Note: In order for this homework (and all of your future assignments) to work properly, you need to download and import the cs115.py file from Canvas. Once you've downloaded it into your CS 115 workspace folder, you can import it by typing the following block of code at the top of your Python source file:

```
from cs115 import [what you need]
```

If you need only the map function, type this:

```
from cs115 import map
```

If you need multiple functions, say map and reduce, type this:

```
from cs115 import map, reduce
```

Writing your own factorial function

In the problem above, we used the factorial function in the math module. Here, you'll write your own factorial function. First, we start with a simple function that returns the product of its two inputs:

```
def mult(x, y):
   """Returns the product of x and y"""
   return x * y
```

Nothing too surprising here. Now, take a look at this:

```
>>> reduce(mult, [2, 3])
6
>>> reduce(mult, [2, 3, 4])
24
>>> reduce(mult, [1, 2, 3, 4])
24
```

Notice that `reduce` takes two inputs: A function and a list and it applies that function to "compress" the list into a single value. In this case, it multiplied all of the values together.

Now, write a function `factorial(n)` that takes a positive integer `n` and returns `n!`.

This is "mean"...

Finally, write a function called `mean(L)` that takes a list as input and returns the mean (average) value in that list. Using `reduce` will be handy here. You may also want to define an `add` function that returns the sum of two numbers. You'll need to know the number of elements in the list. This can be found using the built-in function `len`. For example:

```
>>> len([1, 3, 5])

3

>>> len(range(1,10))

9
```

Here is the `mean` function in action:

```
>>> mean([1, 2, 3])

2

>>> mean([1, 1, 1])

1

>>> mean([1, 2, 3, 4])

2
```

Hmmm, that last value is a little suspicious! Do you see what happened here? Try to fix this!

Testing for Prime Numbers

First, take a look at this friendly little function:

```
def div(k):

    return 42 % k == 0
```

This function takes as input an integer `k` and then returns the result of evaluating the expression

```
42 % k == 0
```

The left-hand side of that expression computes the remainder when 42 is divided by `k`. (`k` need not be an integer, but then computing the remainder modulo `k` is a bit weird!) Next, that remainder is tested to see if it is equal to 0 (that's what the double equal sign is doing). The result is a boolean value - either `True` or `False`. Try this function out.

Next, take a look at this strange Python function called `divides`:

```
def divides(n):
```

```
    def div(k):
        return n % k == 0
    return div
```

Notice that this function has another function, `div`, that is defined inside it.
Moreover, `divides` returns `div`. Weird! We are returning a function rather than a number! This is a lovely feature of Python and many so-called "functional" programming languages (e.g. Scheme, Haskell, ML, among others). Play with `divides` and make sure that you feel comfortable with what is going on here.

Now, here's your challenge. Write a function called `prime(n)` that takes a positive integer `n` as input and returns `True` or `False` depending on whether `n` is prime or composite. You should not use any loop structures or recursion here. Instead, you may use `map`, you may call the `divides` function above, and you may wish to use `sum` which takes a list of numbers as input and returns the sum of the numbers in that list. Aside from the `def prime(n)` line, your program should be at most three lines long (although it can be done in fewer lines!).