Problem D. FFT Algorithm

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1.5 seconds
Memory limit:	256 mebibytes

When I want to apply FFT *algorithm* to polynomial of degree less than 2^k in modular arithmetics, I have to find ω — a primitive 2^k-th root of unity.

Formally, for two given integers m and k, I should find any integer ω such that:

- $0 \le \omega < m$,
- $\omega^{2^k} \equiv 1 \pmod{m}$,
- $\omega^p \not\equiv 1 \pmod{m}$ for all $0 < p < 2^k$.

In this task, I ask you to find ω for me, or determine that it does not exist. Since we talk about application of FFT, I've set some reasonable **limitations for** k: for smaller k naive polynomial multiplication is fine, and for larger k FFT takes more than 1 second (we are competitive programmers after all).

Input

The only line of input contains two integers m and k ($2 \le m \le 4 \cdot 10^{18}$, $15 \le k \le 23$).

Output

Print any ω satisfying the criteria, or print -1 if there is no such ω.

Examples

standard input	standard output
998244353 23	683321333
1048576 15	64609
3 23	-1