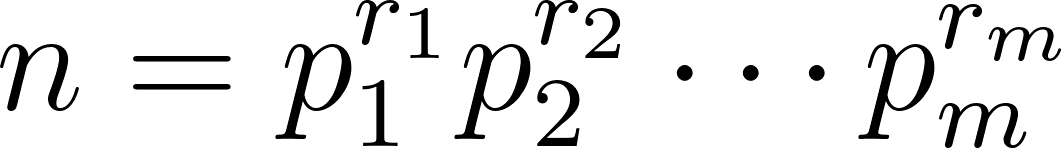
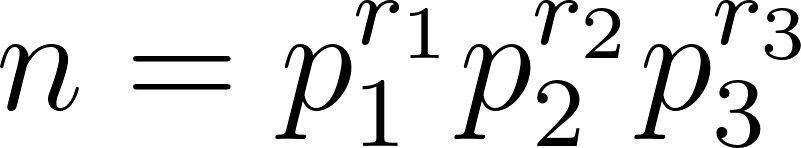
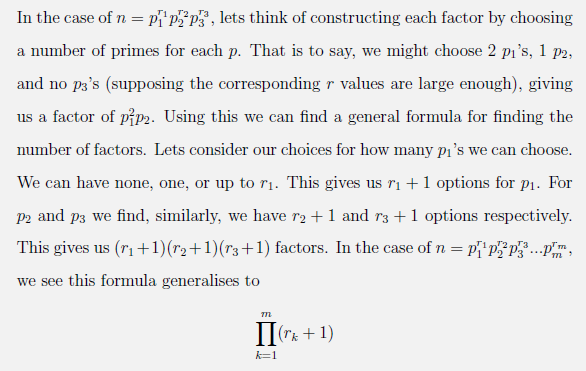
1. **a.** Given an integer [](https://www.codecogs.com/eqnedit.php?latex=n%3Dp_1%5E%7Br_1%7Dp_2%5E%7Br_2%7D%5Ccdots%20p_m%5E%7Br_m%7D#0), how many factors does *n* have? (You can consider a special case of [](https://www.codecogs.com/eqnedit.php?latex=n%3Dp_1%5E%7Br_1%7Dp_2%5E%7Br_2%7Dp_3%5E%7Br_3%7D#0) if you’d prefer to work with a slightly more concrete case)

* The number of factors of *n*, using the multiplication principle, is . This counts the number of integers whose prime factors are a subset of the prime factors of *n* and where the powers on those prime factors are less than or equal to the corresponding powers on the prime factors of *n*, while remaining nonnegative. Each prime factorization in this list will be unique so there will be no repeats of the integers that factorization corresponds to.

Another version (similar idea but it might be more helpful to some because it uses the specific example as a stepping stone):



**b.** Suppose we are looking to find out which *n* have an odd number of factors? What does part **a** tell about *n*?

* Part **a** tells us that for *n* to have an odd number of factors, the product of one more than the exponents of the prime factorization of *n* must be odd. Thus all of the exponents of the prime factorization of *n* must be even, making *n* a perfect square.

Another version:

* All of the powers of the primes must be even so that that powers plus 1 are odd

**c.** Now let’s use the other approach of finding factors of a number: pairing a factor *r* with *n/r*. What does this tell us about *n* if *n* has an odd number of factors? Which method was easier in determining this property?

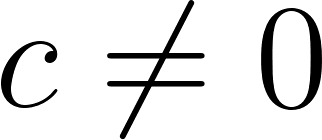
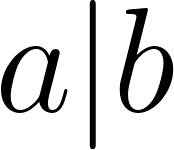
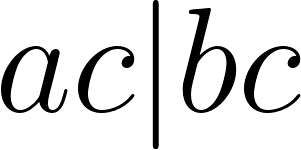
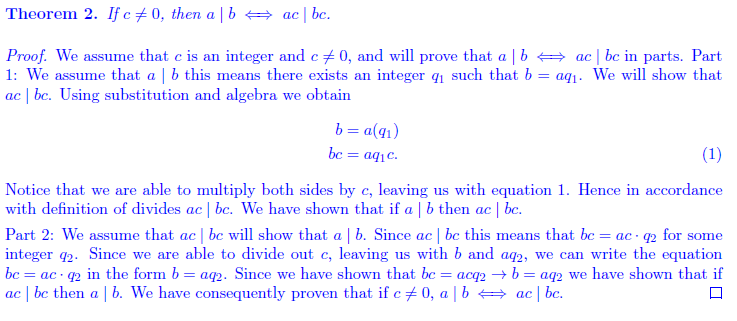
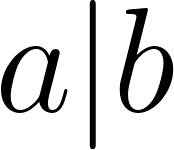
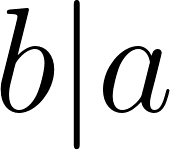
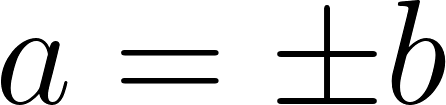
* If *n* has an odd number of factors, then some factor must have been paired with itself. Thus there exists some integer *r* such that and thus that . Thus if *n* has an odd number of factors then *n* is a perfect square. This method was easier in determining that fact.

