Problem L. Leo

In a future where digital logic has evolved beyond binary, a new system of signal processing, known as tri-color logic, has become the standard. This system defines four fundamental states in two categories:

- **Colored**: There are three colored states, denoted by R (Red), G (Green), and B (Blue), respectively. Each serves as a distinguishable "on" state, analogous to the traditional logic 1 signal.
- **Colorless**: The only colorless state is denoted by $*$ (Transparent). It serves as an "off" state, analogous to the traditional logic 0 signal.

Sulfox the fennec fox is designing a computing device based on tri-color logic technology, named Leo. It employs a simple combinational logic architecture, so we can regard it as a directed acyclic graph in which each node holds a state from the set $\{\mathrm{R}, \mathrm{G}, \mathrm{B}, *\}$. After fixing the machine's input scale to n, the graph is constructed as follows:

- Nodes $1, 2, \ldots, n$ are *input nodes*, whose states should be given as the input to the machine. Input nodes have no predecessors.
- Nodes $n+1, n+2, \ldots, n+m$ are *internal nodes*, whose states are computed based on the states of two predecessors, where the two predecessors of each internal node may be the same and are chosen from nodes with indices strictly less than its own. To ensure the machine's operating speed, m (the number of internal nodes) **must not exceed** $6n$.
- The internal node $n+m$ is also the only *output node*, whose state is the output of the machine.

Every internal node is of exactly one of the following two types, depending on how it combines the signals from its two predecessors:

- **Tri-Color AND**: If the states of its predecessors are identical, its own state will be that same state. If they differ, its own state will be $*$.
- **Tri-Color OR**: If the states of its predecessors are identical, its own state will be that same state. If one predecessor's state is $*$ while the other's is colored, its own state will be the colored one. If they differ and both are colored, its own state will be the colored state that is different from both.

With everything prepared, we now specify Leo's intended functionality. The states given to the n input nodes will be guaranteed to contain at least one occurrence of each of R, G, and B, where $*$ may appear any number (possibly zero) of times. For every such input, the state of the output node $n+m$ must be identical to **the third distinct colored state encountered** when examining the input nodes from node 1 to node n (ignoring all $*$). In other words, among R, G, and B it must output the colored state whose first occurrence has the largest index.

Given the machine's fixed input scale n, please complete the design of the internal nodes, i.e., specify the number of internal nodes and the type and predecessors of each internal node.

To verify the correctness of your design, for each test, the judge will generate $\left\lfloor \frac{10^7}{n} \right\rfloor$ valid input cases, each of which will be provided as your machine's input in turn. Your solution passes a test if your machine's output always meets Leo's intended functionality across all input cases.

Input

The only line contains an integer n ($3 \le n \le 10^5$), denoting the number of input nodes.

Output

In the first line, output an integer m ($1 \le m \le 6n$), denoting the number of internal nodes in your design.

Then output m lines, the i-th line containing a character t_i ($t_i \in \{$ '&' , '|' $\}$) and two integers u_i and v_i ($1 \le u_i, v_i < n + i$), denoting the type of the internal node $n + i$ and the indices of its two predecessors, where '&' represents the tri-color AND type and '|' represents the tri-color OR type.

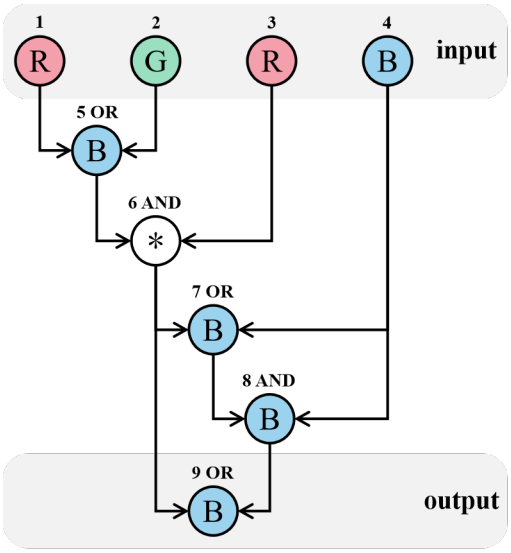
If there are multiple designs, you may output any of them. Note that you do not have to minimize m.

Example

standard input	standard output
4	5 | 1 2 & 3 5 | 4 6 & 4 7 | 6 8

Note

It can be proven that the design in the sample case can always find the third encountered distinct colored state for every valid input of $n = 4$. For example, the figure below illustrates the machine's computation for the input "RGRB":



The figure below illustrates the machine's computation for the input "BRG∗":

