Answers to examination for 14 August, 1996

Problem 1 (20p). To prove the theorem we use the following lemma:

Lemma 1

$$m_1 \leq m_2 \leq m_3 \;\Rightarrow\; \left(\begin{array}{c} a[m_1 : m_2] \leq x \wedge a[m_2 : m_3] \leq x \\ = \\ a[m_1 : m_3] \leq x \end{array}\right)$$

Proof of lemma. **Assume** $m_1 \leq m_2$
Assume $m_2 \leq m_3$

$a[m_1 : m_2] \leq x \wedge a[m_2 : m_3] \leq x$
$=$ {Definition of $\leq$}
$(\forall i : m_1 \leq i < m_2 + 1 : a[i] \leq x) \wedge (\forall i : m_2 \leq i < m_3 + 1 : a[i] \leq x)$
$=$ {Arithmetics}
$(\forall i : m_1 \leq i < m_2 + 1 : a[i] \leq x) \wedge (\forall i : m_2 - 1 < i \leq m_3 : a[i] \leq x)$
$=$ {Definition of $\forall$}

$$\left(\begin{array}{c} (\forall i : m_1 \leq i < m_2 : a[i] \leq x) \\ \wedge \\ a[m_2] \leq x \\ \wedge \\ (\forall i : m_2 - 1 < i \leq m_3 : a[i] \leq x) \end{array}\right)$$

$=$ {Definition of $\forall$}

$$\left(\begin{array}{c} (\forall i : m_1 \leq i < m_2 : a[i] \leq x) \\ \wedge \\ a[m_2] \leq x \wedge a[m_2] \leq x \\ \wedge \\ (\forall i : m_2 < i \leq m_3 : a[i] \leq x) \end{array}\right)$$

$=$ {and-simplification}

$$\left(\begin{array}{c} (\forall i : m_1 \leq i < m_2 : a[i] \leq x) \\ \wedge \\ a[m_2] \leq x \\ \wedge \\ (\forall i : m_2 - 1 < i \leq m_3 : a[i] \leq x) \end{array}\right)$$

$=$ {Definition of $\forall$}
$(\forall i : m_1 \leq i < m_2 : a[i] \leq x) \wedge (\forall i : m_2 - 1 < i \leq m_3 : a[i] \leq x)$
$=$ {Arithmetics}
$(\forall i : m_1 \leq i < m_2 : a[i] \leq x) \wedge (\forall i : m_2 \leq i < m_3 + 1 : a[i] \leq x)$
$=$ {Conditional Substitution; Range Split and $m_1 \leq m_2 \leq m_3$}
$(\forall i : m_1 \leq i < m_3 + 1 : a[i] \leq x)$
$=$ {Definition of $\leq$}
$a[m_1 : m_3] \leq x \quad \Box$

Proof of theorem.

$$\textbf{Assume} \quad \left.\begin{array}{l} m_1 \le m_2 \\ m_2 \le m_3 \end{array}\right\} \text{by transitivity also } m_1 \le m_3$$
$$m_3 \le m_4$$
$$a[m_1 : m_3] \le x$$
$$a[m_2 : m_4] \le x$$

$a[m_1 : m_4] \le x$
= {Conditional Substitution; Lemma 1, $m_1 \le m_3$, and $m_3 \le m_4$}
$a[m_1 : m_3] \le x \wedge a[m_3 : m_4] \le x$
= {Conditional Substitution; Lemma 1, $m_1 \le m_2$, and $m_2 \le m_3$}
$a[m_1 : m_2] \le x \wedge a[m_2 : m_3] \le x \wedge a[m_3 : m_4] \le x$
= {and-simplification}
$a[m_1 : m_2] \le x \wedge a[m_2 : m_3] \le x \wedge a[m_2 : m_3] \le x \wedge a[m_3 : m_4] \le x$
= {Conditional Substitution; Lemma 1, $m_1 \le m_2$, and $m_2 \le m_3$}
$a[m_1 : m_3] \le x \wedge a[m_2 : m_3] \le x \wedge a[m_2 : m_3] \le x \wedge a[m_3 : m_4] \le x$
= {Conditional Substitution; Lemma 1, $m_1 \le m_2$, and $m_2 \le m_3$}
$a[m_1 : m_3] \le x \wedge a[m_2 : m_3] \le x \wedge a[m_3 : m_4] \le x$
= {Conditional Substitution; Lemma 1, $m_2 \le m_3$, and $m_3 \le m_4$}
$a[m_1 : m_3] \le x \wedge a[m_2 : m_4] \le x$
= {Assumption $a[m_1 : m_3] \le x$ and $a[m_2 : m_4] \le x$}
$T \wedge T$ = {and-simplification}
T □

Problem 2 (15p). To prove that $wp(\text{“}S\text{”}, R) = wp(\text{“}S'\text{”}, R)$ for any R we use the following lemma.

