GPC-UPC Second Day Contest (Hardforces)

A. Cave Painting

1 second, 256 megabytes

Imp is watching a documentary about cave painting.

Some numbers, carved in chaotic order, immediately attracted his attention. Imp rapidly proposed a guess that they are the remainders of division of a number n by all integers i from 1 to k. Unfortunately, there are too many integers to analyze for Imp.

Imp wants you to check whether all these remainders are distinct. Formally, he wants to check, if all $n \bmod i$, $1 \le i \le k$, are distinct, i. e. there is no such pair (i, j) that:

- $1 \le i < j \le k$,
- $n \bmod i = n \bmod j$, where $x \bmod y$ is the remainder of division x by y.

Input
The only line contains two integers n, k $(1 \le n, k \le 10^{18})$.

Output
Print "Yes", if all the remainders are distinct, and "No" otherwise.

You can print each letter in arbitrary case (lower or upper).

input
```
4 4
```
output
```
No
```

input
```
5 3
```
output
```
Yes
```

In the first sample remainders modulo 1 and 4 coincide.

B. New Year's Eve

1 second, 256 megabytes

Since Grisha behaved well last year, at New Year's Eve he was visited by Ded Moroz who brought an enormous bag of gifts with him! The bag contains n sweet candies from *the good ol' bakery*, each labeled from 1 to n corresponding to its tastiness. No two candies have the same tastiness.

The choice of candies has a direct effect on Grisha's happiness. One can assume that he should take the tastiest ones — but no, the holiday magic turns things upside down. It is the xor-sum of tastinesses that matters, not the ordinary sum!

A xor-sum of a sequence of integers $a_1, a_2, ..., a_m$ is defined as the bitwise XOR of all its elements: $a_1 \oplus a_2 \oplus \ldots \oplus a_m$, here $\oplus$ denotes the bitwise XOR operation; more about bitwise XOR can be found here.

Ded Moroz warned Grisha he has more houses to visit, so Grisha can take **no more than k** candies from the bag. Help Grisha determine the largest xor-sum (largest xor-sum means maximum happiness!) he can obtain.

Input
The sole string contains two integers n and k $(1 \le k \le n \le 10^{18})$.

Output
Output one number — the largest possible xor-sum.

input
```
4 3
```
output
```
7
```

input
```
6 6
```
output
```
7
```

In the first sample case, one optimal answer is 1, 2 and 4, giving the xor-sum of 7.

In the second sample case, one can, for example, take all six candies and obtain the xor-sum of 7.

C. Fractal

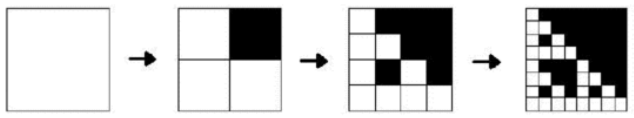
2 seconds, 64 megabytes

Ever since Kalevitch, a famous Berland abstractionist, heard of fractals, he made them the main topic of his canvases. Every morning the artist takes a piece of graph paper and starts with making a model of his future canvas. He takes a square as big as $n \times n$ squares and paints some of them black. Then he takes a clean square piece of paper and paints the fractal using the following algorithm:

Step 1. The paper is divided into n^2 identical squares and some of them are painted black according to the model.

Step 2. Every square that remains white is divided into n^2 smaller squares and some of them are painted black according to the model.

Every following step repeats step 2.



Unfortunately, this tiresome work demands too much time from the painting genius. Kalevitch has been dreaming of making the process automatic to move to making 3D or even 4D fractals.

Input
The first line contains integers n and k $(2 \le n \le 3, 1 \le k \le 5)$, where k is the amount of steps of the algorithm. Each of the following n lines contains n symbols that determine the model. Symbol «.» stands for a white square, whereas «*» stands for a black one. It is guaranteed that the model has at least one white square.

Output
Output a matrix $n^k \times n^k$ which is what a picture should look like after k steps of the algorithm.

input
```
2 3
.*
..
```

output

```
.*******
..******
.*.*****
....****
.***.***
..**..**
.*.*.*.*
........
```

input

```
3 2
.*.
***
.*.
```

output

```
.*.***.*.
*********
.*.***.*.
*********
*********
*********
.*.***.*.
*********
.*.***.*.
```

D. Robbers' watch

2 seconds, 256 megabytes

Robbers, who attacked the Gerda's cab, are very successful in covering from the kingdom police. To make the goal of catching them even harder, they use their own watches.

First, as they know that kingdom police is bad at math, robbers use the positional numeral system **with base** 7. Second, they divide one day in n hours, and each hour in m minutes. Personal watches of each robber are divided in two parts: first of them has the smallest possible number of places that is necessary to display any integer from 0 to $n - 1$, while the second has the smallest possible number of places that is necessary to display any integer from 0 to $m - 1$. Finally, if some value of hours or minutes can be displayed using less number of places in base 7 than this watches have, the required number of zeroes is added at the beginning of notation.

