Root :: Programming Challenges (Skiena & Revilla) :: Chapter 6

Link:https://onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=34

Problemas:

Código em C/C+++

Discentes	**Problemas**
BRIHAN	10183 - How Many Fibs?
CARLOS	10213 How Many Pieces of Land?
DANIEL	10198 Counting
JOÃO GABRIEL	10157 Expressions
JOÃO HENRIQUE	10247 Complete Tree Labeling!
MATEUS	10254 The Priest Mathematician
PEDRO	10049 - Self-describing Sequence
VINICIUS	846 - Steps

UVa Online Judge

10183 How many Fibs?

Recall the definition of the Fibonacci numbers:

$$\begin{aligned} f_1 &:= 1 \\ f_2 &:= 2 \\ f_n &:= f_{n-1} + f_{n-2} \qquad (n \geq 3) \end{aligned}$$

Given two numbers a and b, calculate how many Fibonacci numbers are in the range $[a, b]$.

Input

The input contains several test cases. Each test case consists of two non-negative integer numbers a and b. Input is terminated by $a = b = 0$. Otherwise, $a \leq b \leq 10^{100}$. The numbers a and b are given with no superfluous leading zeros.

Output

For each test case output on a single line the number of Fibonacci numbers f_i with $a \leq f_i \leq b$.

Sample Input

```
10 100
1234567890 9876543210
0 0
```

Sample Output

```
5
4
```

UVa Online Judge

10213 How Many Pieces of Land?

You are given an elliptical shaped land and you are asked to choose n arbitrary points on its boundary. Then you connect all these points with one another with straight lines (that's $n * (n-1)/2$ connections for n points). What is the maximum number of pieces of land you will get by choosing the points on the boundary carefully?

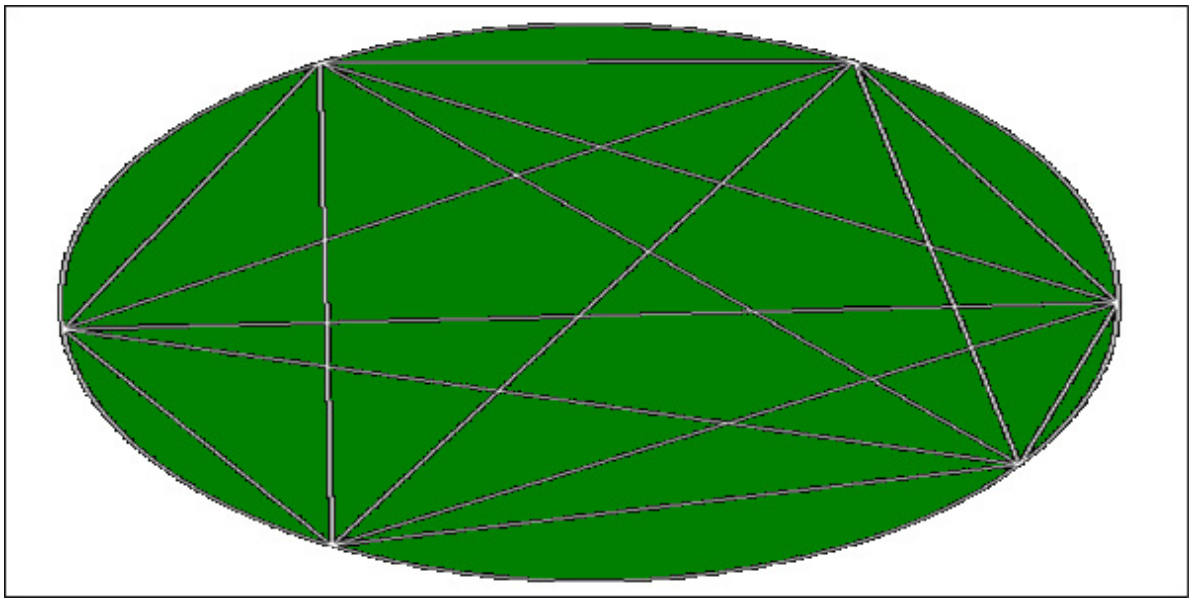

Fig: When the value of n is 6

Input

The first line of the input file contains one integer S $(0 < S < 3500)$, which indicates how many sets of input are there. The next S lines contain S sets of input. Each input contains one integer N $(0 \leq N < 2^{31})$.

