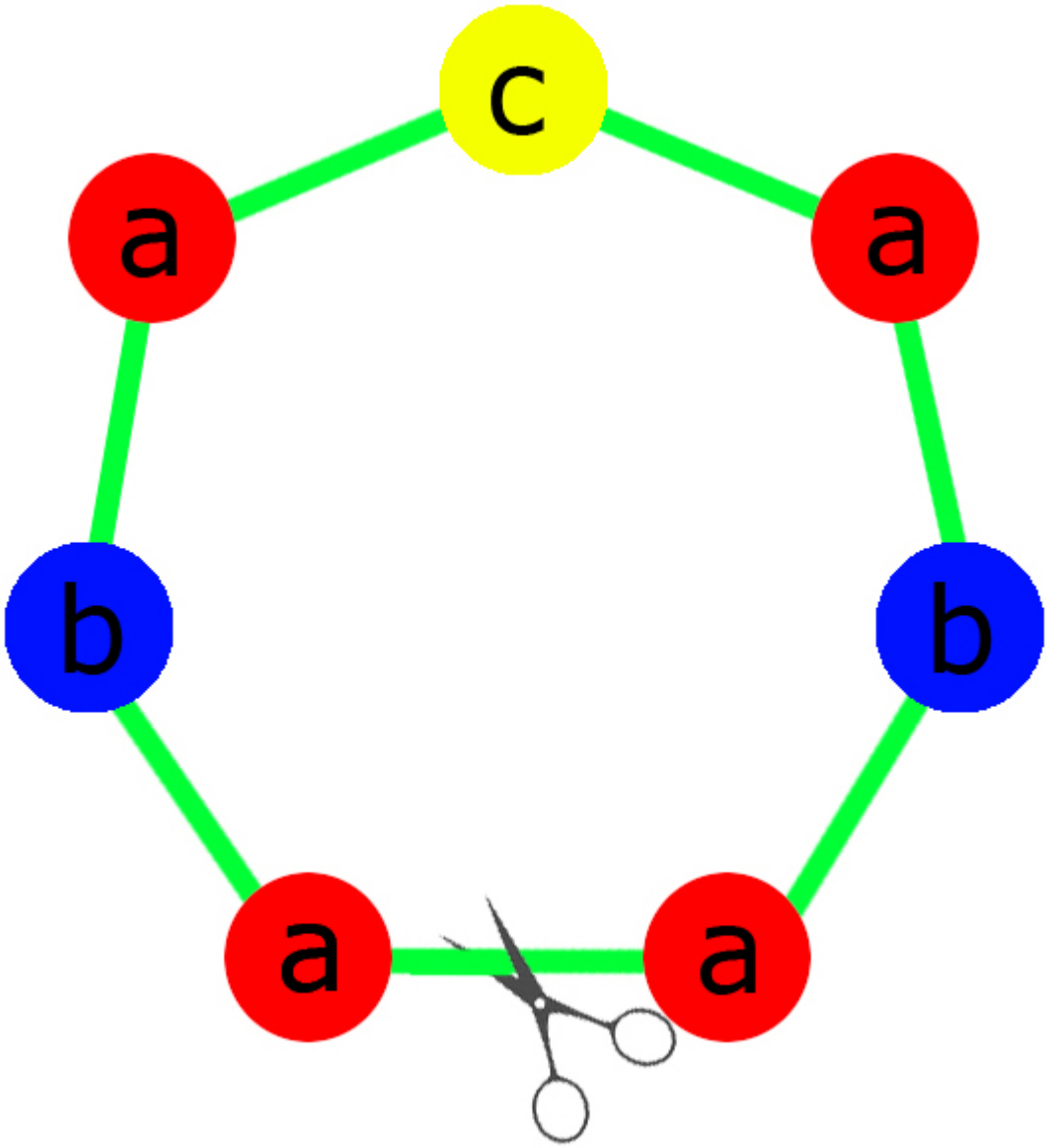
C. Necklace

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Ivan wants to make a necklace as a present to his beloved girl. A *necklace* is a cyclic sequence of beads of different colors. Ivan says that necklace is *beautiful* relative to the cut point between two adjacent beads, if the chain of beads remaining after this cut is a palindrome (reads the same forward and backward).



Ivan has beads of n colors. He wants to make a necklace, such that it's beautiful relative to as many cuts as possible. He certainly wants to use all the beads. Help him to make the most beautiful necklace.

Input

The first line of the input contains a single number n ($1 \le n \le 26$) — the number of colors of beads. The second line contains after n positive integers a_i — the quantity of beads of i-th color. It is guaranteed that the sum of a_i is at least 2 and does not exceed $100\,000$.

Output

In the first line print a single number — the maximum number of beautiful cuts that a necklace composed from given beads may have. In the second line print any example of such necklace.

Each color of the beads should be represented by the corresponding lowercase English letter (starting with a). As the necklace is cyclic, print it starting from any point.