Note that to display number 0 section of the watches is required to have at least one place.

Little robber wants to know the number of moments of time (particular values of hours and minutes), such that all digits displayed on the watches are **distinct**. Help her calculate this number.

Input
The first line of the input contains two integers, given in the decimal notation, n and m $(1 \le n, m \le 10^9)$ — the number of hours in one day and the number of minutes in one hour, respectively.

Output
Print one integer in decimal notation — the number of different pairs of hour and minute, such that all digits displayed on the watches are distinct.

input

```
2 3
```

output

```
4
```

input

```
8 2
```

output

```
5
```

In the first sample, possible pairs are: $(0: 1)$, $(0: 2)$, $(1: 0)$, $(1: 2)$.

In the second sample, possible pairs are: $(02: 1)$, $(03: 1)$, $(04: 1)$, $(05: 1)$, $(06: 1)$.

E. Golden System

1 second, 256 megabytes

Piegirl got bored with binary, decimal and other integer based counting systems. Recently she discovered some interesting properties about number $q = \frac{\sqrt{5}+1}{2}$, in particular that $q^2 = q + 1$, and she thinks it would make a good base for her new unique system. She called it "golden system". In golden system the number is a non-empty string containing 0's and 1's as digits. The decimal value of expression $a_0 a_1 ... a_n$ equals to $\sum_{i=0}^{n} a_i \cdot q^{n-i}$.

Soon Piegirl found out that this system doesn't have same properties that integer base systems do and some operations can not be performed on it. She wasn't able to come up with a fast way of comparing two numbers. She is asking for your help.

Given two numbers written in golden system notation, determine which of them has larger decimal value.

Input
Input consists of two lines — one for each number. Each line contains non-empty string consisting of '0' and '1' characters. The length of each string does not exceed 100000.

Output
Print ">" if the first number is larger, "<" if it is smaller and "=" if they are equal.

input

```
1000
111
```

output

```
<
```

input

```
00100
11
```

output

```
=
```

input

```
110
101
```

output

```
>
```

In the first example first number equals to $((\sqrt{5}+1)/2)^3 \approx 1.618033988^3 \approx 4.236$, while second number is approximately $1.618033988^2 + 1.618033988 + 1 \approx 5.236$, which is clearly a bigger number.

In the second example numbers are equal. Each of them is ≈ 2.618.

F. New Year and Fireworks

2.5 seconds, 256 megabytes

One tradition of welcoming the New Year is launching fireworks into the sky. Usually a launched firework flies vertically upward for some period of time, then explodes, splitting into several parts flying in different directions. Sometimes those parts also explode after some period of time, splitting into even more parts, and so on.

Limak, who lives in an infinite grid, has a single firework. The behaviour of the firework is described with a recursion depth n and a duration for each level of recursion $t_1, t_2, ..., t_n$. Once Limak launches the firework in some cell, the firework starts moving upward. After covering t_1 cells (including the starting cell), it explodes and splits into two parts, each moving in the direction changed by 45 degrees (see the pictures below for clarification). So, one part moves in the top-left direction, while the other one moves in the top-right direction. Each part explodes again after covering t_2 cells, splitting into two parts moving in directions again changed by 45 degrees. The process continues till the n-th level of recursion, when all 2^{n-1} existing parts explode and disappear without creating new parts. After a few levels of recursion, it's possible that some parts will be at the same place and at the same time — it is allowed and such parts do not crash.

Before launching the firework, Limak must make sure that nobody stands in cells which will be visited at least once by the firework. Can you count the number of those cells?

Input

The first line of the input contains a single integer n ($1 \le n \le 30$) — the total depth of the recursion.

The second line contains n integers $t_1, t_2, ..., t_n$ ($1 \le t_i \le 5$). On the i-th level each of 2^{i-1} parts will cover t_i cells before exploding.

Output

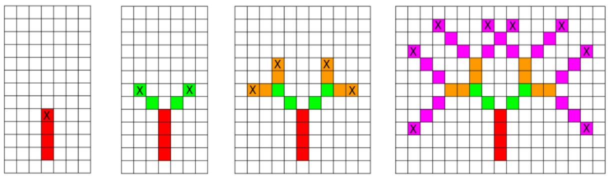
Print one integer, denoting the number of cells which will be visited at least once by any part of the firework.

input
```
4
4 2 2 3
```
output
```
39
```

input
```
6
1 1 1 1 1 3
```
output
```
85
```

input
```
1
3
```
output
```
3
```

For the first sample, the drawings below show the situation after each level of recursion. Limak launched the firework from the bottom-most red cell. It covered $t_1 = 4$ cells (marked red), exploded and divided into two parts (their further movement is marked green). All explosions are marked with an 'X' character. On the last drawing, there are 4 red, 4 green, 8 orange and 23 pink cells. So, the total number of visited cells is $4 + 4 + 8 + 23 = 39$.



For the second sample, the drawings below show the situation after levels 4, 5 and 6. The middle drawing shows directions of all parts that will move in the next level.