Another version:

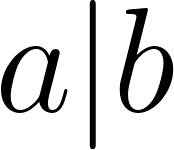
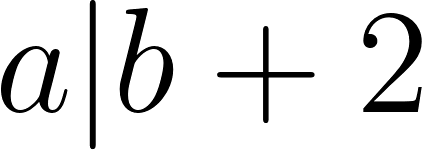
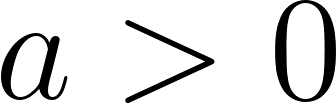
* When pairing factors *r* with *n*/*r*, one factor must be left out and that will be the one in the middle paired with itself. That means *n* is a square.

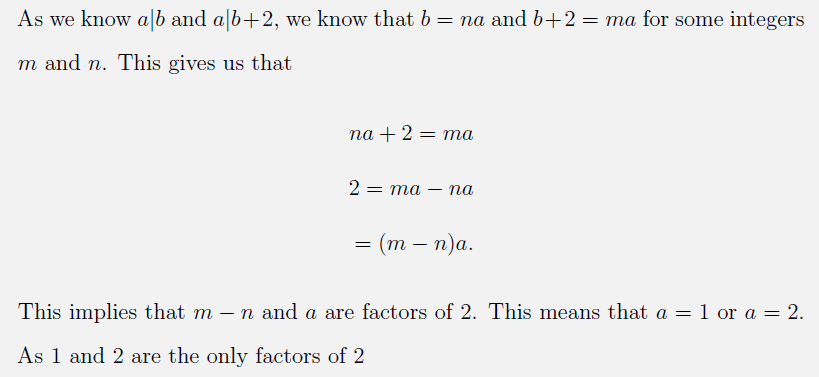
1. If we know that *d* | *a* and *d* | *b*, what other divisibility relations can be concluded? List a few examples and prove the most general conclusion you can make.

* for some integer k
* for some integer m
* *d | a+b*
* *d | a-b*
* d|xa+yb where x,y are integers. This is because a=md and b=nd for some integers m,n. So we can write xa+yb as x(md)+y(nd)=(xm+yn)d. So d divides linear combinations of a and b

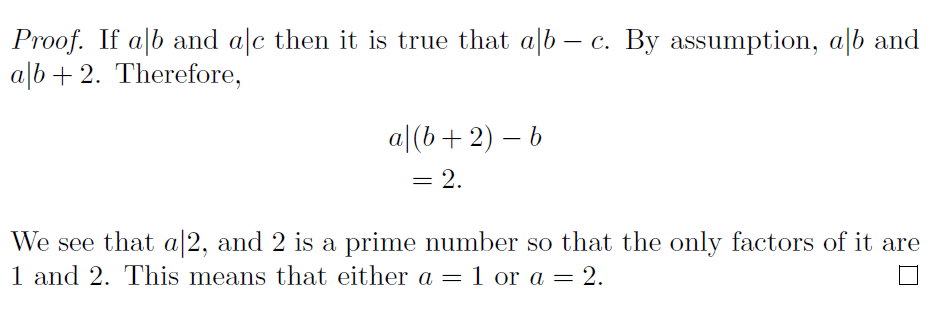
1. Prove: If [](https://www.codecogs.com/eqnedit.php?latex=c%5Cneq%200#0), then [](https://www.codecogs.com/eqnedit.php?latex=a%7Cb#0) iff [](https://www.codecogs.com/eqnedit.php?latex=ac%7Cbc#0). (Iff is short for if and only if.)  
   
2. Prove: If [](https://www.codecogs.com/eqnedit.php?latex=a%7Cb#0) and [](https://www.codecogs.com/eqnedit.php?latex=b%7Ca#0), then [](https://www.codecogs.com/eqnedit.php?latex=a%3D%5Cpm%20b#0).

We will assume that *a* and *b* are nonzero integers. We know that *b=ak* and *a=bm*, where *m* and *k* are some integers. Substituting one of our defined variables, *a* or *b,* in for the other we see that *b=bmk* or *a=amk*. In order for the equation to hold *mk* must equal 1. Since *m* and *k* are integers, the only integers multiplied together to get 1 are if *m* and *k* are both +1 or both -1. With this we know that *m* will always be ±1 and *k* will always be ±1, thus *a=b(*±1)=±*b*.

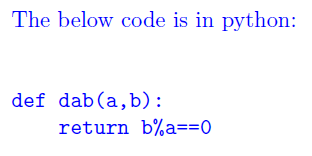
1. Prove: If [](https://www.codecogs.com/eqnedit.php?latex=a%7Cb#0) and [](https://www.codecogs.com/eqnedit.php?latex=a%7Cb%2B2#0) and [](https://www.codecogs.com/eqnedit.php?latex=a%3E0#0), then [](https://www.codecogs.com/eqnedit.php?latex=a%3D1#0) or [](https://www.codecogs.com/eqnedit.php?latex=a%3D2#0).



Another version (similar idea, but it uses a “lemma”, the first statement):



1. Write a code to check if a|b is true or not. (Once you have a draft, feel free to Google to see if there’s a shorter solution.)



Note: Python allows negative numbers in modulo operation. It returns the least residue for positive moduli. There’s no “least residue for negative moduli” because sane mathematicians don’t allow negative moduli ;)