Output

For each set of input you should output in a single line the maximum number pieces of land possible to get for the value of N.

Sample Input

```
4
1
2
3
4
```

Sample Output

```
1
2
4
```

```
8
```

UVa Online Judge

10198 Counting

Gustavo knows how to count, but he is now learning how write numbers. As he is a very good student, he already learned 1, 2, 3 and 4. But he didn't realize yet that 4 is different than 1, so he thinks that 4 is another way to write 1. Besides that, he is having fun with a little game he created himself: he make numbers (with those four digits) and sum their values. For instance:

`132 = 1 + 3 + 2 = 6`
`112314 = 1 + 1 + 2 + 3 + 1 + 1 = 9` (remember that Gustavo thinks that `4 = 1`)

After making a lot of numbers in this way, Gustavo now wants to know how much numbers he can create such that their sum is a number n. For instance, for $n = 2$ he noticed that he can make 5 numbers: 11, 14, 41, 44 and 2 (he knows how to count them up, but he doesn't know how to write five). However, he can't figure it out for n greater than 2. So, he asked you to help him.

Input

Input will consist on an arbitrary number of sets. Each set will consist on an integer n such that $1 \leq n \leq 1000$. You must read until you reach the end of file.

Output

For each number read, you must output another number (on a line alone) stating how much numbers Gustavo can make such that the sum of their digits is equal to the given number.

Sample Input

```
2
3
```

Sample Output

```
5
13
```

UVa Online Judge

10157 Expressions

Let X be the set of *correctly built parenthesis expressions*. The elements of X are strings consisting only of the characters '`(`' and '`)`'. The set X is defined as follows:

- an empty string belongs to X
- if A belongs to X, then `(`A`)` belongs to X
- if both A and B belong to X, then the concatenation AB belongs to X.

For example, the following strings are correctly built parenthesis expressions (and therefore belong to the set X):

```
()(())()
```

```
((()()))
```

The expressions below are not correctly built parenthesis expressions (and are thus not in X):

```
(()))(()
```

```
())(()
```

Let E be a correctly built parenthesis expression (therefore E is a string belonging to X).
The *length* of E is the number of single parenthesis (characters) in E.
The *depth* $D(E)$ of E is defined as follows:

$$D(E) = \begin{cases} 0 & \text{if } E \text{ is empty} \\ D(A)+1 & \text{if } E = \texttt{(}A\texttt{)}, \text{ and } A \text{ is in X} \\ \max(D(A), D(B)) & \text{if } E = AB, \text{ and } A, B \text{ are in X} \end{cases}$$

For example, the length of "`()(())()`" is 8, and its depth is 2. What is the number of correctly built parenthesis expressions of length n and depth d, for given positive integers n and d?

Write a program which

- reads two integers n and d
- computes the number of correctly built parenthesis expressions of length n and depth d;

Input

Input consists of lines of pairs of two integers - n and d, at most one pair on line, $2 \leq n \leq 300$, $1 \leq d \leq 150$.

The number of lines in the input file is at most 20, the input may contain empty lines, which you don't need to consider.

Output

For every pair of integers in the input write single integer on one line - the number of correctly built parenthesis expressions of length n and depth d.

Note: There are exactly three correctly built parenthesis expressions of length 6 and depth 2:

```
(())()
()(())
(()())
```

Sample Input

```
6 2
300 150
```

Sample Output

```
3
1
```

UVa Online Judge

10247 Complete Tree Labeling!

A complete k-ary tree is a k-ary tree in which all leaves have same depth and all internal nodes have degree k. This k is also known as the branching factor of a tree. It is very easy to determine the number of nodes of such a tree. Given the depth and branching factor of such a tree, you will have to determine in how many different ways you can number the nodes of the tree so that the label of each node is less that that of its descendants. You should assume that for numbering a tree with N nodes you have the $(1, 2, 3, N - 1, N)$ labels available.

