Computation of **Discrete Fourier Transform** (DFT) is necessary in many computer programs that require some kind of digital signal processing, like audio/image processing and spectral analysis. The most popular algorithm for computing DFT is the **Fast Fourier Transform** algorithm (FFT), which takes profit from the factorization into small prime factors of the number of samples that the digital signal consists of (say, n). The most widely used variant of the FFT algorithm is the so called Radix-2 FFT, which assumes that n is a natural power of 2. Its main drawback is that the number of samples has to be increased (usually filled with zeroes) to the smallest natural power of 2 that is greater than or equal to n. This implies that given a number of samples n, the actual number of samples to be transformed may grow up to $2n$, which in practical terms means a 100% computation overload.

An alternative that substantially reduces this upper bound is using a mixed Radix-2/3 FFT algorithm. In this case, the number of samples to be transformed should be expressed as a product of prime factors 2 and 3:

$$C_{2,3} = \{n = 2^i \cdot 3^j, \text{ such that } i, j \text{ belong to } N\}$$

Given a positive integer number m, find the smallest number n in the set $C_{2,3}$, as defined above, such that $n \geq m$. We will denote this number as $n = Next_{2,3}(m)$.

Input

The input consists of a sequence of positive integer numbers, m, one number per line. The end of the input is marked by a value $m = 0$. No input number m shall be greater than 2^{31}.

Output

For each non-zero input value m, the program is to write one line with the value of $Next_{2,3}(m)$, as defined above. No trailing/leading blank spaces should be written after/before any output number.

Sample Input

```
100
108
1000
3000
0
```

Sample Output

```
108
108
1024
3072
```