As a general problem-solving tip: If you're confronted with an unusual object or definition, one of the best ways to start understanding it is to just *work out some concrete examples* for yourself. This can help give you an intuition for how the given object behaves, or properties that it tends to have. It works especially well if you ask yourself, "Can the insights I used for these specific cases be generalized?"

For these problems in particular, *just write out the sequences* on pen and paper, and many properties of the sequence will become apparent.

A

The sequence just goes,

$$1, 1, 1, 1, 1, \ldots$$

Proof: We remind you that each next term in the sequence is the *smallest* positive integer such that its sum with any previous term in the sequence always gives a prime.

If **all** previous terms are 1, then the next term will again be 1. That's because the answer has to be at least 1 (the smallest positive integer), and 1 works (because $1 + 1 = 2$ is prime).

You can formalize the above argument with math induction, if that pleases you.

```
print(1)
```

B

The sequence just goes,

$$1, 3, 3, 3, \ldots$$

Proof: We know that $A_2 \neq 1$ and $A_2 \neq 2$ because $1 + 1 = 2$ and $1 + 2 = 3$ are both not composite. But since $1 + 3 = 4$ *is* composite, we conclude that $A_2 = 3$ since 3 is the smallest positive integer satisfying the condition.

The argument then proceeds similarly as in Problem A. If **all** previous terms are 1 and 3, then the next term will again be 3. That's because the answer has to be at least 3 (it can't be anything lower, as otherwise 1 plus it wouldn't be composite), and 3 works (because both $1 + 3 = 4$ and $3 + 3 = 6$ are composite).

Again, you can formalize the above argument with math induction, if that pleases you.

```
print(1 if int(input()) == 1 else 3)
```

K

Let's gather intuition by actually writing out the first few terms of the sequences for small values of m.

$$\begin{array}{llllllll} 1, & 3, & 3, & 3, & 3, & 3, & \dots \\ 2, & 2, & 2, & 2, & 2, & 2, & \dots \\ 3, & 1, & 3, & 3, & 3, & 3, & \dots \\ 4, & 2, & 2, & 2, & 2, & 2, & \dots \\ 5, & 1, & 3, & 3, & 3, & 3, & \dots \end{array}$$

Maybe when writing out $m = 5$, you already start to notice something with what you're doing. You write out the following conjecture.

Claim: If m is odd and greater than 1, then the sequence goes,

$$m, 1, 3, 3, 3, 3, \dots$$

Proof:

- If m is odd and greater than 1, then $m + 1$ is even and greater than 2.
- Therefore, $m + 1$ is composite, and $A_2 = 1$
- $A_3 \neq 1$ and $A_3 \neq 2$ because $1 + 1 = 2$ and $1 + 2 = 3$ are both not composite.
- However, $1 + 3 = 4$ is composite, and $m + 3$ is an even number greater than 2, so it's composite as well. So, $A_3 = 3$.
- By similar logic as in Problem B, $A_n = 3$ for all $n \geq 3$ as well.

At this point, you might be tempted to also write the following conjecture:

Claim (?): If m is even, then the sequence goes,

$$m, 2, 2, 2, 2, 2, \dots$$

Proof (?): With similar logic as in our previous proof, if m is even, then $m + 2$ is an even number greater than 2, which means it is composite. If $A_2 = 2$, then it follows that $A_n = 2$ for all $n \geq 2$ because $2 + 2 = 4$ is composite.

Here's the *big problem* though: We define A_2 to be the **smallest** positive integer such that $m + A_2$ is composite. Surely $m + 2$ is composite... but don't forget that it's possible for $m + 1$ to be composite? If $m + 1$ is not composite, then the previous claim is in fact the pattern. But if it is...

Here's the sequence for $m = 8$. Note that $m + 1 = 9$ is composite (in fact, 9 is the smallest odd composite number).

$$8, 1, 7, 7, 7, 7, \dots$$

In fact, let's analyze a few more sequences where m is even and $m + 1$ is composite.

$$\begin{array}{rllllll} 8, & 1, & 7, & 7, & 7, & 7, & \dots \\ 14, & 1, & 7, & 7, & 7, & 7, & \dots \\ 20, & 1, & 5, & 5, & 5, & 5, & \dots \\ 24, & 1, & 3, & 3, & 3, & 3, & \dots \\ 26, & 1, & 7, & 7, & 7, & 7, & \dots \end{array}$$

Oh, it actually does still have some structure, doesn't it? So you make the following conjecture:

Claim: If m is even and $m + 1$ is composite, then the sequence goes,

$$m, 1, k, k, k, k, \dots$$

where k is the smallest positive integer such that both $m + k$ and $1 + k$ are composite.

Proof: By definition, $A_3 = k$. But note that we can also say that $A_n = k$ for $n \geq 4$ because $k+k = 2k$ is composite, because $2k$ is even and greater than 2. We know it's greater than 2 because we can show that $k \neq 1$ (by noting that $1 + 1 = 2$ is not prime).

The final question: *What's the formula for finding this k?* Well... there's no easy "formula" for it. But that's okay... just do a brute force search! Check the positive integers $k = 1, 2, 3, 4, \ldots$ until you find the first one such that $m+k$ and $1 + k$ are both composite.

