1.

The language \( L \) and functions \( R, A, \) and \( T \) defined below are the same as in Section 7.6.

\[
L \subseteq \{ a, b, \cdot \}^+
\]

1. \( \cdot \in L \)
2a. \( u \in L \Rightarrow au \in L \)
2b. \( u \in L \Rightarrow bu \in L \)
3. n. e.

<table>
<thead>
<tr>
<th>( A: L^2 \rightarrow L )</th>
<th>( R: L \rightarrow L )</th>
<th>( T: L^2 \rightarrow L )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. ( A(\cdot, v) = v )</td>
<td>( R(\cdot) = \cdot )</td>
<td>( T(\cdot, v) = v )</td>
</tr>
<tr>
<td>2a. ( A(au, v) = aA(u, v) )</td>
<td>( R(au) = A(R(u, a\cdot)) )</td>
<td>( T(au, v) = T(u, av) )</td>
</tr>
<tr>
<td>2b. ( A(bu, v) = bA(u, v) )</td>
<td>( R(bu) = A(R(u, b\cdot)) )</td>
<td>( T(bu, v) = T(u, bv) )</td>
</tr>
</tbody>
</table>

It is proved in the book that

\* **Proposition 7.6.** \( A \) is associative; that is, for all \( u, v, w \in L \), \( A(u, A(v, w)) = A(A(u, v), w) \).

\* **Proposition 7.9.** Assuming Proposition 7.8, below, \( R \) is self-cancelling; that is, for all \( u \in L \), \( R(R(u)) = u \).

Prove the following:

(a) **Proposition 7.7.** For all \( u \in L \), \( A(u, \cdot) = u \).

(b) **Proposition 7.8.** For all \( u, v \in L \), \( R(A(u, v) = A(R(v), R(u))) \).

(c) **Proposition 7.11.** For all \( u, v \in L \), \( T(u, v) = A((R(u), v)) \).

(d) **Proposition 7.10.** For all \( u \in L \), \( T(u, \cdot) = R(u) \).

(e) **Proposition 7.12.** For all \( u \in L \), \( T(T(u, \cdot), \cdot) = u \).

**Solution**

(a) **Proposition 7.7.** For all \( u \in L \), \( A(u, \cdot) = u \).

**Proof.** By induction on \( u \in L \).

**Base Case.** \( A(\cdot, \cdot) \overset{\text{A1}}{=} \cdot \).

**Induction:** Assume \( IH \equiv A(\cdot, \cdot) = u \).

\[
A(au, \cdot) \overset{\text{A2a}}{=} a A(u, \cdot) \overset{IH}{=} au
\]
Similarly, for $bu$,

$$A(bu, \bullet) \overset{A.2b}{=} b A(u, \bullet) \overset{IH}{=} bu$$

(b) **Proposition 7.8.** For all $u, v \in L$, $R(A(u, v) = A(R(v), R(u))$. **Proof.**

By induction on $u \in L$. **Base Case.**

\begin{align*}
R(A(\bullet, v)) &= R(v) \quad (A.1) \\
&= A(R(v), \bullet) \quad (\text{Prop 7.7}) \\
&= A(R(v), R(\bullet)) \quad (R.1)
\end{align*}

**Induction.**

(c) **Proposition 7.11.** For all $u, v \in L$, $T(u, v) = A((R(u), v)$. **Proof.** By induction on $u \in L$

**Base Case.**

\begin{align*}
T(\bullet, v) &= v \quad (T.1) \\
&= A(\bullet, v) \quad (A.1) \\
&= A(R(\bullet), v) \quad (R.1)
\end{align*}

**Induction:** Assume $T(u, v) = A(R(u), v)$.

\begin{align*}
T(au, v) &= T(u, av) \quad (T.2a) \\
&= A(R(u), av) \quad (\text{I.H.}) \\
&= A(R(u), A(u, v)) \quad (A.2; A.1) \\
&= A(A(R(u), a\bullet), v) \quad (\text{Prop. 7.6}) \\
&= A(R(au), v) \quad (R.2a)
\end{align*}

Similarly for $T(bu, v)$.

(d) **Proposition 7.10.** For all $u \in L$, $T(u, \bullet) = R(u)$. **Proof.** Induction is not needed. By Proposition 7.11, $T(u, \bullet) = A(R(u), \bullet)$, and by Proposition 7.7, $A(R(u), \bullet) = R(u)$.