Lemma 2

$$(B_1 \wedge B_2 \Rightarrow wp(\text{“}S\text{”}, R)) = (B_1 \wedge B_2 \Rightarrow wp(\text{“}S_1\text{”}, R) \wedge wp(\text{“}S_2\text{”}, R)).$$

Proof of lemma.

$B_1 \wedge B_2 \Rightarrow wp(\text{“}S\text{”}, R)$
= {Definition of S and Alternative Command Theorem}
$B_1 \wedge B_2 \Rightarrow (B_1 \vee B_2) \wedge (B_1 \Rightarrow wp(\text{“}S_1\text{”}, R)) \wedge (B_2 \Rightarrow wp(\text{“}S_2\text{”}, R))$
= {Replace with T}
$B_1 \wedge B_2 \Rightarrow (T \vee T) \wedge (T \Rightarrow wp(\text{“}S_1\text{”}, R)) \wedge (T \Rightarrow wp(\text{“}S_2\text{”}, R))$
= {Left identity of $\Rightarrow$}
$B_1 \wedge B_2 \Rightarrow (T \vee T) \wedge wp(\text{“}S_1\text{”}, R) \wedge wp(\text{“}S_2\text{”}, R)$
= {or-simplification}
$B_1 \wedge B_2 \Rightarrow T \wedge wp(\text{“}S_1\text{”}, R) \wedge wp(\text{“}S_2\text{”}, R)$
= {and-simplification}
$B_1 \wedge B_2 \Rightarrow wp(\text{“}S_1\text{”}, R) \wedge wp(\text{“}S_2\text{”}, R)$ □

Proof of theorem. **Assume** $B_2 \Rightarrow B_1$

$wp(\text{“}S'\text{”}, R)$
$= \{\text{Definition of } S' \text{ and Alternative Command Theorem}\}$
$(B_1 \vee B_2) \wedge (B_1 \Rightarrow wp(\text{“}S_1\text{”}, R)) \wedge (B_2 \Rightarrow wp(\text{“}S\text{”}, R))$
$= \{\text{and-simplification}\}$
$(B_1 \vee B_2) \wedge (B_1 \Rightarrow wp(\text{“}S_1\text{”}, R)) \wedge (B_2 \wedge T \Rightarrow wp(\text{“}S\text{”}, R))$
$= \{\text{Assumption } B_2 \Rightarrow B_1\}$
$(B_1 \vee B_2) \wedge (B_1 \Rightarrow wp(\text{“}S_1\text{”}, R)) \wedge (B_2 \wedge (B_2 \Rightarrow B_1) \Rightarrow wp(\text{“}S\text{”}, R))$
$= \{\text{Elimination of } \Rightarrow\}$
$(B_1 \vee B_2) \wedge (B_1 \Rightarrow wp(\text{“}S_1\text{”}, R)) \wedge (B_2 \wedge B_1 \Rightarrow wp(\text{“}S\text{”}, R))$
$= \{\text{Lemma 2}\}$
$(B_1 \vee B_2) \wedge (B_1 \Rightarrow wp(\text{“}S_1\text{”}, R)) \wedge (B_2 \wedge B_1 \Rightarrow wp(\text{“}S_1\text{”}, R) \wedge wp(\text{“}S_2\text{”}, R))$
$= \{\text{Distributivity of } \wedge \text{ over } \Rightarrow\}$
$(B_1 \vee B_2) \wedge (B_1 \Rightarrow wp(\text{“}S_1\text{”}, R)) \wedge (B_2 \wedge B_1 \Rightarrow wp(\text{“}S_2\text{”}, R)) \wedge (B_2 \wedge B_1 \Rightarrow wp(\text{“}S_1\text{”}, R))$
$= \{\text{Proof by Cases}\}$
$(B_1 \vee B_2) \wedge (B_1 \vee B_2 \wedge B_1 \Rightarrow wp(\text{“}S_1\text{”}, R)) \wedge (B_2 \wedge B_1 \Rightarrow wp(\text{“}S_2\text{”}, R))$
$= \{\text{or-simplification}\}$
$(B_1 \vee B_2) \wedge (B_1 \Rightarrow wp(\text{“}S_1\text{”}, R)) \wedge (B_2 \wedge B_1 \Rightarrow wp(\text{“}S_2\text{”}, R))$
$= \{\text{Introduction of } \Rightarrow\}$
$(B_1 \vee B_2) \wedge (B_1 \Rightarrow wp(\text{“}S_1\text{”}, R)) \wedge (B_2 \wedge (B_2 \Rightarrow B_1) \Rightarrow wp(\text{“}S_2\text{”}, R))$
$= \{\text{Assumption}\}$
$(B_1 \vee B_2) \wedge (B_1 \Rightarrow wp(\text{“}S_1\text{”}, R)) \wedge (B_2 \wedge T \Rightarrow wp(\text{“}S_2\text{”}, R))$
$= \{\text{and-simplification}\}$
$(B_1 \vee B_2) \wedge (B_1 \Rightarrow wp(\text{“}S_1\text{”}, R)) \wedge (B_2 \Rightarrow wp(\text{“}S_2\text{”}, R))$
$= \{\text{Alternative Command Theorem and definition of } S\}$
$wp(\text{“}S\text{”}, R)$ $\square$

