1) The value of “3 div -2” is -1. 1.4.3 defines integer division in terms of the usual mathematical division. The first case holds, so the result is the negative of the largest integer not larger than the magnitude of the real result. It is important to note that this definition is not dependent in any way upon LAX real arithmetic, so it remains valid even though MINILAX has only integer arithmetic.

2) According to 1.4.2, the two allowable coercions are dereferencing and voiding. In the third line of Figure 2.2, v is dereferenced to obtain its current value in order to make the comparison. In the tenth line of Figure 2.2, c:= c + 1 returns a ref integer that is voided.

3) If the expression of the iteration is initially false, the iteration will never be executed. Therefore it can deliver no result in this case. But if there is one case in which it can deliver no result, then it should never deliver a result. If it were to deliver a result sometimes but not others, the definition and the proof rule for the construct would become too complex, and it would serve as a source of programming errors.

4) One example is “if i = 0 then 10 else x end”. This clause yields an integer value. The first statement list yields a value of type integer, while the second yields a value of type ref integer. Balancing requires that the result of the second statement list be coerced to integer. This is the minimum set of coercions that will allow the two statement lists to yield values of the same type.

5) The program can be developed from the formal specification by replacing the constant u by the variable j in the result expression and adding a conjunct:

\[ R = \sum_{i=l+1}^{j-1} i \quad \text{and} \quad j = u \]

The first conjunct becomes the loop invariant of a while statement, and the inverse of the second becomes the test. \( R := 0; \ j := l + 1 \); establishes the loop invariant initially, and \( j := j + 1 \) makes progress towards the goal because it always decreases the integer \( t = (u - j) \). The complete program is:

\begin{verbatim}
declare l: integer; u: integer
\end{verbatim}
begin
    declare R: integer; j: integer
    begin
        R := 0;
        if l < u then j := l + 1 else j := u + 1; u := l end;
        while j < u do R := R + j; j := j + 1 end;
        R
    end
end