Solution 1: Our first solution is based upon the “merge two ordered lists” algorithm.

\[
x := \text{in1.get()}; y := \text{in2.get();}
\]

// loop invariant:

// \( x \geq \text{Max(yPast)} \land y > \text{Max(xPast)} \land \text{outPast + (x, cntr) = f(xPast + x, yPast)} \)

do while (\( x \neq +\infty \lor y \neq +\infty \))

if (\( x \leq y \)) then

\[\text{out.put(x); out.put(cntr); cntr := 0; x := \text{in1.get();}}\]

else if (\( x = y \)) then

\[\text{cntr := cntr + 1; y := \text{in2.get();}}\]

else // \( x > y \)

\[y := \text{in2.get();}\]

fi

od

// postcondition: \( x = +\infty \land y = +\infty \land \text{outPast = f(xPast,yPast)} \)

Recall that a loop invariant is a boolean function (i.e., a predicate) of the program variables having the property that it is true immediately before and immediately after each iteration of the loop. To be useful in making a convincing argument of correctness, a loop invariant must have the further property that it, conjuncted with the negation of the loop guard, guarantees the postcondition of the loop.

Imagine that we were to take a snapshot of the algorithm’s state at some point during its execution. We use \( \text{xPast} \) (respectively, \( \text{yPast} \)) to refer to the sequence of values assumed by \( x \) (respectively, \( y \)) prior to assuming its current value. (That is, \( \text{xPast} \) (respectively, \( \text{yPast} \)) is the sequence of values returned by all calls to \( \text{in1.get()} \) (respectively, \( \text{in2.get()} \)) prior to the most recent call. Similarly, \( \text{outPast} \) is the sequence of ordered pairs produced by calls to \( \text{out.put()} \) so far. Let \( f \) be the function that, given two sequences as described in the problem description, produces the sequence of ordered pairs that should be the program’s output.

The + symbol denotes the concatenation operator on sequences. \( \text{Max} \), when applied to a sequence, yields the maximum value in that sequence, or \(-\infty \) if the sequence is empty.
Solution 2: Our second solution is an adaptation of the Balanced Line Algorithm.

\[
x := \text{in1.get(); } y := \text{in2.get();}
// loop invariant:
// \quad x \geq \text{Max}(y_{\text{Past}}) \land y > \text{Max}(x_{\text{Past}}) \land \text{outPast} = f(x_{\text{Past}}, y_{\text{Past}})
\]
\[
\quad \text{do while } (x \neq +\infty \lor y \neq +\infty)
\]
\[
\quad \quad \text{lesser} := \text{Min}(x, y);
\quad \text{cntr} := 0;
// loop invariant:
// \quad ((x = \text{lesser} \implies \text{outPast} + (x, \text{cntr}) = f(x_{\text{Past}} + x, y_{\text{Past}})) \land
// \quad ((x \neq \text{lesser} \implies \text{outPast} = f(x_{\text{Past}}, y_{\text{Past}}))
\]
\[
\quad \quad \text{do while } (y = \text{lesser})
\quad \quad \quad \text{cntr} := \text{cntr} + 1;
\quad \quad \quad y := \text{in2.get();}
\quad \text{od}
\]
\[
\quad \text{if } (x = \text{lesser}) \text{ then}
\quad \quad \text{out.put(x); out.put(cntr);}
\quad \text{fi}
\]
\[
\quad x := \text{in1.get();}
\text{od}
// postcondition: \quad x = +\infty \land y = +\infty \land \text{outPast} = f(x_{\text{Past}}, y_{\text{Past}})
\]

We can simplify this by observing that there is no point in counting occurrences of the current value of \( y \) if that value is smaller than \( x \)'s current value. This leads to ...

Solution 3:

\[
x := \text{in1.get(); } y := \text{in2.get();}
// loop invariant:
// \quad x \geq \text{Max}(y_{\text{Past}}) \land y > \text{Max}(x_{\text{Past}}) \land \text{outPast} = f(x_{\text{Past}}, y_{\text{Past}})
\]
\[
\quad \text{do while } (x \neq +\infty \lor y \neq +\infty)
\]
\[
\quad \quad \quad \text{do while } (x > y)
\quad \quad \quad \quad y := \text{in2.get();}
\quad \text{od}
\]
\[
\quad \text{cntr} := 0;
// loop invariant:
// \quad x \leq y \land x \geq \text{Max}(y_{\text{Past}}) \land y > \text{Max}(x_{\text{Past}}) \land \text{out} + (x, \text{cntr}) = f(x_{\text{Past}} + x, y_{\text{Past}})
\]
\[
\quad \quad \text{do while } (x = y)
\quad \quad \quad \text{cntr} := \text{cntr} + 1; y := \text{in2.get();}
\quad \text{od}
\]
\[
\quad \text{out.put(x); out.put(cntr);}
\quad x := \text{in1.get();}
\text{od}
// postcondition: \quad x = +\infty \land y = +\infty \land \text{outPast} = f(x_{\text{Past}}, y_{\text{Past}})