Problem 3 (20p).

Using the alternative command theroem we have to prove:

1. $$(y = x - 2) \vee (y = x + 2) \Rightarrow (y \leq x) \vee (x \leq y).$$

   Proof.
   $(y = x - 2) \vee (y = x + 2) \Rightarrow (y \leq x) \vee (x \leq y)$
   $= \{\text{Arithmetics}\}$
   $(y = x - 2) \vee (y = x + 2) \Rightarrow T$
   $= \{\text{Left zero of } \Rightarrow\}$
   T

2. $$\left(\begin{array}{c} ((y = x - 2) \vee (y = x + 2)) \wedge (y \leq x) \\ \Rightarrow \\ wp(\text{“}x, y := y + 3, x - 1\text{”}, (y = x - 2) \vee (y = x + 2)) \end{array} \right)$$

Proof.

$$
\begin{array}{l}
\left(\begin{array}{c}((y = x-2) \vee (y = x+2)) \wedge (y \leq x) \\ \Rightarrow \\ wp\,(\text{“}x, y := y+3, x-1\text{”}, (y = x-2) \vee (y = x+2))\end{array}\right) \\
= \{\text{Definition of assignment}\} \\
\left(\begin{array}{c}((y = x-2) \vee (y = x+2)) \wedge (y \leq x) \\ \Rightarrow \\ (x-1 = y+3-2) \vee (x-1 = y+3+2)\end{array}\right) \\
= \{\text{Distributivity of } \vee \text{ over } \wedge\} \\
\left(\begin{array}{c}((y = x-2) \wedge (y \leq x) \vee (y = x+2) \wedge (y \leq x)) \\ \Rightarrow \\ (x-1 = y+3-2) \vee (x-1 = y+3+2)\end{array}\right) \\
= \{\text{Substitution}\} \\
\left(\begin{array}{c}((y = x-2) \wedge (y \leq x) \vee (y = x+2) \wedge (x+2 \leq x)) \\ \Rightarrow \\ (x-1 = y+3-2) \vee (x-1 = y+3+2)\end{array}\right) \\
= \{\text{Arithmetics}\} \\
((y = x-2) \wedge (y \leq x) \vee F \;\Rightarrow\; (x-1 = y+1) \vee (x-1 = y+5) \\
= \{\text{or-simplification}\} \\
((y = x-2) \wedge (y \leq x) \;\Rightarrow\; (x-1 = y+1) \vee (x-1 = y+5) \\
= \{\text{Substitution}\} \\
((y = x-2) \wedge (y \leq x) \;\Rightarrow\; (x-1 = x-2+1) \vee (x-1 = x-2+5) \\
= \{\text{Arithmetics}\} \\
((y = x-2) \wedge (y \leq x) \;\Rightarrow\; (x-1 = x-1) \vee (x-1 = x+3) \\
= \{\text{Identity}\} \\
((y = x-2) \wedge (y \leq x) \;\Rightarrow\; T \vee (x-1 = x+3) \\
= \{\text{or-simplification}\} \\
((y = x-2) \wedge (y \leq x) \;\Rightarrow\; T \\
= \{\text{Left zero of } \Rightarrow\} \\
T \quad \square
\end{array}
$$

