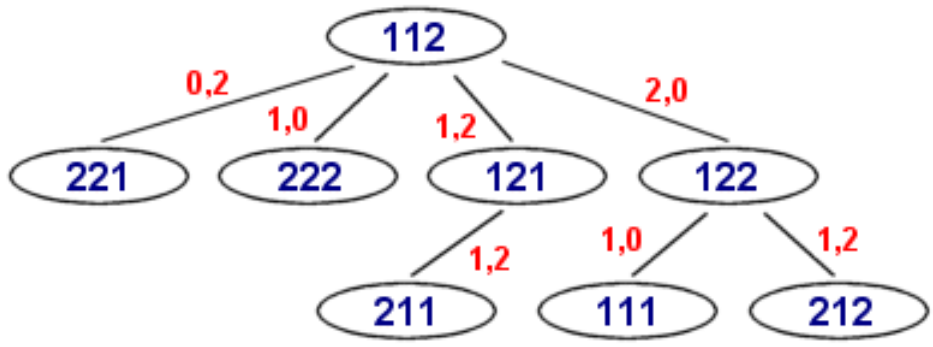
Mastermind is a classic two-player game between a codemaker and a codebreaker. The codemaker secretly selects a n-digit code, where each digit is between 1 and m. The codebreaker then tries to guess the code by repeatedly guessing a valid code (n digits, each digit between 1 and m). After each guess, the codemaker tells the codebreaker two numbers, (b, w): the number of correct digits in the correct position (b) and the number of digits that are part in the code but not in the correct position (w). For example, if the code is 1233 and the codebreaker's guess is 3243, then the codemaker's response would be $b = 2$, $w = 1$, since the codebreaker has guessed the '2' and the second '3' correctly and in the correct positions, while having guessed the first '3' correctly but in the wrong position.

A *deterministic* Mastermind strategy is when you, as codebreaker, make the same first guess every time you play, and depending on the answer, always make the same second guess, etc. Such a strategy can be depicted as a selection tree:



In this strategy, the codebreaker starts with the guess 112. If the reply from the codemaker is 2,0, the codebreaker tries 122. If the reply is 1,0, the codebreaker knows the answer is 111, which will be his third and final guess. Note that in the picture above, all 8 states should have an extra edge (3,0) in case this is the final state.

An *optimal* Mastermind strategy is a deterministic strategy that minimizes the number of guesses, on average, assuming all codes have the same probability of being selected by the codemaker. The picture above shows the selection tree for one optimal strategy with $n = 2$ and $m = 3$.

Write a program that, given n and m, calculates an optimal Mastermind strategy. The program should output the total number of guesses required for this strategy if the game is played m^n times (each time with a distinct code; that is, all different codes are used), as well as the whole selection tree. In the strategy in the picture above, the total number of guesses is $1 + 2 + 2 + 2 + 3 + 2 + 3 + 3 = 18$. The program should also be able to handle a fixed first guess. That is, finding the optimal strategy with the constraint that the first guess is given.

Input

Each line in the input contains three integers: n ($1 \leq n \leq 4$), the length of the code, m ($1 \leq m \leq 6$), the highest valid digit in a code (the lowest valid digit will always be 1) and k, the fixed first guess. If $k = 0$, the first guess isn't fixed. Otherwise k will be a valid guess, containing exactly n digits, all of them between 1 and m. Also, $n * m \leq 12$ for all input cases. The input is terminated with end of file. There will be at most 30 sets of input.

Output

The first line in the output for each input set should contain an integer, the total number of guesses required with the optimal strategy. Then follows m^n lines, each line corresponding to the sequence of guesses for each possible code. The guesses must form a legal selection tree as stated above. The format of each line should be:

guess:*b*,*w* *guess*:*b*,*w* ... *guess*:*b*,*w*

Where *guess* is a code and b and w have the meaning mentioned earlier. The last guess in each row should be the correct code. The first guess in each line should be k, if k is not equal to 0. No special ordering of these rows are necessary; any one will do as long as all possible codes are covered. Since there exist many different strategies yielding the same total number of guesses, any such strategy will be accepted. Output a blank line after each set of input.

Note: The first case in the sample output corresponds to the strategy in the picture. This line is not a part of output

Sample Input

```
3 2 0
3 2 222
1 4 3
```

Sample Output

```
18
112:0,2 221:3,0
112:1,0 222:3,0
112:1,2 121:1,2 211:3,0
112:1,2 121:3,0
112:2,0 122:1,0 111:3,0
112:2,0 122:1,2 212:3,0
112:2,0 122:3,0
112:3,0

21
222:0,0 111:3,0
222:1,0 112:1,2 121:1,2 211:3,0
222:1,0 112:1,2 121:3,0
222:1,0 112:3,0
222:2,0 122:1,2 212:1,2 221:3,0
222:2,0 122:1,2 212:3,0
222:2,0 122:3,0
222:3,0

10
3:0,0 1:0,0 2:0,0 4:1,0
3:0,0 1:0,0 2:1,0
3:0,0 1:1,0
3:1,0
```