Input

The input file will contain several lines of input. Each line will contain two integers k and d. Here k is the branching factor of the complete k-ary tree and d is the depth of the complete k-ary tree ($k > 0$, $d > 0$, $k * d \leq 21$).

Output

For each line of input, produce one line of output containing a round number, which is the number of ways the k-ary tree can be labeled, maintaining the constraints described above.

Sample Input

```
2 2
10 1
```

Sample Output

```
80
3628800
```

UVa Online Judge

10254 The Priest Mathematician

The ancient folklore behind the *"Towers of Hanoi"* puzzle invented by E. Lucas in 1883 is quite well known to us. One more recent legend tells us that the Brahmin monks from Benares never believed that the world could vanish at the moment they finished to transfer the 64 discs from the needle on which they were to one of the other needles, and they decided to finish the task as soon as possible.

Fig: The Four Needle (Peg) Tower of Hanoi

One of the priests at the Benares temple (who loved the mathematics) assured their colleagues to achieve the transfer in the afternoon (the rhythm they had thought was a disc-per-second) by using an additional needle. They couldn't believe him, but he proposed them the following strategy:

- First move the topmost discs (say the top k discs) to one of the spare needles.
- Then use the standard three needles strategy to move the remaining $n - k$ discs (for a general case with n discs) to their destination.
- Finally, move the top k discs into their final destination using the four needles.

He calculated the value to k in order to minimize the number of movements and get a total of 18433 transfers, so they spent just 5 hours, 7 minutes and 13 seconds against the more than 500000 millions years without the additional needle (as they would have to do $2^{64} - 1$ disc transfers. Can you believe it?)

Try to follow the clever priest's strategy and calculate the number of transfer using four needles but according with the fixed and immutable laws of Brahma, which require that the priest on duty must not move more than one disc at a time and that he must place this disc on a needle so that there is no smaller disc below it. Of course, the main goal is to calculate the k that minimize the number of transfers (even thought it is not know for sure that this is always the optimal number of movements).

Input

The input file contains several lines of input. Each line contains a single integer N, which is the number of disks to be transferred. Here $0 \leq N \leq 10000$. Input is terminated by end of file.

Output

For each line of input produce one line of output which indicates the number of movements required to transfer the N disks to the final needle.

Sample Input

```
1
2
28
64
```

Sample Output

```
1
3
769
18433
```

UVa Online Judge

10049 Self-describing Sequence

Solomon Golomb's *self–describing sequence* $\langle f(1), f(2), f(3), \ldots \rangle$ is the only nondecreasing sequence of positive integers with the property that it contains exactly $f(k)$ occurrences of k for each k. A few moments thought reveals that the sequence must begin as follows:

n	1	2	3	4	5	6	7	8	9	10	11	12
$f(n)$	1	2	2	3	3	4	4	4	5	5	5	6

In this problem you are expected to write a program that calculates the value of $f(n)$ given the value of n.

Input

The input may contain multiple test cases. Each test case occupies a separate line and contains an integer n ($1 \leq n \leq 2,000,000,000$). The input terminates with a test case containing a value 0 for n and this case must not be processed.

Output

For each test case in the input output the value of $f(n)$ on a separate line.

Sample Input

```
100
9999
123456
1000000000
0
```

Sample Output

```
21
356
1684
438744
```

UVa Online Judge

846 Steps

One steps through integer points of the straight line. The length of a step must be nonnegative and can be by one bigger than, equal to, or by one smaller than the length of the previous step.

What is the minimum number of steps in order to get from x to y? The length of the first and the last step must be 1.

Input and Output

Input consists of a line containing n, the number of test cases. For each test case, a line follows with two integers: $0 \leq x \leq y < 2^{31}$. For each test case, print a line giving the minimum number of steps to get from x to y.

Sample Input

```
3
45 48
45 49
45 50
```

Sample Output

```
3
3
4
```