**Program State**

At any state of computation, we may want to evaluate a predicate (logical statement involving constants and values of program variables) describing the state; for instance, \( x := 1; \) \( \{ x = 1 \} \) will evaluate to `TRUE`; if \( A[1..3] \) is an array with entries 9, 5, 2, then the predicate \( \{ \forall i, j : 1 \leq i \leq j \leq 3 : A[i] \leq A[j] \} \) (we may call it ”Sorted(A, 1, 3)”) will be true.

We will treat program segments as “predicate transformers” in a notation \( \{ P_1 \} S \{ P_2 \} \) meaning

"if program segment \( S \) is executed in the state when \( P_1 \) is true then, upon termination, the state of computation will be such that \( P_2 \) will be true."

(If we don’t discuss termination, this is considered a “partial correctness” argument.)

**Loop Invariant**

Loop \( L: \) while \( C \) do \( B \)

We want to show \( \{ P \} \) while \( C \) do \( B \) \( \{ Q \} \)

where

\( P \) the initial state

\( Q \) the goal (final) state

\( C \) the condition

\( B \) the body of the loop

**Partial Correctness**

\( I \) is an invariant of \( L \) if \( \{ C \land I \} B \{ I \} \)

\( I \) is a useful invariant (for the purpose of showing \( \{ P \} L \{ Q \} \)) if \( P \Rightarrow I \) (”the base case of induction”) and \( I \land \neg C \Rightarrow Q \).

**Termination (Total Correctness)**

\( t \) is a termination function (integer-valued) if \( \{ (t = t_0) \land C \land I \} B \{ t < t_0 \} \)

and \( t \leq 0 \Rightarrow \neg C \).
HeapSort

Consider the following program segment purporting to sort in non-increasing manner elements of an array $A[1..n]$ ($n > 0$):

Heapify($A, 1, n$); \{Heap[$A[1..n]$]\}

$i := n$;
\{P\}
while $i > 1$ do
begin swap($A, 1, i$); $i := i - 1$; sift($A, 1, i$) end;
\{Q\}

where
$P : (n \geq 1) \land (A = a) \land $ Heap($A, 1, n$)
$Q : Sorted(A, 1, n) \land (1 \leq i \leq n)$

A useful invariant is
$I : Perm(A, a) \land $ Heap($A, 1, i$) \land (1 \leq i \leq n) \land
(\forall p, q : (i \leq p \leq n) \land (1 \leq q \leq n) : q \leq p \Rightarrow A[p] \leq A[q]$)

Exponentiation

To raise an integer $x$ to a non-negative power $y$, we can do better than multiplying $x \cdot y$ times (the algorithm is patterned on one that is sometimes called “Russian Peasant’s multiplication: $p \cdot q$ is (almost, when the division gives an integer result) $2p \cdot q/2$.)

$a := x$; $b := y$; $c := 1$;
while $b > 0$ do
begin
while even($b$) do begin $a := a \cdot a$; $b := b \div 2$ end;
$b := b - 1$; $c := c \cdot a$
end

The invariant of the inner loop is $I_{inn} : a^b = d$; some constant $d_1$
for the outer loop we have $I_{out} : c \cdot a^b = d_2$; some constant $d_2$. 

2