RSA factoring

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 megabytes

RSA is a cryptosystem, which security relies on numbers $n = pq$, where p and q are different prime numbers. The number n is called an RSA modulus and is being used for calculations. RSA is believed to be secure because of the fact that for a given number n there is no known fast algorithm for factoring n into prime factors, when the bit length of n is high enough (1024 bits and longer). It is usually adviced to pick p and q as large random primes of approximately the same length. Generating the RSA modulus n is a process that doesn't allow mistakes, since there are many RSA attacks exploiting the faulty generation method. There are more details about RSA, but you are not going to need them to solve this problem.

When Carl read about using close primes in RSA, he implemented the following algorithm.

1. Generate a random prime number p_1 with b bits.
2. Starting from $p_1 + 1$, try all possible numbers in increasing order, until we encounter the next prime number p_2.
3. Return $n = p_1 p_2$.

Because on the first step we chose a random prime p_1 and the distance between adjacent primes is small on average, this algorithm is going to find p_2 rather quickly. Carl's friend Pierre realized that numbers found by Carl's algorithm can't be used for RSA, because it can be factored efficiently. Then Pierre offered to use not just two prime numbers, but four! Independently of p_1 we also choose a random b-bit prime number q_1 and the next prime after it q_2. Then, we set the RSA modulus to be equal to the product of all four primes: $n = p_1 p_2 q_1 q_2$. This algorithm turned out to be faulty too: it is still possible to factor such n.

You are given n generated using the first Carl's algorithm with 2 prime divisors, or with Pierre's updated algorithm with 4 prime divisors. Find all its prime divisors.

Input

The first line contains two integers b and k ($4 \le b \le 60$, $k = 2$ or $k = 4$). The next line contains number n in hexadecimal format, most significant digits first, without leading zeros.

It's guaranteed that n is a product of exactly k prime numbers, and it's generated randomly using either Carl's or Pierre's algorithm described in the problem statement. Each prime number has exactly b bits in binary, all prime divisors are different.

Output

Output k prime divisors of n in hexadecimal format without leading zeros, each one in its own line.

Scoring

Subtask	Score	Constraints
1	10	$b \le 8$, $k = 2$
2	10	$b \le 8$, $k = 4$
3	7	$b \le 15$, $k = 2$
4	8	$b \le 15$, $k = 4$
5	15	$b \le 30$, $k = 2$
6	15	$b \le 30$, $k = 4$
7	15	$b \le 60$, $k = 2$
8	20	$b \le 60$, $k = 4$

Examples

standard input	standard output
4 2 8f	b d
6 4 534ee3	25 29 3b 3d

Explanations

In the first example $n = \texttt{8f}_{16} = 143 = 11 \cdot 13$. $\texttt{b}_{16}$ is 11, and $\texttt{d}_{16}$ is 13. In the second example $n = 5459683$, which is represented as a product $37 \cdot 41 \cdot 59 \cdot 61$.