3.

$$
\begin{array}{c}((y = x-2) \vee (y = x+2)) \wedge (x \leq y) \\ \Rightarrow \\ wp\,(\text{“}y := 4; x := 6\text{”}, (y = x-2) \vee (y = x+2))\end{array}
$$

Proof.

$$
\begin{array}{l}
\left(\begin{array}{c}((y = x-2) \vee (y = x+2)) \wedge (x \leq y) \\ \Rightarrow \\ wp\,(\text{“}y := 4; x := 6\text{”}, (y = x-2) \vee (y = x+2))\end{array}\right) \\
= \{\text{Definition of sequential composition}\} \\
\left(\begin{array}{c}((y = x-2) \vee (y = x+2)) \wedge (x \leq y) \\ \Rightarrow \\ wp\,(\text{“}y := 4\text{”}, wp\,(\text{“}x := 6\text{”}, (y = x-2) \vee (y = x+2)))\end{array}\right) \\
= \{\text{Definition of assignment}\} \\
\left(\begin{array}{c}((y = x-2) \vee (y = x+2)) \wedge (x \leq y) \\ \Rightarrow \\ wp\,(\text{“}y := 4\text{”}, (y = 6-2) \vee (y = 6+2))\end{array}\right) \\
= \{\text{Definition of assignment}\} \\
((y = x-2) \vee (y = x+2)) \wedge (x \leq y) \;\Rightarrow\; (4 = 6-2) \vee (4 = 6+2) \\
= \{\text{Arithmetics}\} \\
((y = x-2) \vee (y = x+2)) \wedge (x \leq y) \;\Rightarrow\; T \vee F \\
= \{\text{or-simplification}\} \\
((y = x-2) \vee (y = x+2)) \wedge (x \leq y) \;\Rightarrow\; T \\
= \{\text{Left zero of } \Rightarrow\} \\
T
\end{array}
$$

Problem 4 (20p). To prove that the invariant is invariant we prove that the invariant holds for each of the alternative guarded commands. They are:

1. $Q \wedge \neg \mathit{flag} \;\Rightarrow\; wp\,(\text{“}\mathit{flag} := T; \mathit{proc}_1 := T\text{”}, Q)$
2. $Q \wedge \neg \mathit{flag} \;\Rightarrow\; wp\,(\text{“}\mathit{flag} := T; \mathit{proc}_2 := T\text{”}, Q)$
3. $Q \wedge \mathit{flag} \wedge \mathit{proc}_1 \;\Rightarrow\; \textbf{skip}$
4. $Q \wedge \mathit{flag} \wedge \mathit{proc}_2 \;\Rightarrow\; \textbf{skip}$
5. $Q \wedge \mathit{flag} \wedge \mathit{proc}_1 \;\Rightarrow\; wp\,(\text{“}\mathit{proc}_1 := F; \mathit{flag} := F\text{”}, Q)$
6. $Q \wedge \mathit{flag} \wedge \mathit{proc}_2 \;\Rightarrow\; wp\,(\text{“}\mathit{proc}_2 := F; \mathit{flag} := F\text{”}, Q)$

We only prove case 1, 3, and 5 since the other cases are similar.

Case 1 **Assume** $\mathit{proc}_1 \vee \mathit{proc}_2 \;\Rightarrow\; \mathit{flag}$
Assume $\neg(\mathit{proc}_1 \wedge \mathit{proc}_2)$
Assume $\neg \mathit{flag}$

