If your program needs to specify P parties to determine the costume assignment for a test case, it will score

- 0 points if $11\,500 < P$,
- 20% of all points if $3\,500 < P \leq 11\,500$ and
- all points if $P \leq 3\,500$.

Sample

```
grader:  5
program:  5 1 2 3 4 5
grader:  3
program:  2 2 5
grader:  1
program:  2 1 2
grader:  2
program:  1 4
grader:  1
program:  0 2 1 2 3 1
```

The first example is about 5 friends with costume numbers 1 2 1 3 2. In the example interaction, the lines beginning with *grader* describe the input read by a solution program. The lines beginning with *program* describe the output of the solution program. The parties specified in the first example do not suffice to safely determine the costume assignment, of course – the solution program was just lucky to guess correctly.

```
grader:  3
program:  3 1 2 3
grader:  2
program:  2 1 3
grader:  1
program:  0 1 2 1
```

The second example is about 3 friends with costume numbers 1 2 1, and it is sufficient to specify these two parties to determine the costume assignment.

Limits

Time limit: 1 s
Memory limit: 256 MB

Feedback

There is full feedback given for this task, i.e. the public score shown equals your real score and you are shown the verdicts for all the testcases.