Little John studies numeral systems. After learning all about fixed-base systems, he became interested in more unusual cases. Among those cases he found a *Fibonacci system*, which represents all natural numbers in an unique way using only two digits: zero and one. But unlike usual binary scale of notation, in the Fibonacci system you are not allowed to place two `1`s in adjacent positions.

One can prove that if you have number $N = \overline{a_n a_{n-1} \ldots a_1}_F$ in Fibonacci system, its value is equal to $N = a_n \cdot F_n + a_{n-1} \cdot F_{n-1} + \ldots + a_1 \cdot F_1$, where F_k is a usual Fibonacci sequence defined by $F_0 = F_1 = 1$, $F_i = F_{i-1} + F_{i-2}$.

For example, first few natural numbers have the following unique representations in Fibonacci system:

$$\begin{aligned}
1 &= 1_F \\
2 &= 10_F \\
3 &= 100_F \\
4 &= 101_F \\
5 &= 1000_F \\
6 &= 1001_F \\
7 &= 1010_F
\end{aligned}$$

John wrote a very long string (consider it infinite) consisting of consecutive representations of natural numbers in Fibonacci system. For example, the first few digits of this string are `110100101100010011010...`

He is very interested, how many times the digit `1` occurs in the N-th prefix of the string. Remember that the *N-th prefix* of the string is just a string consisting of its first N characters.

Write a program which determines how many times the digit `1` occurs in N-th prefix of John's string.

Input

The input file contains several test cases, each of them as described below.

The input contains a single integer N ($0 \leq N \leq 10^{15}$).

Output

For each test case, output a single integer — the number of '`1`'s in N-th prefix of John's string — on a line by itself.

Sample Input

```
21
```

Sample Output

```
10
```