Format for Program Verification using Hoare Axioms

I believe the easiest way to employ these axioms is to add lines of assertions to the actual code. We might think of these as comments (and one programming language – ADA – actually had the intention at one point of developing compilers that could use the comments to do mechanical verification on the code). Thus the general form will be:

<line of code>
____<one or more lines of assertions>

In my solutions I will always make it clear what is code and what is assertion by the hosrizontal lines – one per assertion.

To get an idea how this works, let's do an example verification.

Problem: Assuming x and y are integer variables, prove correct this code is correct with respect to precondition "y is defined" and postcondition "$x \geq 1$":

```
if y > 0 then
        x := y+6
        if x > 11 then
                x : = x-10
        endif
else
        x := 4-y
        y := y-1
        if y = -3 then
                x : = x-3
        endif
endif
```

__________________________ *y is defined*

if y > 0 **then** ____________ $y > 0$

  x := y+6____________ $y > 0 \wedge x = y + 6$

  ____________ $x > 6$

  if x > 11 **then**_______ $x > 11$

    x : = x-10_____ $(x' > 11) \wedge (x = x' - 10)$

  ____________ $x > 1$

  endif ______________ $(x > 6) \vee (x > 1)$

  ____________ $x > 1$

else _____________________ $y \leq 0$

  x := 4-y____________ $(y \leq 0) \wedge (x = 4 - y)$

  ____________ $x \geq 4$

  y := y-1 ____________ $x \geq 4$

  if y = -3 **then**________ $x \geq 4$

    x : = x-3______ $(x' \geq 4) \wedge (x = x' - 3)$

  ____________ $x \geq 1$

  endif_____________ $(x \geq 4) \vee (x \geq 1)$

  ____________ $x \geq 1$

endif____________________ $(x > 1) \vee (x \geq 1)$

__________________________ $x \geq 1$

Remarks:

1. Notice the precondition is the first assertion and the postcondition is the last one.

2. Sometimes a line of code may be followed by more than one assertion line. For example:

x := y+6____________ $y > 0 \wedge x = y + 6$
____________ $x > 6$

This is fine as long as the successive assertions are consequences of the ones immediately preceding. Sometimes students are tempted to bring back assertions that don't occur on the immediately preceding lines. They see this as a sort of shorthand but it is very dangerous because, if assumptions need not be explicit, we have little idea of what actually is being claimed at each line. Thus, we don't want this sort of treatment:

x := 6____________ $x = 6$
y := 2____________ $y = 2$
____________ $x = 6 \wedge y = 2$

Since the assertion about x has been dropped but then reintroduced. This is the correct treatment:

x := 6____________ $x = 6$
y := 2____________ $x = 6 \wedge y = 2$

3. Not inconsistent with what was just said, recognize that the interpretation of "immediately preceding assertion" takes into consideration that the flow of control may actually skip some lines of code. This will happen with conditionals and loops. Thus, the fragment above

__________________ $x > 6$
if x > 11 **then**_______ $x > 11$
    x : = x-10_____ $(x' > 11) \wedge (x = x' - 10)$
____________ $x > 1$
endif ______________ $(x > 6) \vee (x > 1)$

is fine. We know that the assertion "$x > 6$" is true prior to the **if** and thus will continue to be true in the case that the **if** condition "x > 11" is not satisfied. That is why we are able to assert "$(x > 6) \vee (x > 1)$" at the end of the conditional:

either
"$x > 6$" holds because the **if** condition was false prior to the conditional
or
"$x > 1$" holds because the **if** condition was true.

Don't think of this as reintroducing an unstated assertion as was covered in Remark 2.