

We are asked to calculate the maximum segment sum and its start and end indices. The program for calculating the sum is

```

[[ con  $N : \text{int } \{0 \leq N\}$ ;  $f : \text{array}[0..N)$  of int;
   var  $r, n, s : \text{int}$ ;
    $n, r, s := 0, 0, 0 \{ \text{inv} : (1) \wedge (2) \wedge (3) \}$ 
; do  $n \neq N \rightarrow$ 
   if  $0 \leq s + f.n \rightarrow s := s + f.n$ 
   []  $s + f.n \leq 0 \rightarrow s := 0$ 
   fi
; if  $s \leq r \rightarrow \text{skip}$ 
   []  $r \leq s \rightarrow r := s$ 
   fi
;  $n := n + 1$ 
od
]].

```

where

$$(1) \quad r = \langle \uparrow i, j : 0 \leq i \leq j \leq n : S.i.j \rangle$$

$$(2) \quad 0 \leq n \leq N$$

$$(3) \quad s = \langle \uparrow i : 0 \leq i \leq n : S.i.n \rangle$$

To calculate the indices we strengthen the invariant with the conjunction of

$$(4) \quad r = S.p.q$$

$$(5) \quad s = S.k.n$$

$$(6) \quad 0 \leq p \leq q \leq N$$

$$(7) \quad 0 \leq k \leq N$$

which we establish with $n, r, s, k, p, q := 0, 0, 0, 0, 0, 0$.

* * *

Our first task is to maintain the invariant $s = S.k.n$. There are three assignment statements which could conceivably violate this invariant. We investigate each in turn, working backwards. First we observe

$$\begin{aligned}
& \llbracket \\
& \triangleright \\
& \quad (n := n + 1).(s = S.k.n) \\
& = \quad \{ \text{substitution} \} \\
& \quad s = S.k.(n + 1) \\
& \rrbracket
\end{aligned}$$

This gives us the postcondition of the other two commands. As they are at the beginning of the loop body, we must ensure that their precondition is $s = S.k.n$.

$$\begin{aligned}
& \llbracket \text{Cxt: } s = S.k.n \\
& \triangleright \\
& \quad (s := s + f.n).(s = S.k.(n + 1)) \\
& = \quad \{ \text{substitution} \} \\
& \quad s + f.n = S.k.(n + 1) \\
& = \quad \{ \text{arithmetic} \} \\
& \quad s = S.k.(n + 1) - f.n \\
& = \quad \{ \text{definition of } S \} \\
& \quad s = S.k.n \\
& = \quad \{ \text{context} \} \\
& \quad \text{true} \\
& \rrbracket
\end{aligned}$$

and

$$\begin{aligned}
& \llbracket \text{Cxt: } s = S.k.n \\
& \triangleright \\
& \quad (s := 0).(s = S.k.(n + 1)) \\
& = \quad \{ \text{substitution} \} \\
& \quad 0 = S.k.(n + 1) \\
& = \quad \{ \text{assuming } k = n + 1 \} \\
& \quad \text{true} \\
& \rrbracket
\end{aligned}$$

The above calculation tells us that if we wish $s := 0$ to establish $s = S.k.(n + 1)$ we had better assign $n + 1$ to k . We now have a program of the form:

$$\begin{aligned}
& n, r, s, k, p, q := 0, 0, 0, 0, 0, 0 \\
& ; \text{do } n \neq N \rightarrow \\
& \quad \text{if } 0 \leq s + f.n \rightarrow s := s + f.n
\end{aligned}$$

```

    []  $s + f.n \leq 0 \rightarrow s, k := 0, n + 1$ 
  fi
; if  $s \leq r \rightarrow skip$ 
  []  $r \leq s \rightarrow r := s$ 
  fi
;  $n := n + 1$ 
od

```

Now we may deal with the invariant $r = S.p.q$.

[[*Cxt*: $s = S.k.(n + 1)$

▷

```

  ( $r := s$ ).( $r = S.p.q$ )
= { substitution }
   $s = S.p.q$ 
= { assuming  $p = k \wedge q = n + 1$  }
   $s = S.k.(n + 1)$ 
= { context }
  true

```

]]

Thus to establish $r = S.p.q$ we need to assign k to p and $n + 1$ to q resulting in the final program

```

 $n, r, s, k, p, q := 0, 0, 0, 0, 0, 0$ 
; do  $n \neq N \rightarrow$ 
  if  $0 \leq s + f.n \rightarrow s := s + f.n$ 
  []  $s + f.n \leq 0 \rightarrow s, k := 0, n + 1$ 
  fi
; if  $s \leq r \rightarrow skip$ 
  []  $r \leq s \rightarrow r, p, q := s, k, n + 1$ 
  fi
;  $n := n + 1$ 
od

```

31 December 2006