CS 281
Prof. Alex Pothen
Webbook:
www.cs.odu.edu/~webbook
www.cs.odu.edu/~pothen/Courses/CS281
CLASS NOW IN CONSTANT 120
Wed Oct 21, 1998
Lecture 15
Sec. 3.2 Induction, Sec 3.5
Program Correctness
Next Lecture: Sec 3.4 Recursive Algorithms
Correctness of Programs

• To prove that a program segment $S$ does what it is intended to do, we need:
  an initial assertion $p$, true of the input;
  a final assertion $q$, true of the output.

• Hoare triple: $p \{S\} q$
  If $p$ is true for the input, and the program $S$ terminates, then $q$ is true for the output.
Loop Invariants

• A technique for showing that a loop does what it is claimed to do.

• A loop invariant is a statement $S$ that is true each time we begin to execute the loop. We prove that $S$ is true by induction on the loop index (or some variable related to the loop index).

• Find the loop invariant by considering what happens as the loop is executed a few times.

• For loops terminate when the loop index becomes greater than the final value. We need to show that while loops terminate.
Multiplication

Consider the problem of multiplying two positive integers $m$, and $n$. We can do this by adding $m$ to itself $n$ times:

$$m + m + m + \ldots + m.$$  
$n$ times

(1) $\text{index} := 0; \quad \text{product} := 0;$
(2) while $\text{index} < n$ do
(3) $\text{index} := \text{index} + 1;$
(4) $\text{product} := \text{product} + m;$
   od

Loop Invariant:
When $\text{index}$ has the value $k$ (in statement (2)), $\text{product}$ contains the value $m \times k$.

Basis: $\text{index} = 0$: true since $\text{product}$ is set to zero in (1).
Multiplication II

Ind. Hyp. Assume when $index = (k - 1)$ in statement (2), that $product$ contains the value $m \times (k - 1)$.

Ind. step: Show that when $index = k$ in statement (2), that $product$ contains $m \times k$.

At the beginning of the loop, $index = (k - 1)$. As a result of statement (3), $index$ is equal to $k$; by statement (4), $product$ is incremented by $m$. Hence the new value of $product$ is $m \times (k - 1) + m = m \times k$.

When $index$ is equal to $n$ in statement (2), then the algorithm terminates with $product$ equal to $m \times n$. 
Selection Sort Algorithm

procedure Selection sort

for $step := 1$ to $(n - 1)$ do
    \{ $A[1:step - 1]$ is sorted at begin. of this loop; \}
    \{ $A[1:step]$ is sorted at end of this loop. \}

    \{ 1. select smallest element in $A[step:n]$ \}
    small := step;
    for $index := step + 1$ to $n$ do
            small := index;
        fi
    rof
    \{ 2. move this element to posn. of $A[step]$ \}
rof
end
Inner loop of Selection Sort

\{ Select smallest element in $A[step : n]$ \}
(1) \hspace{1em} small := step;
(2) \hspace{1em} for \hspace{1em} index := step + 1 \text{ to } n \hspace{1em} do
(3) \hspace{1em} \hspace{1em} if \hspace{1em} A[index] < A[small] \hspace{1em} then
(4) \hspace{1em} \hspace{1em} \hspace{1em} small := index;
(5) \hspace{1em} \hspace{1em} fi
rof

Loop Invariant:
When $index$ is set to $k$ (in statement (2)),
small is the index of a smallest element in

Basis: \hspace{1em} index = step + 1. small is the index of $A[step]$ after statement (1).
**Inner loop of Selection Sort II**

Ind. Hyp.: Assume true for $step \leq index \leq (k-1)$, for $k \leq n$. Prove for $index = k$.

Ind. step: At the beginning of the loop, $index = k - 1$, and $small$ contains the index of a smallest element in $A[step : k-2]$. In statements (3) and (4), we compare $A[k-1]$ with $A[small]$ and replace $small$ with the index of the smaller of these two elements. After executing the loop, when $index$ is set to $k$ in statement (2), $small$ contains the index of a smallest element in $A[step : k-1]$. When $index$ is incremented to $n+1$, then the for loop terminates with a smallest element in $A[step : n]$. 
Selection Sort

(1) for \( step := 1 \) to \((n - 1)\) do
(2) \( small \) is index of a smallest element in \( A[step : n] \);
(3) swap \( A[small] \) with \( A[step] \);
end

Loop Invariant:
When \( step \) is set to \( k \) (in statement (1)),
(a). \( A[1 : k - 1] \) is in nondecreasing order;
(b). every element in \( A[k : n] \) is at least as big as \( A[k - 1] \).

Basis: \( step = 1 \): true since \( A[1 : 0] \) is empty.
Selection Sort II

Ind. Hyp. Assume when \( step = k - 1 \) in statement (1), that \( A[1 : k - 2] \) is sorted, and that every element in \( A[k - 1 : n] \) is at least as great as \( A[k - 2] \).
Selection Sort III

Ind. step: At the beginning of the loop, \( step = k \). As a result of statement(2), \( small \) contains the index of a smallest element in \( A[k : n] \). Statement (3) then stores the smallest element in \( A[k] \). Since this element was at least as big as \( A[k-1] \), we have that \( A[1 : k] \) is non-decreasing. Further, since \( A[k] \) now contains a smallest element in \( A[k : n] \), every element in \( A[k+1 : n] \) is at least as big as \( A[k] \). When \( step \) is incremented to \( n \), then the algorithm terminates with \( A[1 : n-1] \) sorted. But this implies that \( A[1 : n] \) is sorted as well.