Problem G: The Joy of Cats

The book *Abstract and Concrete Cats: The Joy of cats* by Adámek, Herrlich, Strecker is full of definitions and interesting concepts, for example **Cat** or **CAT** on page 40.

In order not to lose track of all the definitions, the authors want to assemble an index using a *trie*. A trie is a tree data structure that can be used to store a set of strings. In the tree each node corresponds to a prefix of one of the strings, with the root node being the empty string. The edges go from shorter to longer prefixes so that for every node the labels on the path from the root spell out the corresponding prefix. An example can be seen below:

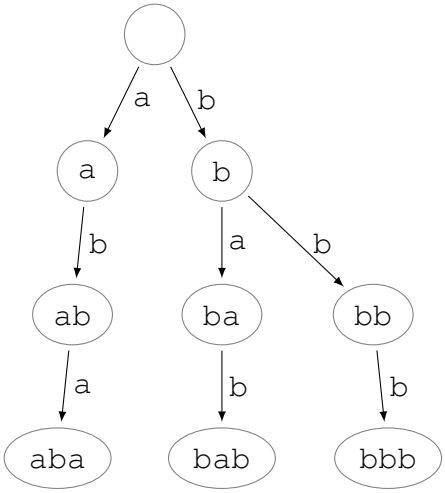

Figure G.1: An example trie built on the strings `aba`, `bab` and `bbb`, with 9 nodes.

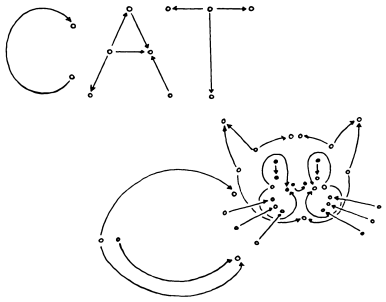

Figure G.2: An example illustration from *The Joy of Cats* (page 12)

The authors have coded a trie in their favourite programming language, but they are unsure whether their implementation is efficient enough. To have a good stress test for their code they are interested in test data that will cause as many trie nodes as possible to be allocated. They have asked you to come up with such test data.

More specifically, they know that the book defines up to n words, where each definition is not longer than m letters. Since most of the defined words are quite similar (like **cat** or **ccc**), your test data should only use the first k lowercase letters of the English alphabet. Given these constraints, find a set of strings that generates a trie of maximal size.

Input

The input consists of one line with three integers:

- n ($1 \leq n \leq 10\,000$), the number of words;
- m ($1 \leq m \leq 20$), the maximum length of each word;
- k ($1 \leq k \leq 26$), the size of the alphabet.

Output

On the first line, output the maximum number of nodes in a trie, followed by the number w ($0 \leq w \leq n$) of strings used in your test data. The following w lines should contain your test data, one string per line. The strings must be non-empty.

Sample Input 1

```
3 3 2
```

Sample Output 1

```
9 3
aba
bab
bbb
```

Sample Input 2

```
5 1 3
```

Sample Output 2

```
4 4
b
a
c
a
```