(e) **Proposition 7.12.** For all $u \in L$, $T(T(u, \bullet), \bullet) = u$. **Proof.** Induction is not needed. By Proposition 7.10, used twice, $T(T(u, \bullet), \bullet) = T(R(u), \bullet) = R(R(u))$. And by Proposition 7.9, $R(R(u)) = u$. 
2. The program below is called Wensley’s algorithm for computing the quotient of real numbers $x$ and $y$ to within tolerance $t$. Use the Theorem on Loop Invariants to prove the this program satisfies the post-condition $\{ z \leq x/y < z + t \}$.

\[
\begin{align*}
\{0 \leq x < y \leq 1\} \\
\text{begin} \\
z := 0; \quad d := 1; \quad u := 0; \quad v := \frac{1}{2} y; \\
\text{while } d > t \text{ do} \\
\{ \text{ INV } \equiv z \leq x/y < z + d \land u = zy \land v = \frac{1}{2} dy \} \\
\text{begin} \\
d := \frac{1}{2} d; \\
\text{if } u + v > x \text{ then skip} \\
\text{else begin} \\
z := z + d; \quad u := u + v; \quad \text{ end;} \\
v := \frac{1}{2} v \\
\text{end} \\
\{ z \leq x/y < z + t \}
\end{align*}
\]

SOLUTION

There are three things to prove:

INITIALIZATION. The program starts with $0 \leq x < y \leq 1$, and first reaches the while-loop with $z = 0$, $d = 1$, $u = 0$, and $v = \frac{1}{2} y$. The invariant INV is the conjunction of three propositions. All three are true when the variables’ current values are substituted.

1. $z \leq x/y \leq z + d$ $\rightarrow$ $0 \leq x/y \leq 0 + 1$, which holds because $0 \leq x < y \leq 1$.
2. $u = zy$ $\rightarrow$ $0 = 0 \cdot y$, which is true.
3. $v = \frac{1}{2} dy$ $\rightarrow$ $\frac{1}{2} \cdot 1 \cdot y = \frac{1}{2} y$, which is the initial value assigned to $v$.

INVARINANCE. If INV $\equiv z \leq x/y < z + d \land u = zy \land v = \frac{1}{2} dy$ holds and $d > t$ then the loop body executes, always computing new values for $d' = \frac{1}{2} d$ and $v' = \frac{1}{2} v$. The values of $x$ and $y$ are unchanged through out.

A. If conditional test $u + v > x$ is true, all other variables are unchanged.

1. $z' \leq x/y \leq z' + d' \rightarrow z \leq x/y \leq z + \frac{1}{2} d$. The first part, $z \leq x/y$, is given by INV. Also by INV, $x < u + v = zy + \frac{1}{2} dy = (z + \frac{1}{2} d)y = (z' + d')y$. Dividing through by $y$ preserves the inequality because $y$ is positive and gives $x/y \leq z' + d'$ as desired.
2. $u' = z'y'$ $\rightarrow$ $u = zy$, which is given by INV.
3. $v' = \frac{1}{2} d'y'$ $\rightarrow$ $\frac{1}{2} v = \frac{1}{2}(1/2) y$. Dividing this equation by $\frac{1}{2}$, we get $v = \frac{1}{2} dy$, which is given by INV.
B. If the conditional test, $u + v > x$ fails $z$ and $u$ are also changed and in addition to $d'$ and $v'$ we have.

$$z' = z + d' \quad \text{because new assignment to } d \text{ has already occurred}$$
$$= z + \frac{1}{2}d$$
$$u' = u + v \quad \text{because the conditional test fails}$$

Looking again at Inv one conjunct at a time,

1. $z' \leq x/y \leq z' + d' \quad \Rightarrow \quad z + \frac{1}{2}d \leq x/y \leq (z + \frac{1}{2}d) + \frac{1}{2}d. \quad z + \frac{1}{2}d + \frac{1}{2}d = z + d$, and Inv gives us $x/y < z + d$, so the right-hand inequality is valid. To show $z + \frac{1}{2}d \leq x/y$, Inv gives us that $u = zy$ and $v = \frac{1}{2}dy$, so $u + v = (z + \frac{1}{2}d)y$. At this point of the program, $u + v \leq x$ and $y > 0$. Thus, $(z + \frac{1}{2}d)y \leq x$, and dividing both sides by $y$ gives us what we want.

2. $u' = z'y' \quad \Rightarrow \quad u + v = (z + \frac{1}{2}d)y$. By Inv we have $u + v = zy + \frac{1}{2}dy = (z + \frac{1}{2}d)y$ as needed.

3. $v' = \frac{1}{2}d'y' \quad \Rightarrow \quad \frac{1}{2}v = \frac{1}{2}(\frac{1}{2}d)y$. Dividing this equation by $\frac{1}{2}$, we get $v = \frac{1}{2}dy$, which is given by Inv.

