# Application: Correctness of Algorithms

# Definitions:

Consider an algorithm that is designed to produce a certain final state from a certain initial state. Both the initial and final states can be expressed as predicates involving the input and output variables.

***pre-condition***

Often the predicate describing the initial state is called the pre-condition for the algorithm, and

***post-condition***

the predicate describing the final state is called the *post-condition* for the algorithm.

# Example:

Algorithm to compute a product of nonnegative integers

**Pre-condition:** The input variables $m$ and $n$ are nonnegative integers.

**Post-condition:** The output variable $p$ equals $mn$.

# Definition:

A ***loop invariant*** is a predicate with domain a set of integers, which satisfies the condition:

For each iteration of the loop, if the predicate is true before the iteration, then it is true after the iteration.

Example: show that if the predicate is true before entry to the loop, then it is also true after exit from the loop.

loop:

while ($m \geq 0$ and $m \leq 100$)

$m := m + 1$

$n := n - 1$

end while

predicate: $m + n = 100$

# Example: show that if the predicate is true before entry to the loop, then it is also true after exit from the loop.

loop:

while ($m \geq 0$ and $m \leq 100$)

$m := m + 1$

$n := n - 1$

end while

predicate: $m + n = 100$

Let $m_{old}, n_{old}$ be the values of the algorithm variables before the entry to the loop.
Also assume that the given predicate is true for these values of the algorithm variables, that is

$$m_{old} + n_{old} = 100$$

Now let $m_{new}, n_{new}$ be the values of the algorithm variables after exiting from the loop. Then

$$m_{new} := m_{old} + 1$$

$$n_{new} := n_{old} - 1$$

The sum of the new values of the variables will be

$$m_{new} + n_{new}$$
$$= (m_{old} + 1) + (n_{old} - 1)$$
$$= 100$$

Therefore, the predicate is true after exit from the loop.

# Definition:

A loop is defined as ***correct*** with respect to its pre- and post-conditions if, and only if, whenever

(a) the algorithm variables satisfy the pre-condition for the loop and

(b) the loop terminates after a finite number of steps,

(c) the algorithm variables satisfy the post-condition for the loop.

Establishing the correctness of a loop uses the concept of loop invariant.

If the predicate satisfies the following two additional conditions, the loop will be correct *with respect to it pre- and post-conditions*:

1. It is true before the first iteration of the loop.
2. If the loop terminates after a finite number of iterations, the truth of the loop invariant ensures the truth of the post-condition of the loop.

# Loop Invariant Theorem

Let a while loop with guard G be given, together with pre- and post-conditions that are predicates in the algorithm variables. Also let a predicate $I(n)$, called the loop invariant, be given. If the following four properties are true, then the loop is correct with respect to its pre- and post-conditions.

- **Basis Property:** The pre-condition for the loop implies that $I(0)$ is true before the first iteration of the loop.
- **Inductive Property:** For all integers $k \geq 0$, if the guard G and the loop invariant $I(k)$ are both true before an iteration of the loop, then $I(k+1)$ is true after iteration of the loop.
- **Eventual Falsity of Guard:** After a finite number of iterations of the loop, the guard G becomes false.
- **Correctness of the Post-Condition:** If $N$ is the least number of iterations after which G is false and $I(N)$ is true, then the values of the algorithm variables will be as specified in the post-condition of the loop.

# Example:

[Pre-condition: $m$ is a nonnegative integer, $x$ is a real number, $i = 0$, and $exp = 1$.]

while ($i \neq m$)
$exp := exp \cdot x$
$i := i + 1$
end while

[Post-condition: $exp = x^m$]

loop invariant: $I(n)$ is "$exp = x^n$ and $i = n$."

Use the loop invariant theorem to prove that the while loop is correct with respect to the given pre- and post-conditions.

[Pre-condition: $m$ is a nonnegative integer, $x$ is a real number, $i = 0$, and $exp = 1$.]

while ($i \neq m$)
$exp := exp \cdot x$
$i := i + 1$
end while

[Post-condition: $exp = x^m$]

$I(n)$: $exp = x^n$ and $i = n$

**Basis Property:** The pre-condition for the loop implies that $I(0)$ is true before the first iteration of the loop.

Pre-condition suggests that the algorithm variable $exp$ has the value 1 and $i = 0$.

When $n = 0$, $I(0)$ is $exp = x^0 = 1$ and $i = 0$, which is in accordance with the pre-condition.

Therefore, $I(0)$ is true before the first iteration of the loop.

[Pre-condition: $m$ is a nonnegative integer, $x$ is a real number, $i = 0$, and $exp = 1$.]

while ($i \neq m$)

$exp := exp \cdot x$

$i := i + 1$

end while

[Post-condition: $exp = x^m$]

$I(n)$: $exp = x^n$ and $i = n$

**Inductive Property:** For all integers $k \geq 0$, if the guard G and the loop invariant $I(k)$ are both true before an iteration of the loop, then $I(k+1)$ is true after iteration of the loop.

Let $k$ be an arbitrary but particular integer ≥0 such that the guard G and the loop invariant $I(k)$ are both true before an iteration of the loop.

This means that

$$exp_{old} = x^k \text{ and } i_{old} = k \text{ and}$$

$$i_{old} \neq m \text{ or } i_{old} < m.$$

then after $(k+1)$th iteration of the loop, we get

$$\exp_{new} = \exp_{old}. x = x^{k+1},$$

$$i_{new} = i_{old} + 1 = k + 1$$

Which implies that $I(k+1)$ is true after the next iteration of the loop.

[Pre-condition: $m$ is a nonnegative integer, $x$ is a real number, $i = 0$, and $exp = 1$.]

while ($i \neq m$)
$exp := exp \cdot x$
$i := i + 1$
end while

[Post-condition: $exp = x^m$]

$I(n)$: $exp = x^n$ and $i = n$

**Eventual Falsity of Guard:** After a finite number of iterations of the loop, the guard G becomes false.

After $m$ number of iterations of the loop, the guard G becomes false.

[Pre-condition: $m$ is a nonnegative integer, $x$ is a real number, $i = 0$, and $exp = 1$.]

while ($i \neq m$)
  $exp := exp \cdot x$
  $i := i + 1$
end while

[Post-condition: $exp = x^m$]

$I(n)$: $exp = x^n$ and $i = n$

**Correctness of the Post-Condition:** If $N$ is the least number of iterations after which G is false and $I(N)$ is true, then the values of the algorithm variables will be as specified in the post-condition of the loop.

Since $m$ is the least number of iterations after which G is false and $I(m)$ is true. This means, $exp = x^m$ and $i = m$ which is as specified in the post-condition of the loop.