In Class Exercises – Axiomatic Semantics
Sample Solutions

1. (a) INVALID; \(3 \neq 4\).
   (b) INVALID; statement ends, but postcondition is false.
   (c) VALID; precondition is false.
   (d) INVALID; if \(y\) is negative and \(|y| > x\) the postcondition will be false.
   (e) VALID; \(y > x \land x > 0, \) so \(y > 0\). Therefore \(x + y > y\).
   (f) VALID; statement never terminates, so postcondition irrelevant (for partial correctness)
   (g) VALID; statement terminates and postcondition is true.
   (h) VALID; precondition is false.

<table>
<thead>
<tr>
<th>Result of assignment rule</th>
<th>Simplifies to</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) (Q : x * 3 = 15)</td>
<td>(Q : x = 5)</td>
</tr>
<tr>
<td>(b) (Q : 14 = 14)</td>
<td>(Q : true)</td>
</tr>
<tr>
<td>(c) (Q : (r - 1)^3 + sin(r - 1) - 12 = 4)</td>
<td>same</td>
</tr>
<tr>
<td>(d) (Q : x + a[y + 1] = \sum_{i=1}^{y+1} a[k])</td>
<td>(Q : x = \sum_{k=1}^{y} a[k])</td>
</tr>
</tbody>
</table>

2. {\(Q : (w = z \land z = 40) \text{ or } (w \neq z \land (w = 0 \lor z = 10))\)}
   if \(w = z\) then begin
   \{z = 40\}
   \{z = 10 * 4\}
   \(w := 4;\)
   \{z = 10 * w\}
   \{z ÷ 2 = 5 * w\}
   \(z := z ÷ 2;\)
   \{z = 5 * w\}
end
else begin
   \{w = 0 \lor z = 10\}
   \{w * z = 10 * w\}
   \(z := w * z;\)
   \{z = 10 * w\}
   \{z = 5 * 2 * w\}
   \(w := 2 * w\)
   \{z = 5 * w\}
end
\{z = 5 * w\}

3. \(\{Q\}\) Initialization \(\{P\}\)
   \{0 < n\}
   \(i := 1;\)
   \(\{\text{inv } P: (0 < i \leq n) \land (i \text{ is a power of } 2)\}\)

   1 is assigned to \(i\). From the precondition \(0 < n\), so the first clause is satisfied. 1 is a power of 2, so the second clause is satisfied.

   \(\{P \land B\}\) S \(\{P\}\)
\[
\{(0 < i \leq n) \land (i \text{ is a power of } 2) \land (2 \ast i \leq n)\}
\]

\[
i := i \ast 2
\]

\[
\{\text{inv } P: \ (0 < i \leq n) \land (i \text{ is a power of } 2)\}
\]

The precondition tells us that \(2 \ast i \leq n\), so assigning \(2 \ast i\) to \(i\) will result in \(i \leq n\). \(i > 0\) initially, and doubling \(i\) will maintain that. \(i\) is a power of 2 before the assignment statement, so doubling \(i\) will maintain that as well.

- \(P \land \neg B \Rightarrow R\)

\[
(0 < i \leq n) \land (i \text{ is a power of } 2) \land \text{not}(2 \ast i \leq n)
\]

\[
(0 < i \leq n) \land (i \text{ is a power of } 2) \land (2 \ast i > n)
\]

\[
(0 < i \leq n) \land (n < 2 \ast i) \land (i \text{ is a power of } 2)
\]

- Prove bound function \(t\) decreases

\(i\) doubles each time through the loop. The bound function is \(n - i\). If the loop guard was true, \(2 \ast i < n\), so doubling \(i\) decreases \(n - i\) in each iteration.

- \(P \land B \Rightarrow t > 0\)

\[
(0 < i \leq n) \land (i \text{ is a power of } 2) \land (2 \ast i \leq n)
\]

Since \(2 \ast i \leq n\) it is the case that \(i \neq n\). Therefore \(n - i > 0\).

5. Find a loop guard \(B\).

At the beginning we only know that \(n > 0\). To establish \(P\), we need to initialize \(a\), \(b\), and \(i\). If \(i = 1\), the job is easy; \(a = 1\) and \(b = 0\). So our initialization is

\[
\{\text{pre } Q: \ n > 0\}
\]

\[
\{\text{inv } P: \ (1 \leq i \leq n) \land (a = F_i) \land (b = F_{i-1})\}
\]

\[
\{\text{post } R: \ a = F_n\}
\]

The answer to the question “When are we done?” is “When \(P\) and \(R\) mean the same thing.” That happens when \(i = n\); when that happens, \(a = F_n\). Therefore, we are not done when \(i \neq n\), and that is our loop guard.

- Find initialization to establish invariant \(P\).

As a first guess, let’s try \(n - i\). (We got this by looking at the invariant and seeing that \(i\) ranges between 1 and \(n\).) We’ll come back to this after our loop is finished to make sure it works.

We can decrease \(t\) by incrementing \(i\):

\[
i := i + 1;
\]

- Guess bound function \(t\) and find ways to decrease it.

As a first guess, let’s try \(n - i\). (We got this by looking at the invariant and seeing that \(i\) ranges between 1 and \(n\).) We’ll come back to this after our loop is finished to make sure it works.

We can decrease \(t\) by incrementing \(i\):

\[
i := i + 1;
\]

- Ensure that \(P\) is reestablished.

We need to add code to ensure that the following triple is valid.

\[
\{\text{inv } P: \ (1 \leq i \leq n) \land (a = F_i) \land (b = F_{i-1})\}
\]

while \(i \neq n\) do begin

\[
i := i + 1 \end;
\]

\[
\{\text{inv } P: \ (1 \leq i \leq n) \land (a = F_i) \land (b = F_{i-1})\}\]
After $i$ is incremented, $a = F_{i-1}$ and $b = F_{i-2}$. Applying the formula from the problem, we get

$$\{ \text{inv } P: \ (1 \leq i \leq n) \land (a = F_i) \land (b = F_{i-1}) \}$$

while $i \neq n$ do begin
  $i := i + 1$;
  $temp := a$;
  $a := a + b$;
  $b := temp$;
end;

$$\{ \text{inv } P: \ (1 \leq i \leq n) \land (a = F_i) \land (b = F_{i-1}) \}$$

Lastly, we check that $n - i$ is okay as a bound function $(P \land B \Rightarrow t > 0)$. Since $t = n - i$, and the loop guard is $i \neq n$, and $i$ is initialized to 1 and is incremented by 1 in each iteration, $i$ is always $\leq n$. If there are iterations left, therefore, $i < n$, so $0 < n - i$. 