#2) - Algorithm: need 2 divide b by 2
- use a ~~for~~ while loop (int; b>0;) {...}
- repeated square only work for powers of 2 (even powers)
  - handle odd powers
    - condition for when b is odd
- need a third variable to pass in as answer
- any a^{even} can be written as $a_1 * \ldots * a_b$
  - can be written using powers of a ($a^{n/2}$)
- at every odd # multiply ~~return value~~ answer by a
  - multiply ~~squared square~~ ans by ~~a~~ repeated squares

Also. a = #1
b = #2
ans = 1
while (b > 0) {
if b is odd. #3
multiply ans by a // counts a for repeat #4 ☐
// since s.s. only works for pow of 2
a *= a
b /= 2
}

P2) Proof: Induction

So we have $a^{\wedge}b$

if everytime we divide b and update it's value : $b/=2$, and the new b is even we multiply a by a and update it's value so $a*=a$, which is a^2

as we continue to loop, a^2 becomes $(a^2)^2$ and $((a^2)^2)^2 \ldots a^{\wedge}2^{\wedge}i$, where i is the # of loops

eventually, $b/2$ will equal 1 and at this point, when b is odd, we can multiply ans which = 1, by the current value of a , which is the answer we are looking for

Now, when b is odd and $!=1$ we can apply the same equation used for $b=1$

So say that ans is currently = to $a^{\wedge}2^{\wedge}i$ and we have run n loops since then, then

$a = a^{\wedge}2^{\wedge}i*n$

now we update ans, so $ans = a^{\wedge}2^{\wedge}i * a^{\wedge}2^{\wedge}i*n$

OR $ans = a^{\wedge}2^{\wedge}i+i*n$ so, essentially $ans = a^{2i+2in}$

where $k=2i+in$ ans $= a^{2i+2in}$ which can be a^{2i+2in} Since $2^{\wedge}0=1$ we can write $a^{\wedge}2^{\wedge}i + 2^{\wedge}i*n$ as any power of a, odd powers inclusive.

Algo Analysis:

```
a = a
b = b          } O(1)
ans = 1

while ( b > 0 ) {          O(log b)
    if ( b%2 == 1 ) {
        ans *= a           } O(1)
    }
    a *= a;  O(1)
    b /= 2;  O(1)  // brings us 2x closer to goal
}
```

$O(1) + O(1) + O(\log b)$

$O(1) + O(\log b)$

$O(\log b)$

For an algorithm to run in $O(\log b)$ its loop has to get twice as close to its goal with each iteration. With $b/2$, b becomes half as large with each iteration, which brings us 2x as close to our goal of ~~b > 0~~ $b! > 0$ (to exit loop)

$\frac{b}{2^i}$ where i = # of loops

denominator doubles with each iteration which leads to $O(\log b)$