Recursive Functions

Here is a recursive function to find $n!$ (factorial).

```python
def fact(n):
    if n == 0:
        return 1
    else:
        return n * fact(n - 1)
```

To find $\text{fact}(4)$, we simply write out each step in the recursive computation:

- $\text{fact}(4)$
- $4 \times \text{fact}(3)$
- $4 \times 3 \times \text{fact}(2)$
- $4 \times 3 \times 2 \times \text{fact}(1)$
- $4 \times 3 \times 2 \times 1 \times \text{fact}(0)$
- $4 \times 3 \times 2 \times 1 \times 1$

**Exercise 1**

```python
def m(a, b):
    if a < b:
        return a
    else:
        return m(a - b, b)
```

Find $m(3, 5)$.
Find $m(7, 5)$.
Find $m(14, 5)$.
What does $m$ do?
**Exercise 2**

```python
def gcd(a, b):
    if b == 0:
        return a
    else:
        return gcd(b, a % b)
```

Use this definition to find gcd(15, 9).
Now find gcd(13, 8).

**Exercise 3**

The function `power` should compute the value of $b^n$, where $n$ is any non-negative integer. Fill in the blanks below, so that `power` behaves correctly.

```python
def power(b, n):
    if n == ________:
        return ________
    else:
        return ________ * power(________, ________)
```

Hint: How does knowing the value of $3^9$ help you find the value of $3^{10}$?

**Exercise 4**

```python
def double(n):
    if n == 0:
        return 1
    else:
        return double(n - 1) + double(n - 1)
```

Find `double(3)`.
What does this function do?
Can you modify the definition of `double` so that it computes the same result with a single recursive call?
**Exercise 5**

def f(x):
    if x == 1:
        return 1
    else:
        if x % 2 == 0:
            return f(x / 2)
        else:
            return f(3 * x + 1)

Find \( f(3) \).
Find \( f(7) \).
Can you find a positive integer \( x \) so that \( f(x) \) results in an infinite loop?