Example: "Brute force" for GCD

(greatest common divisor)

$$\gcd(a,b) = \max_{d \in \mathbb{Z}} \{ d \mid a \text{ and } d \mid b \}.$$

$\uparrow$

$\exists\ q \in \mathbb{Z}$ s.t. $a = d \cdot q$

$\uparrow$

"there exists"

E.g., $\gcd(6, 9) = 3$

$2 \cdot 3 \quad 3 \cdot 3$

Idea: start w/ largest candidate ($\min(a,b)$)
& count backwards until the first common divisor is found.

```
int main() {
    int a,b;
    int min;
    cin >> a >> b;
    if (a<b)
        min = a;
    else
        min = b;
    int d = min; // d = candidate for gcd
    while ( a%d!=0 || b%d!=0)
        d--;      // keep looking
    cout << "gcd = " << d;
```

```
    return 0;
}
```

Exercise! Find the exponent of 2 in the factorization of an integer.

E.g. if input = 40 = $5 \cdot 2^{3}$ ← answer is 3.

$9 = 3^{2} \cdot 2^{0}$

$6 = 3^{1} \cdot 2^{1}$

Again, we'll kind of "brute force" the answer.

Idea: start w/ 2^0 and keep multiplying by 2 until division has a remainder.

`1<<k` == 2^k

$$100 = 10^2 \quad \text{(base 10)}$$

$$100 = 2^2 \quad \text{(base 2)}$$

Alternate approach:

count the # of times 2 divides the input.

$n = r \cdot 2^k$, r odd (so k == answer)

((n/2)/2)/2

count # of divisions to find k.

40 / 2 = 20

20/2 = 10

10/2 = 5 odd so k = 3.

```
int n;
cin >> n;
int k = 0; // count # times was divided by 2.
while(n%2 == 0) {
    n = n/2;
    k++; // k = k+1
}
cout << k;
```