Examples

input
3 4 2 1
output
1 abacaba

input
1 4
output
4 aaaa

input
2 1 1
output
0 ab

Note

In the first sample a necklace can have at most one beautiful cut. The example of such a necklace is shown on the picture.

In the second sample there is only one way to compose a necklace.

E. Hidden Bipartite Graph

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Bob has a simple undirected connected graph (without self-loops and multiple edges). He wants to learn whether his graph is bipartite (that is, you can paint all vertices of the graph into two colors so that there is no edge connecting two vertices of the same color) or not. As he is not very good at programming, he asked Alice for help. He does not want to disclose his graph to Alice, but he agreed that Alice can ask him some questions about the graph.

The only question that Alice can ask is the following: she sends s — a subset of vertices of the original graph. Bob answers with the number of edges that have both endpoints in s. Since he doesn't want Alice to learn too much about the graph, he allows her to ask no more than 20000 questions. Furthermore, he suspects that Alice might introduce false messages to their communication channel, so when Alice finally tells him whether the graph is bipartite or not, she also needs to provide a proof — either the partitions themselves or a cycle of odd length.

Your task is to help Alice to construct the queries, find whether the graph is bipartite.

Input

The first line contains a single integer n ($1 \le n \le 600$) — the number of vertices in Bob's graph.

Interaction

First, read an integer n ($1 \le n \le 600$) — the number of vertices in Bob's graph.

To make a query, print two lines. First of which should be in the format "? k" ($1 \le k \le n$), where k is the size of the set to be queried. The second line should contain k space separated distinct integers $s_1, s_2, \ldots, s_k$ ($1 \le s_i \le n$) — the vertices of the queried set.

After each query read a single integer m ($0 \le m \le \frac{n(n-1)}{2}$) — the number of edges between the vertices of the set $\{s_i\}$.

You are not allowed to ask more than 20000 queries.

If $m = -1$, it means that you asked more queries than allowed, or asked an invalid query. Your program should immediately terminate (for example, by calling exit(0)). You will receive `Wrong Answer`; it means that you asked more queries than allowed, or asked an invalid query. If you ignore this, you can get other verdicts since your program will continue to read from a closed stream.

After printing a query do not forget to print end of line and flush the output. Otherwise, you will get `Idleness limit exceeded`. To do this, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `flush(output)` in Pascal;
- `stdout.flush()` in Python;
- see documentation for other languages.

When you know the answer, you need to print it.

The format of the answer depends on whether the graph is bipartite or not.

If the graph is bipartite, print two lines. The first should contain the letter "Y" (short for "YES") followed by a space, and then a single integer s ($0 \le s \le n$) — the number of vertices in one of the partitions. Second line should contain s

integers $a_1, a_2, \ldots, a_s$ — vertices belonging to the first partition. All a_i must be distinct, and all edges in the main graph must have exactly one endpoint in the set $\{a_i\}$.

If the graph is not bipartite, print two lines. The first should contain the letter "N" (short for "NO") followed by a space, and then a single integer l ($3 \le l \le n$) — the length of one simple cycle of odd length. Second line should contain l integers $c_1, c_2, \ldots, c_l$ — the vertices along the cycle. It must hold that for all $1 \le i \le l$, there is an edge $\{c_i, c_{(i \bmod l)+1}\}$ in the main graph, and all c_i are distinct.

If there are multiple possible answers, you may print any of them.

Hacks format For hacks, use the following format:

The first line contains two integers n and m ($1 \le n \le 600$, $0 \le m \le \frac{n(n-1)}{2}$) — the number of vertices and edges of the graph, respectively.

Each of the next m lines contains two integers u_i and v_i ($1 \le u_i, v_i \le n$) mean that there is an edge between u_i and v_i. There must not be any multiple edges, no loops, and the graph must be connected.

For example, you need to use this test to get the first sample:

```
4 4
4 1
1 3
3 2
2 4
```

Examples

input
4 4 0 1 1 1 0

output
? 4 1 2 3 4 ? 2 1 2 ? 2 1 3 ? 2 1 4 ? 2 2 4 ? 2 3 4 Y 2 1 2

input
4 4 3

output

```
? 4
1 4 2 3
? 3
1 2 4
N 3
2 1 4
```

Note

In the first case, Alice learns that there are 4 edges in the whole graph. Over the course of the next three queries, she learns that vertex 1 has two neighbors: 3 and 4. She then learns that while vertex 2 is adjacent to 4, the vertex 3 isn't adjacent to 4. There is only one option for the remaining edge, and that is (2, 3). This means that the graph is a cycle on four vertices, with (1, 2) being one partition and (3, 4) being the second. Here, it would be also valid to output "3 4" on the second line.

In the second case, we also have a graph on four vertices and four edges. In the second query, Alice learns that there are three edges among vertices (1, 2, 4). The only way this could possibly happen is that those form a triangle. As the triangle is not bipartite, Alice can report it as a proof. Notice that she does not learn where the fourth edge is, but she is able to answer Bob correctly anyway.