1. An Iterative Greatest Common Divisor

Take a look at the following algorithm for finding the gcd (greatest common divisor) of two numbers:

```python
def gcd(a, b):
    # precondition: a, b positive natural numbers
    while b != 0:
        d = abs(b - a)  # a is the minimum of the previous a and b
        a = min(a, b)  # b is the difference between the previous a and b
        b = d
    # postcondition: (a) a is a positive natural number
                   (b) a | a0, b0
                   (c) if k | a0, b0, then k | a
    return a
```

[Recall that | means “divides.”]

Here, $a$ is the final value of $a$, and $a_0$ and $b_0$ are the initial values of $a$ and $b$, respectively.

First, let’s trace through a couple examples to see how this algorithm works.

<table>
<thead>
<tr>
<th>$a$</th>
<th>$b$</th>
<th>$a$</th>
<th>$b$</th>
<th>$a$</th>
<th>$b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>3</td>
<td>6</td>
<td>15</td>
<td>12</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>6</td>
<td>9</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>6</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note that at every step, the greatest common divisor of $a$ and $b$ is the same. Intuitively, this is because if $a$ and $b$ share a divisor, so does their minimum as well as their difference. Justify to yourself that the following is true to get some intuition behind the algorithm:

$k | a, b ⇔ k | a_0, b_0$  
$k | a, b ⇔ k | abs(a - b)$

Recall that in order to prove program correctness, we need to prove Partial Correctness and Termination. Proving a loop invariant (something that is true at every iteration) is an essential part of proving partial correctness of iterative programs. We need to formalize the idea that the gcd of $a$ and $b$ never changes.

2. Partial Correctness

**Loop Invariant:**

(i) $a$ and $b$ are natural numbers [this is easy to prove since abs and min preserve this]
(ii) $a$ is positive [this is also easy, since $a$ is the minimum of two positive numbers]
(iii) $k | a_0, b_0$ iff $k | a, b$ [this is harder to prove, but still not so bad!]

Recall that $a_i$ is the value of the variable $a$ after $i$ iterations. Proving the loop invariant is proving the following lemma:

**Lemma 1.** $\forall n \in \mathbb{N}$, if the precondition holds and the loop is run for $n$ steps, then:

(i) $a_n, b_n \in \mathbb{N}$
(ii) $a_n > 0$
(iii) $\forall k \in \mathbb{N}, k | a_0, b_0 \leftrightarrow k | a_n, b_n$

**Proof.** Proof by Induction on $n$ (the number of iterations).

**Base case:** Assume the precondition holds and the loop is run for 0 iterations. Then (i) and (ii) are true from the precondition. (iii) becomes $\forall k \in \mathbb{N}, k | a_0, b_0 \leftrightarrow k | a_0, b_0$, which is a tautology.

**Induction step:** Let $n \in \mathbb{N}$. Assume that if the precondition holds and the loop is run for $n$ steps, then (i), (ii) and (iii) hold. [IH]

Now assume that the precondition holds and the loop is run for $n + 1$ steps.

Note that $a_{n+1} = \min(a_n, b_n)$ and $b_{n+1} = |b_n - a_n|$. 

1
To prove (i):
\(a_{n+1} \in \mathbb{N}\) since \(a_n, b_n \in \mathbb{N}\) by IH and \(\min\) preserves this.
\(b_{n+1} \in \mathbb{N}\) since \(a_n, b_n \in \mathbb{N}\) by IH and subtraction and absolute value preserves this.

To prove (ii):
\(a_n > 0\) by IH. \(b_n > 0\) since \(b_n \in \mathbb{N}\) by IH and \(b_n \neq 0\) by the while loop condition. Since \(a_{n+1} = \min(a_n, b_n)\), \(a_{n+1} > 0\).

To prove (iii):
We'll prove both directions of the iff statement separately.

Assume \(k | a_0, b_0\). Then \(k | a_n, b_n\) by IH. Our goal is to show that \(k \mid a_{n+1}, b_{n+1}\). By the definition of divides, there are \(a', b' \in \mathbb{N}\) such that \(a_n = ka'\) and \(b_n = kb'\).
\(a_{n+1} = \min(a_n, b_n) = ka'\) or \(kb'\), depending which is smaller. So \(k | a_{n+1}\).
\(b_{n+1} = |b_n - a_n| = |k b' - k a'| = k | b' - a'|.\) So \(k | b_{n+1}\).

Now assume \(k | a_{n+1}, b_{n+1}\). Our goal is to show that \(k | a_0, b_0\). Note that because of IH, it is sufficient to show that \(k | a_n, b_n\). By the definition of divides, there are \(a'', b'' \in \mathbb{N}\) such that \(a_n = ka''\) and \(b_n = kb''\). This direction is a little trickier, so we'll look at cases (not necessary, but it may be easier to think about this way):

Case 1: \(a_n \leq b_n\)
Then \(a_{n+1} = a_n\) and \(b_{n+1} = b_n - a_n\).
\(a_n = \min(a_n, b_n) = a_{n+1} = ka''\). So \(k | a_n\).
\(b_n = (b_n - a_n) + a_n = |b_n - a_n| + \min(a_n, b_n) = b_n + a_n + a_{n+1} = kb'' + ka'' = k(a'' + b'').\) So \(k | b_n\).

