4 Program verification

17. The program Div is specified to compute the dividend of integers \( a \) by \( y \); this is defined to be the unique integer \( z \) such that there exist some integer \( r \) such that \( a = y \cdot z + r \). For example, if \( a = 15 \) and \( y = 6 \), then \( d = 2 \) because \( 15 = 2 \cdot 6 + 3 \), where \( r = 3 \neq 6 \). Let Div be given by

\[
\begin{align*}
\text{d} & := 0; \\
\text{while } (r \geq y) \\
& \quad \text{d} := y; \\
& \quad \text{r} := r - y; \\
& \quad \text{q} := \text{d} + 1; \\
\end{align*}
\]

Show that \( \text{Div}(\text{r}, \text{y}) \) is valid.

18. \( \text{Div}(\text{r}, \text{y}) \) is valid, where DivSetSub is given by

\[
\begin{align*}
\text{a} & := z; \\
\text{y} & := y; \\
\text{q} & := 0; \\
\text{while } (r > 0) \\
& \quad \text{a} := a + q; \\
& \quad \text{r} := r - q; \\
& \quad \text{q} := q + 1; \\
\end{align*}
\]

19. Why can, or cannot, you prove the validity of \( \text{Div}(\text{r}, \text{y}) \) by induction on \( a \)?

20. Let all while statements while \( (\text{a} \geq \text{b}) \) in \( P \) be annotated with invariant \( \text{in} \) at the start of their bodies, and \( \text{out} \) at the beginning of their body.

(a) Explain how a proof of \( \text{Div}(\text{a}, \text{b}) \) can be automatically reduced to showing the validity of some \( \text{in} \land \text{out} \land \text{in} \land \text{out} \).

(b) Identify such a sequence \( \text{in} \land \text{out} \land \text{in} \land \text{out} \) for the proof in Example 4.17 on page 287.

21. Given \( n = 5 \) test the correctness of Min.Sun on the arrays below.

\[
\begin{align*}
(\text{a}) & := [-3,1,-2,1,6] \\
(\text{b}) & := [1,4,6,-1,2] \\
(\text{c}) & := [2,4,1,5,3] \\
(\text{d}) & := [8,4,-1,3,2] \\
(\text{e}) & := [3,4,1,2,9] \\
(\text{f}) & := [1,2,3,4,5] \\
\end{align*}
\]

22. If we swap the first and second assignment in the while-statement of Min.Sun, it is false. Assign to \( a \) and then to \( t \), is the program still correct? Justify your answer.

23. Prove the partial correctness of S2 for Min.Sun.

24. The program Min.Sun does not reveal where a minimal sum subsection may be found in an input array. Adapt Min.Sun to achieve that. Can you do this with a single pass through the array?

25. Consider the proof rule

\[
\begin{align*}
\text{(a) } & \text{C } (\text{a}) \quad \text{(b) } \text{C } (\text{b}) \quad \text{(c) } \text{C } (\text{c}) \\
\text{(d) } & \text{C } (\text{d}) \quad \text{(e) } \text{C } (\text{e}) \quad \text{Con} \\
\end{align*}
\]

* You may have to strengthen your invariant.

4.6 Exercises

(a) Show that this proof rule is sound for \( \text{Div} \).

(b) Derive this proof rule from the ones on page 270.

(c) Explain how this rule, or its derived version, is used to establish the overall correctness of Min.Sun.

26. The maximal sum problem is to compute the maximal sum of all sections on page 268.

(a) Adapt the program from page 268 so that it computes the maximal sum of these sections.

(b) Prove the partial correctness of your modified program.

(c) Which aspects of the correctness proof given in Figure 4.3 on page 291 can be reused?

Exercises 4.4

1. Prove the validity of the following total-correctness sequence.

   (a) \( \text{while } (\text{a} \geq \text{b}) \text{ C } (\text{a} = \text{b}) \).

2. Prove total correctness of S1 and S2 for Min.Sun.

3. Prove that \( \text{Div} \) is sound for \( \text{Div} \).

4. Prove that \( \text{Div} \) is sound for \( \text{Div} \).

5. Implement program \text{Collatz} in a programming language of your choice such that the value of \( x \) is the program's input and the final value of \( x \) is its output.

6. Write a function that takes an integer \( f \) and returns the number of iterations it takes to reach 1.

7. Write an artifact that implements a function \( f \) that takes two integers \( a \) and \( b \) and returns their sum.

8. Calculate the sum of all even numbers between 1 and 100.

9. Calculate the sum of all odd numbers between 1 and 100.

10. Calculate the sum of all numbers between 1 and 100 that are divisible by 3.
4.7 Bibliographic notes

An early exposition of the program logic for partial and total correctness of programs written in an imperative language can be found in [Howard].
The text [Dijkstra76] contains a formal treatment of weakest preconditions.