"For loops" happens just once. happens @ end of body.

```
for (int i=0; i<10; | i++) {
```

"body"

checked before running body

```
    // arbitrary statements (including
    // other loops ...)
}
```

Equivalent while loop:

```
int i=0;
while (i<10) {
    // statements...
    i++;
}
```

Exercise: read 2 integers, n, m and draw a n×m rectangle of '*' characters to stdout.

E.g. if n=3, m=4

```
****
****
****
```

New "technique": Brute force — leverage the speed of the computer to search through the entire space of solutions.

All that's needed: ① a test: "is x a solution?"
② a method to enumerate all possible solutions.

Brute force:

```
for (/* x in possible solns */) {
    if (/* x passes test */) {
        cout << x;
        break;
    }
}
```

Example: compute the gcd of two integers m, n.
(GCD ≡ greatest common divisor)

e.g. gcd(10, 15) = 5. ($10 = 2^1 \cdot 3^0 \cdot 5^1$, $15 = 2^0 \cdot 3^1 \cdot 5^1$)

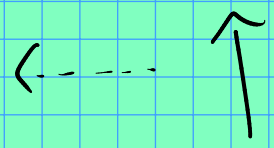
Test for solution?
Warm up: test for common divisor?
How to check if one integer d divides another n?
see if n/d has no remainder.
Note: the % operator gives you the remainder!
So, $\boxed{n \% d == 0} \iff$ d divides n
∴ we can check to see if d is a common divisor of m, n via:
(m%d == 0 && n%d == 0)

What about (2)? List of candidates for gcd(m, n)?

Smallest value: 1

Largest value: $\min\{m, n\}$

Our list: $[1, 2, 3, \ldots, \min\{m, n\}]$.



Idea: starting from largest candidate, test it and count backwards. First candidate that passes is the gcd.