Case 2: \(a_n > b_n\)
Then \(a_{n+1} = b_n\) and \(b_{n+1} = a_n - b_n\).
\(a_n = (a_n - b_n) + b_n = |b_n - a_n| + \min(a_n, b_n) = b_n + a_n + a_{n+1} = kb'' + ka'' = k(a'' + b'').\) So \(k | a_n\).
\(b_n = \min(a_n, b_n) = a_{n+1} = ka''\). So \(k | b_n\).

[It’s possible to combine the cases by just noticing that \(a_{n+1} + a_{n+1} = a_n\) and \(b_n\) (not necessarily in that order).] \(\square\)

So now that we have the loop invariant, we can prove Partial Correctness, mainly that the precondition and termination imply the postcondition.

**Theorem (Partial Correctness).** If the precondition holds and the loop terminates after \(n\) steps, then the postcondition holds:
(a) \(a \in \mathbb{N}\) and \(a > 0\).
(b) \(a(a_0, b_0)\)
(c) \(k | a_0, b_0 \rightarrow k | a\)

**Proof.** Note that since the loop has terminated, the while loop condition gives us that \(b_n = 0\).

To prove (a):
Just notice that it follows directly from parts (i) and (ii) of Lemma 1 (the loop invariant).

To prove (b):
Let \(k = a_n\). Note that \(k = a_n | a_n\) and \(k | b_n = 0\). By the \(\rightarrow\) direction of (iii) of Lemma 1, \(k | a_0, b_0\).

To prove (c):
Assume \(k | a_0, b_0\). By the \(\rightarrow\) direction of (iii) of Lemma 1, \(k | a_n\) (and also \(k | b_n = 0\)). \(\square\)

3. Termination

Finally, we can prove that the loop terminates.

Note that neither \(a_0, a_1, \ldots\) nor \(b_0, b_1, \ldots\) are sequences of decreasing natural numbers. However, the maximum and also the sum are.

We’ll show that \(a_0 + b_0, a_1 + b_1, a_2 + b_2, \ldots\) is a sequence of decreasing natural numbers until \(b_1 = 0\). We could also show that \(\max(a_0, b_0), \max(a_1, b_1), \max(a_2, b_2), \ldots\) is a sequence of decreasing natural numbers until \(a_i = b_i\).

Recall that to show that the sequence is decreasing, we need to show that \(a_{n+1} + b_{n+1} < a_n + b_n\).

**Proof.** Assume that the loop is run for \(n + 1\) iterations. Recall that \(a_{n+1} = \min(a_n, b_n)\) and \(b_{n+1} = |b_n - a_n|\).
\(a_{n+1} + b_{n+1} = \min(a_n, b_n) + |b_n - a_n| = \max(a_n, b_n)\)
Since the loop ran through \(n + 1\) iterations, \(b_n \neq 0\) by the while loop condition. Also the loop invariant gives us \(a_n > 0\). Thus \(\max(a_n, b_n) > a_n + b_n\). \(\square\)

4. And now a Recursive GCD

We can similarly define a recursive algorithm for GCD. Note that the algorithm is essentially the same as the iterative one.

def gcd(a, b):
    #precondition: a, b positive natural numbers
    if a == b:
        return a
    if a > b:
        return gcd(b, a - b)
    return gcd(a, b - a)
return a
else:
    return gcd(min(a, b), abs(b-a))

#postcondition: returns c such that:
    (a) c is a positive natural number
    (b) c|a,b
    (c) if k|a,b, then k|c

Proving correctness of recursive programs is very similar to proving correctness of iterative programs. Here, we don’t have a separate partial correctness and termination. We prove them together. Here is the predicate that we’ll prove for all n:

P(n) = if a, b ∈ N and 1 ≤ a, b ≤ n, then gcd(a, b) terminates and returns the gcd of a and b

Theorem. ∀n ∈ N, n ≥ 1 → P(n).

Proof. By induction on n.

    Base case: Let n = 1. So a = b = 1. Then gcd returns 1.
    Induction step: Let n ∈ N such that n ≥ 2.

Case 1: a = b
Then gcd terminates and returns a, which is the gcd of a and b.

Case 2: a ≠ b
Assume that ∀k ∈ N, 1 ≤ k ≤ n → P(k). [IH]
Since a ≠ b and a, b ≤ n, min(a, b) < n. Furthermore, since 1 ≤ a, b ≤ n, abs(b - a) < n.
By IH, gcd(min(a, b), abs(b - a)) terminates and returns the gcd of min(a, b) and abs(b - a).
Therefore, gcd(a, b) terminates and returns the gcd of min(a, b) and abs(b - a).

Now we just need to show that gcd(min(a, b), |b - a|) = gcd(a, b). Let c = gcd(min(a, b), |b - a|). First we need to show that c|a, b. Like what we did above for the iterative algorithm, a and b are min(a, b) and min(a, b) + |b - a| (which is which depends on which of a or b is larger). Since c = gcd(min(a, b), |b - a|), we know that min(a, b) = cx and |b - a| and |b - a| = cy for some x, y ∈ N. So a and b are cx and c(x + y) (not necessarily in that order). Thus c|a, b.
Now assume k|a, b. Then k|min(a, b) and k||b - a|. So k|c.
Thus, gcd(min(a, b), |b - a|) = gcd(a, b). □