Task Popcount

***Mi**niature **Al**gebraic **Na**tural **R**elay* (also called `MALNAR`) is the latest technological advancement in the flourishing realm of small programmable devices. You can write your own programs for this device using *MalnarScript*, an esoteric programming language with the following set of features:

- Input to the program is a single non-negative integer strictly less than 2^N.
- Output of the program is a single non-negative integer strictly less than 2^N.
- When programming in *MalnarScript* you can only use one N-bit unsigned integer variable A. At the beginning of the program, this variable holds the input and its value at the end of the program is considered to be the program's output.
- The source code of *MalnarScript* must consist of at most K commands of the form `A=<expr>` which are executed in order and each of them must consist of **at most a thousand** characters. The symbol `<expr>` is defined recursively as follows:

```
<expr> = A | <num> | (<expr><operator><expr>)
```

In other words, symbol `<expr>` can either be a variable A, or it can conform to the definition of symbol `<num>`, or it can (inside parentheses) represent a two-membered expression in which each operand conforms to the same `<expr>` definition.

The symbol `<num>` in the definition above represents a non-negative decimal integer strictly less than 2^N, while the symbol `<operator>` can either be `+`, `-`, `|`, `&`, `<<` or `>>` which (in order) represent the operations of addition, subtraction, bitwise or, bitwise and, left shift and right shift.

Also the character `A` can appear `at most 5 times` in the `<expr>` symbol.

In the case of overflow or underflow when performing the operations of addition and subtraction, *MalnarScript* will perform those operations modulo 2^N. For example, when $N = 3$ the expression $(7 + 3)$ will evaluate to 2 and the expression $(2 - 5)$ will evaluate to 5 in `MalnarScript`.

The right side of the equation in each command evaluates into a single number which will then be stored into A. In order to evaluate the right hand side expression, *MalnarScript* first replaces each occurrence of A with its current value. The calculation of expression then proceeds as it would in any mathematical expression, i.e., the parentheses take precedence. Note that the priorities of operators (in term of operation order) are irrelevant because the final result is completely defined by placement of parentheses.

Your task is to write a program which outputs a program in *MalnarScript* which calculates the number of ones in a binary representation of the input value.

Input

The first line contains two integers N and K from the task description.

Output

In the first line you should output the number of commands of the produced *MalnarScript* program.

In the remaining lines you should output the commands of the sought program. Each command must be printed in a separate line and must satisfy the syntax of `MalnarScript` as described in the task description.

It is important that there are no unnecessary empty lines or extra whitespace characters in the output. Each line (including the last) must be terminated by the end-of-line character (`'\n'`).

Scoring

Subtask	Score	Constraints
1	15	$2 \leq N \leq 100$, $K = N - 1$
2	15	$N = 500$, $K = 128$
3	35	$1 \leq N \leq 40$, $K = 7$
4	45	$100 \leq N \leq 500$, $K = 10$

Examples

input

```
2 2
```

output

```
1
A=(A-((A&2)>>1))
```

input

```
3 5
```

output

```
2
A=((A&4)|((A&3)-((A&2)>>1)))
A=((A&3)+((A&4)>>2))
```

Clarification of the first example:
$input = 0 \Rightarrow output = (0 - ((0\&2) >> 1)) = (0 - (0 >> 1)) = (0 - 0) = 0$
$input = 1 \Rightarrow output = (1 - ((1\&2) >> 1)) = (1 - (0 >> 1)) = (1 - 0) = 1$
$input = 2 \Rightarrow output = (2 - ((2\&2) >> 1)) = (2 - (2 >> 1)) = (2 - 1) = 1$
$input = 3 \Rightarrow output = (3 - ((3\&2) >> 1)) = (3 - (2 >> 1)) = (3 - 1) = 2$