$wp\,(\text{“}\mathit{flag} := T; proc_1 := T\text{”}, (proc_1 \vee proc_2 \Rightarrow \mathit{flag}) \wedge \neg(proc_1 \wedge proc_2))$
$= \{\text{Definition of sequential composition}\}$
$wp\,(\text{“}\mathit{flag} := T\text{”}, wp\,(\text{“}proc_1 := T\text{”}, (proc_1 \vee proc_2 \Rightarrow \mathit{flag}) \wedge \neg(proc_1 \wedge proc_2)))$
$= \{\text{Definition of assignment}\}$
$wp\,(\text{“}\mathit{flag} := T\text{”}, (T \vee proc_2 \Rightarrow \mathit{flag}) \wedge \neg(T \wedge proc_2))$
$= \{\text{Definition of assignment}\}$
$(T \vee proc_2 \Rightarrow T) \wedge \neg(T \wedge proc_2)$
$= \{\text{and-simplification}\}$
$(T \vee proc_2 \Rightarrow T) \wedge \neg proc_2$
$= \{\text{Left zero of } \Rightarrow\}$
$T \wedge \neg proc_2$
$= \{\text{and-simplification}\}$
$\neg proc_2$
$= \{\text{Introduction of } \Rightarrow\}$
$proc_2 \Rightarrow F$
$= \{\text{Negation of } F\}$
$proc_2 \Rightarrow \neg T$
$= \{\text{Assumption } \neg\mathit{flag}\}$
$proc_2 \Rightarrow \neg\neg \mathit{flag}$
$= \{\text{Negation}\}$
$proc_2 \Rightarrow \mathit{flag}$
$= \{\text{and-simplification}\}$
$T \wedge (proc_2 \Rightarrow \mathit{flag})$
$= \{\text{Assumption } proc_1 \vee proc_2 \Rightarrow \mathit{flag}\}$
$(proc_1 \vee proc_2 \Rightarrow \mathit{flag}) \wedge (proc_2 \Rightarrow \mathit{flag})$
$= \{\text{Proof by cases}\}$
$proc_1 \vee proc_2 \vee proc_2 \Rightarrow \mathit{flag}$
$= \{\text{or-simplification}\}$
$proc_1 \vee proc_2 \Rightarrow \mathit{flag}$
$= \{\text{Assumption } proc_1 \vee proc_2 \Rightarrow \mathit{flag}\}$
$T \quad \square$

Case 3 **Assume** $proc_1 \vee proc_2 \Rightarrow \mathit{flag}$
Assume $\neg(proc_1 \wedge proc_2)$
Assume $\mathit{flag} \wedge proc_1$

$wp\,(\text{“}\mathbf{skip}\text{”}, (proc_1 \vee proc_2 \Rightarrow \mathit{flag}) \wedge (\neg(proc_1 \wedge proc_2)))$
$= \{\text{Definition of } \mathbf{skip}\}$
$(proc_1 \vee proc_2 \Rightarrow \mathit{flag}) \wedge (\neg(proc_1 \wedge proc_2))$
$= \{\text{Assumption } proc_1 \vee proc_2 \Rightarrow \mathit{flag}\}$
$T \wedge (\neg(proc_1 \wedge proc_2))$
$= \{\text{Assumption } \neg(proc_1 \wedge proc_2)\}$
$T \wedge T$
$= \{\text{and-simplification}\}$
T

Case 5 **Assume** $\neg(proc_1 \wedge proc_2)$
Assume $\neg(proc_1 \wedge proc_2)$
Assume *flag*
Assume $proc_1$

$wp\,(\text{“}proc_1 := F; \mathit{flag} := F\text{”}, (proc_1 \vee proc_2 \Rightarrow \mathit{flag}) \wedge (\neg(proc_1 \wedge proc_2)))$
$= \{\text{Definition of sequential composition}\}$
$wp\,(\text{“}proc_1 := F\text{”}, wp\,(\text{“}\mathit{flag} := F\text{”}, (proc_1 \vee proc_2 \Rightarrow \mathit{flag}) \wedge (\neg(proc_1 \wedge proc_2))))$
$= \{\text{Definition of assignment}\}$
$wp\,(\text{“}proc_1 := F\text{”}, (proc_1 \vee proc_2 \Rightarrow F) \wedge (\neg(proc_1 \wedge proc_2)))$
$= \{\text{Definition of assignment}\}$
$(F \vee proc_2 \Rightarrow F) \wedge (\neg(F \wedge proc_2))$
$= \{\text{or-simplification}\}$
$(proc_2 \Rightarrow F) \wedge (\neg(F \wedge proc_2))$
$= \{\text{and-simplification}\}$
$(proc_2 \Rightarrow F) \wedge \neg F$
$= \{\text{Definition of } F\}$
$(proc_2 \Rightarrow F) \wedge \neg\neg T$
$= \{\text{Negation}\}$
$(proc_2 \Rightarrow F) \wedge T$
$= \{\text{and-simplification}\}$
$proc_2 \Rightarrow F$
$= \{\text{Elimination of } \Rightarrow\}$
$\neg proc_2$
$= \{\text{and-simplification}\}$
$\neg(T \wedge proc_2)$
$= \{\text{Assumption } proc_1\}$
$\neg(proc_1 \wedge proc_2)$
$= \{\text{Assumption } \neg(proc_1 \wedge proc_2)\}$
T □