We can show that we will not have to test that many numbers, because we can put an **upper bound** to k: the value of k will not exceed 8, because 8 is guaranteed to work. Both $m + 8$ (an even number greater than 2) and $1 + 8 = 9$ are both composite. So we will never need to do more than 8 primality tests. In fact, the only numbers we need to test are $k = 3, 5, 7$, and if they all fail, then the answer is 8.

Okay, the *real* final question is how to do a quick primality test. But this part of the problem is easily Google-able. We can check if a number n is prime in $\mathcal{O}(n)$ time by directly implementing the definition of a prime number: Does there exist a d such that $1 < d < n$ and n is divisible by d? Check all d in this range.

But there's a neat optimization. Let d be a positive divisor of n. Note that n/d is also a positive divisor of n. Furthermore, **at least one of d and n/d is $\leq \sqrt{n}$**. If both d and n/d were greater than $\sqrt{n}$, then we would have a contradiction:

$$\begin{aligned} d &> \sqrt{n}, \\ n/d &> \sqrt{n}, \\ \text{(multiply the two inequalities) } d \times (n/d) = n &> \sqrt{n} \times \sqrt{n} = n, \end{aligned}$$

but the implication of $n > n$ is absurd.

Essentially, **divisors come in pairs**, with one value being $\leq \sqrt{n}$ and the other one being $\geq \sqrt{n}$. This gives us an easy way to modify the obvious primality testing algorithm to run in $\mathcal{O}(\sqrt{n})$ time: **only check d up to $\sqrt{n}$**. This works because *if such a d did exist*, then either it is $\leq \sqrt{n}$, or its "partner" is. If no divisors were found $\leq \sqrt{n}$, then no divisors *overall* exist, and n is prime.

Since this primality testing algorithm runs in square root time, and we only need to test $m + k$ for k up to 8, our overall solution runs in $\mathcal{O}(\sqrt{m})$ time for this case.

```
def is_prime(n):
    d = 2
    while d*d <= n:
        if n % d == 0:
            return False
        d += 1
    return True

def is_composite(n):
    return n > 1 and not is_prime(n)

m, n = map(int, input().split())

if m == 1:
    if n == 1:
        print(m)
    else:
        print(3)
elif m % 2 == 1:
    if n == 1:
        print(m)
    elif n == 2:
        print(1)
```

```
    else:
        print(3)
else:
    if is_composite(m+1):
        if n == 1:
            print(m)
        elif n == 2:
            print(1)
        else:
            k = 2
            while not (is_composite(m+k) and is_composite(1+k)):
                k += 1
            print(k)
    else:
        if n == 1:
            print(m)
        else:
            print(2)
```

D

The main idea for the solution to this problem is that *the described sequence is very short.* Note that each next term in the sequence is found by just taking the square root of the previous term. Then you round down or maybe do -1 because of the strict inequality, but those are minor details.

Now consider, intuitively, that the size of the terms in the sequence should look something like this:

$$m, \sqrt{m}, \sqrt{\sqrt{m}}, \sqrt{\sqrt{\sqrt{m}}}, \ldots$$

Just try out some examples for m. Even if we start with a huge m, that above sequence shrinks *very* quickly. For the maximum $m = 10^9$, we can make the values in the sequence ≤ 2 by using only 5 square roots.

In fact, the terms of A are actually *smaller* than taking repeated square roots, because of the rounding and strict inequality to keep things as integers. So A is likely to be very, very short.

So, just simulate the process. That should be fast enough. You can compute each next term in the sequence in $\mathcal{O}(\sqrt{\text{previous term}})$ by just directly checking $1, 2, 3, 4, \ldots$ one by one. So, the running time is bounded by $\mathcal{O}(\text{length of sequence} \times \sqrt{m})$, where we expect "length of sequence" to not be greater than ≈ 5.

```
m, n = map(int, input().split())

def iterate(A):
    if A <= 1:
        return -1
    else:
        # find the smallest k that _fails_ the test
        k = 2
        while k*k < A:
            k += 1
        return k-1

A = m
t = 1
while t != n:
    if A == -1:
        break
    else:
        A = iterate(A)
    t += 1

print(A)
```

For bonus enrichment, let's figure out how to bound the growth of the length of the sequence. I promise it's not too hard!

If you recall that $\sqrt{m} = m^{1/2}$, then the kth term in the repeated-square-root sequence (m enclosed in $k-1$ square root symbols) can be written as $m^{1/2^{k-1}}$. And since we said that the terms of A are actually *smaller* than the ones in the repeated-square-root sequence, we have that $A_k \leq m^{1/2^{k-1}}$.

Let's do some algebra to try to place a bound on the first k such that $m^{1/2^{k-1}} \leq 2$. Since $m^{1/2^{k-1}}$ is a strictly decreasing sequence on k (if $m > 1$), let's set $m^{1/2^{k-1}} = 2$ and solve for k. The actual value for k will then just be

some nearby integer.

$$\begin{aligned} m^{1/2^{k-1}} &= 2 \\ m &= 2^{2^{k-1}} \\ \log_2(m) &= 2^{k-1} \\ \log_2(\log_2(m)) &= k-1 \\ k &= \log_2(\log_2(m)) + 1. \end{aligned}$$

Therefore, the length of the sequence will be $\mathcal{O}(\log_2(\log_2(m)))$, which is an *incredibly* slow-growing function.