Termination. On termination we have $\text{Inv} \land d \leq t$. So it is immediate that $z \leq x/y < z + d \leq z + t$, thus satisfying the postcondition.
3. Define \( F : \mathbb{N} \to \mathbb{N} \) and \( G : \mathbb{N}^2 \to \mathbb{N} \) as follows:

\[
F(0) = 1 \quad \quad \quad G(0, m) = m
\]

\[
F(k+1) = (k + 1) \times F(k) \quad \quad \quad G(k+1, m) = G(k, m \times (k + 1))
\]

(a) Prove by induction on \( n \in \mathbb{N} \): For all \( n, m \in \mathbb{N} \), \( G(n, m) = m \times G(n, 1) \).

(b) Prove: For all \( n \in \mathbb{N} \), \( F(n) = G(n, 1) \).

**Solution**

**Proposition 0.** For all \( n \in \mathbb{N} \), \( F(n) = G(n, 1) \).

**Base Case.**

\[
F(0) = 1 = G(0, 1)
\]

**Induction.** Assume that \( F(k) = G(k, 1) \).

\[
F(k+1) = (k + 1) \times F(k) \quad \quad \quad \text{(Defn. } F) \]

\[
= (k + 1) \times G(k, 1) \quad \quad \quad \text{(I.H.)}
\]

\[
\vdash \quad \text{(Nowhere to go from here...)}
\]

\[
\equiv \quad G(k + 1, 1) \quad \quad \quad \text{(to reach the goal.)}
\]

You need to find and prove a more general result about \( G \).

**Proposition 1.** For all \( n, m \in \mathbb{N} \), \( G(n, m) = m \times G(n, 1) \).

**Proof.** The proof is by induction on \( k \in \mathbb{N} \).

**Base Case.**

\[
G(0, m) \overset{G} = m = m \times 1 \overset{F} = m \times F(0)
\]

**Induction.** Assume that \( G(k, m) = m \times G(k, 1) \). Then

\[
G(k + 1, m) = G(k, m(k + 1)) \quad \quad \quad \text{(Defn. } G) \]

\[
= [m(k + 1)] \times G(k, 1) \quad \quad \quad \text{(I.H.)}
\]

\[
= m \times [(k + 1) \times G(k, 1)] \quad \quad \quad \text{(algebra)}
\]

\[
= m \times G(k, k + 1) \quad \quad \quad \text{(I.H.)}
\]

\[
= m \times G(k + 1, 1) \quad \quad \quad \text{(Defn. } G)
\]

**Comment.** A more “elegant” version of Proposition 1 would be For all \( n, m, k \in \mathbb{N} \), \( G(n, m \times k) = m \times G(n, k) \). However, we don’t need that much generality for our purposes.
Corollary 2. For all \( n \in \mathbb{N} \), \( F(n) = G(n, 1) \).

proof. The proof is by induction on \( k \in \mathbb{N} \).

Base case.

\[
F(0) = 1 = G(0, 1)
\]

Induction. Assume that \( F(k) = G(k, 1) \).

\[
F(k + 1) = (k + 1) \times F(k) \quad \text{(Defn. } F) \\
= (k + 1) \times G(k, 1) \quad \text{(I.H.)} \\
= G(k, k + 1) \quad \text{(Prop. 1)} \\
= G(k + 1, 1) \quad \text{(Defn. } G) 
\]

\[\square\]
4. Performance estimation for recursive programs often involves recurrence relations like the one below. Let \( a \in \mathbb{N} \). The function \( T : \mathbb{N} \to \mathbb{N} \) is defined recursively by

\[
T(0) = a \\
T(k + 1) = T(k) + k + 1
\]

We would like to find a closed form for \( T \), that is, an algebraic expression that does not involve recursion.

Prove that for all \( n \in \mathbb{N} \), \( T(n) = a + \frac{n^2 + n}{2} \).

**Solution**

**Proof.** The proof is by induction on \( n \in \mathbb{N} \).

**Base case.**

\[
T(0) = a = a + \frac{0 + 0^2}{2}
\]

**Induction case.** Assume that \( T(k) = a + \frac{k^2 + k}{2} \).

\[
T(k + 1) = T(k) + k + 1 \quad \text{(Defn. } T) \\
= a + \frac{k^2 + k}{2} + k + 1 \quad \text{(I.H.)} \\
= a + \frac{k^2 + k + 2k + 2}{2} \quad \text{ (multiply } k + 1 \text{ by } 1 = \frac{2}{2}) \\
= a + \frac{k^2 + 3k + 2}{2} \quad \text{ (adding fractions)} \\
= a + \frac{(k^2 + 2k + 1) + (k + 1)}{2} \quad \text{ (algebra)} \\
= a + \frac{(k + 1)^2 + (k + 1)}{2} \quad \text{ (factoring } k^2 + 2k + 1 \text{)}
\]

as needed. This completes the induction. \( \square \)
Supplemental Problem. *H241 students should attempt this programming problem, but don’t spend more than two or three hours on it.*

In a programming language of your choice, write a program that takes no input and outputs its own source code.

---

**Solution**

No solution provided.