Problem 5 (20p). The steps are as follows:

1. *Develop pre- and post-conditions!*

   We first state the post condition of the problem. Since we want the result of evaluating the polynom stored in a variable r we state the post condition as

   $$R\text{: } r = eval(p, x)$$

   which by expanding according to the definition of $eval(p, x)$ becomes

   $$R\text{: } r = (\Sigma i : 0 \leq i < n + 1 : p[i] * x^{n-i}).$$

   As our precondition it is convenient (but not necessary) to use $n \geq 0$. Hence we have

   $$Q\text{: } n \geq 0$$
   $$R\text{: } r = (\Sigma i : 0 \leq i < n + 1 : p[i] * x^{n-i}).$$

2. *Develop the invariant!*

   We develop an invariant from the post-condition. By replacing the constant n with a variable and place suitable restrictions on the variable we get the invariant

   $$P\colon\ r = (\Sigma i : 0 \leq i < j+1 : p[i] * x^{j-i}) \wedge (0 \leq j \leq n).$$

3. *Find a guard B such that $P \wedge \neg B \Rightarrow R$.*

   As a free bonus we get that $\neg B\colon\ j = n$, hence we can use

   $$B\colon\ j \neq n$$

   as guard for the loop.

4. *Find bound function t.*

   Since we want $j = n$ in the end, and $j \leq n$ is in the invariant a suitable bound function is

   $$t\colon\ n - j + 1.$$

5. *Find statement S_t to decrease the bound function.*

   Only one alternative:

   $$S_t\colon\ j := j + 1.$$

   We could increase j with more, but since there is no obvious reason to why we should, we don't.

6. *Find statement S_{inv} such that $P \wedge B \Rightarrow wp\,(\text{“}S_{inv}; S_t\text{”}, P)$.*

   (a) We begin by computing $wp\,(\text{“}S_t\text{”}, P)$.

   $$\begin{aligned}
   & wp\,(\text{“}j := j+1\text{”}, (r = (\Sigma i : 0 \leq i < j+1 : p[i] * x^{j-i})) \wedge (0 \leq j \leq n)) \\
   =& \{\text{Definition of assignment}\} \\
   & (r = (\Sigma i : 0 \leq i < j+2 : p[i] * x^{(j+1)-i})) \wedge (0 \leq j+1 \leq n)) \\
   =& \{\text{Definition of } \Sigma\} \\
   & (r = (\Sigma i : 0 \leq i < j+1 : p[i] * x^{(j+1)-i}) + p[j+1] * x^{(j+1)-(j+1)}) \wedge (0 \leq j+1 \leq n)) \\
   =& \{\text{Arithmetics}\} \\
   & (r = x * (\Sigma i : 0 \leq i < j+1 : p[i] * x^{j-i}) + p[j+1] * x^0) \wedge (0 \leq j+1 \leq n) \\
   =& \{\text{Arithmetics}\} \\
   & (r = x * (\Sigma i : 0 \leq i < j+1 : p[i] * x^{j-i}) + p[j+1]) \wedge (0 \leq j+1 \leq n)
   \end{aligned}$$

   (b) We compare the above with P and see what is “missing”.
   The guard $j \neq n$ together with $j \leq n$ gives us that $0 \leq j+1 \leq n$.
   Regarding r we make the following observation:

   $$(r = x * \underbrace{(\Sigma i : 0 \leq i < j+1 : p[i] * x^{j-i})}_{\text{old } r} + p[j+1]) \wedge (0 \leq j \leq n).$$

   Hence we can use

   $$S_{inv}\colon\ r := x * r + p[j+1].$$

7. *Find initial statement S_{init} to establish the invariant.*

   This is simply $j := -1; r := 0$, or alternatively $j := 0; r := p[0]$.

Hence the complete program is:

$\{Q\colon\ n \geq 0\}$
$j, r := -1, 0;$
$\{\text{inv } P\colon\ r = (\Sigma i : 0 \leq i < j+1 : p[i] * x^{j-i}) \wedge (0 \leq j \leq n)\}$
$\{\text{bound } t\colon\ n - j + 1\}$
do $i \neq n \rightarrow r := x * r + p[j+1]; j := j + 1$ **od**
$\{R\colon\ r = eval(p, x)\}$