Professor X would like to give a term assignment to his students. This would be a group task, and he wants to form the groups himself. In his class of N students, he wants each group to have exactly K students. Here at most one group can have less than K students when it would be impossible for all the groups to have exactly K students.

The good thing about X's students is that they do not have any preference or any objection about the groups that the professor would form. But that is actually the problem for the professor: he has got too many options open for him. Your task would be to aid the professor in two types of query:

1. `COUNT` N K: Count the number of ways the groups can be formed.
2. `GENERATE` N K: Generate the complete list of ways the groups can be formed.

Let A be a possible solution, then A would be a set of $Ceiling(n/k)$ groups. One student can be a member of exactly one group in A. And you should also notice that two different placement orders of the same students in a group are considered to be the same. In other words, the task is about generating the set of all possible A's or to count how many A's are possible.

Input

There can be at most 1200 test cases. Each test case starts with a string '`GENERATE`', or '`COUNT`' followed by two integers N and K. For the '`GENERATE`' queries you can assume $1 \leq N, K \leq 15$ and for the '`COUNT`' queries, $1 \leq N, K \leq 30$. The end of the test cases is denoted with a '`END`' string as shown in the sample output.

Output

For each of the '`COUNT`' queries, output an integer in one line giving the total number of solutions. You can assume that the output will always fit in a 64bit unsigned integer. For each of the '`GENERATE`' queries, output the total number of solutions, followed by that many number of lines. In each of those lines you are to print the groups. The professor has assigned the first N letters (uppercase) of the english alphabet to his N students. In each group the students are to be listed in alphabetic order and all the groups in a solution are to be sorted in ascending order. You have to use a single space to *separate* two groups in a solution. The solutions themselves have to be sorted in lexicographic order (to be more specific, you can use the ASCII values to do the comparison). Because of the sorting criterion the output will always be unique. You can assume that the number of solutions printed for each of the '`GENERATE`' queries will not be more than 10000 lines and collectively the solutions printed for all the '`GENERATE`' queries will not be more than 30000 lines.

Sample Input

```
COUNT 4 2
GENERATE 4 3
END
```

Sample Output

```
3
4
A BCD
ABC D
ABD C
ACD B
```