Algorithm Correctness
(or how to prove programs are correct)

**Pre-condition of the algorithm:** a predicate that describes the initial state (before execution)

**Post-condition of the algorithm:** a predicate that describes the final state (after execution)

---

The algorithm is correct if it can be proved that if the pre-condition is true, the post-condition must be true.

Pre-condition ⇒ Post-condition

---

**Example 1:**

program mystery
begin
    Pre-condition: \( a = 3 \quad b = 5 \)
end
Post-condition: \( a = 5 \quad b = 3 \)

**Example 2:**

\[
\begin{align*}
x &= 2 \\
z &= x + y \\
\text{if} \ (y > 0) \\
& \quad z = z + 1 \\
\text{else} \\
& \quad z = 0
\end{align*}
\]

Pre-condition: \( y = 6 \)
Post-condition: \( z = 9 \)
Programs may be sliced into sequential parts and assertions inserted

```
program mystery2
begin
  Segment 1
  Segment 2
  Segment 3
  .
  .
  Segment n
end
```

What about loops?
Consider a while loop with Pre-Condition and Post-Condition and guard G.

To prove that the loop is “correct” do the following:

1. Write a loop invariant that “captures” the purpose of the loop; call it I(n).  (Difficult!!) I(n) is a predicate that is true after n iterations of the loop.

2. Show that the pre-condition implies the loop invariant I(0).  (Basis Step)

3. Show that if I(k) and guard G are true, then I(k+1) is true.  (Inductive Step) In other words, if I(n) is true for n = k and the guard G is still true, then it follows that I(n) is true for n = k+1.

4. Show that guard G eventually becomes false (looping stops).

5. If the loop iterates a maximum of N times, show that I(N) implies the post-condition.
Example 1

**Pre-condition:** m is a non-negative integer, x is real, Count = 0, Product = 0

```
while ( Count != m)
{
    Product = Product + x;
    Count = Count + 1;
}
```

**Post-condition:** Count = m and Product = m ⋅ x

---

1. Write loop invariant \( I(n) \): (Remember that \( n \) represents the number of times the loop has iterated)

2. Show that the pre-condition implies that \( I(0) \) is true.

3. **Inductive Step:** Show that if \( I(k) \) and \( G \) are true, \( I(k+1) \) is true.
   
   Assume \( G \): and \( I(k): \) Product = \( k \cdot x \) and Count = \( k \) are true.
   
   Show \( I(k+1): \) Product = \( (k+1) \cdot x \) and Count = \( k + 1 \) is true.

4. Show that \( G \) will eventually be false.

5. Show that the post-condition follows when the loop stops.
Example 2.

**Pre-condition:** Max = A[0]; Count = 0

```c
while ( Count < 10)
    { Count = Count + 1;
        if ( A[Count] > Max )
            Max = A[Count];
    }
```

**Post-condition:** Max contains the largest value in A[0]..A[10]; Count = 10

1) Write loop invariant I(n):

2) Show that the pre-condition implies that I(0) is true.

3) Show I(k) and G imply I(k+1).
   
   I(k):
   
   I(k+1):

4) Show that G will eventually be false.

5) Show the post-condition follows when the loop stops