Program Proof Examples

The programs that follow should all be interpreted as operating over $\mathbb{N}$, the natural numbers. All program variables range of values in $\mathbb{N}$ and the operations of addition, subtraction, division, and multiplicaiton are restricted to results in $\mathbb{N}$.

In particular, subtraction is only defined when the minuend is greater than or equal to the subtrahend, that is, when the resulting difference is non-negative. Division returns the integer quotient.

Synthesize and prove the verification conditions for each program and translate them to English. The following page is and example.

Example

{ PRE}
begin
z := 0;
while TEST **inv** { INV } **do** BODY
end
{ POST}

$$
\begin{aligned}
\text{PRE} &\equiv x = A \wedge y = B \\
\text{INV} &\equiv z + xy = AB \\
\text{POST} &\equiv z = AB \\
\text{TEST} &\equiv x \neq 0 \\
\text{THEN} &\equiv \texttt{begin } x \texttt{ := } \tfrac{1}{2}x \texttt{ ; } y \texttt{ := } 2y \texttt{ end} \\
\text{ELSE} &\equiv \texttt{begin } x \texttt{ := } x - 1 \texttt{ ; } z \texttt{ := } z + y \texttt{ end} \\
\text{BODY} &\equiv \texttt{if } even?(x) \texttt{ then } \text{THEN} \texttt{ else } \text{ELSE} \\
\text{LOOP} &\equiv \texttt{while } \text{TEST} \texttt{ inv } \{ \text{INV} \} \texttt{ do } \text{BODY}
\end{aligned}
$$

Goals:

(1) $\{\text{PRE}\}$ **begin** z := 0 ; LOOP **end** $\{\text{POST}\}$

(2) $\{\text{PRE}\}$ z := 0 $\{\text{INV}\}$

(3) $\{\text{INV}\}$ LOOP $\{\text{POST}\}$

(4) $\{\text{PRE}\} \Rightarrow \text{INV}\begin{bmatrix}0\\z\end{bmatrix}$

(5) $\{\text{INV} \wedge x \neq 0\}$ BODY $\{\text{INV}\}$

(6) $\text{INV} \wedge x = 0 \Rightarrow \text{POST}$

(7) $\{\text{INV} \wedge x \neq 0 \wedge even?(x)\}$ THEN $\{\text{INV}\}$

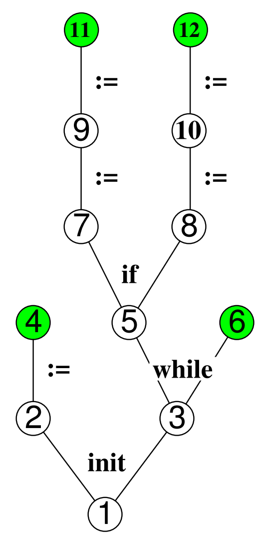
(8) $\{\text{INV} \wedge x \neq 0 \wedge \neg even?(x)\}$ ELSE $\{\text{INV}\}$

(9) $\{\text{INV} \wedge x \neq 0 \wedge even?(x)\}\ x := \frac{1}{2}x\ \{\text{INV}\begin{bmatrix}2y\\y\end{bmatrix}\}$

(10) $\{\text{INV} \wedge x \neq 0 \wedge \neg even?(x)\}\ x := x - 1\ \{\text{INV}\begin{bmatrix}z+y\\z\end{bmatrix}\}$

(11) $\text{INV} \wedge x \neq 0 \wedge even?(x) \Rightarrow \text{INV}\begin{bmatrix}2y\\y\end{bmatrix}\begin{bmatrix}\frac{1}{2}x\\x\end{bmatrix}$

(12) $\text{INV} \wedge x \neq 0 \wedge \neg even?(x) \Rightarrow \text{INV}\begin{bmatrix}z+y\\z\end{bmatrix}\begin{bmatrix}x-1\\x\end{bmatrix}$



Verification Conditions:

(4) $x = A$ and $y = B$ imply $0 + xy = AB$

(6) $z + xy = AB$ and $x = 0$ imply $z = AB$

(11) $z + xy = AB$ and $x \neq 0$ and $even?(x)$ imply $z + \left(\frac{1}{2}x\right)(2y) = AB$

(12) $z + xy = AB$ and $x \neq 0$ and $odd?(x)$ imply $(z + y) + (x - 1)y = AB$

Program 1

```
{x = A ∧ y = B ∧ A ≥ B}
while  y ≠ 0 do
   begin
   x := x − 1;
   y := y − 1
   end
{x = A − B}
```

Program 2

```
{x = A ∧ y = B}
begin
z := 1;
while  y ≠ 0 do  {z · x^y = A^B}
 begin
 z := z * x;
 y := y - 1
 end
end
{z = A^B}
```

Program 3

```
{x = A ∧ y = B}
begin
q := 0;
r := x;
while  r ≥ y do  {q · y + r = A}
   begin
   q := q + 1;
   r := r − y
   end
end
{(q · y + r = A) ∧ (r < y)}
```

Program 4

$\{x = A\}$
```
begin
z := 1;
while x ≠ 0 do {z · x! = A!}
   begin
   z := z * x;
   x := x − 1
   end
end
```
$\{z = A!\}$ (z is "A factorial.")

Program 5

$\{x = A \wedge y = B\}$
begin
z := 1;
while $y \neq 0$ do $\{z \cdot x^y = A^B\}$
  if $even?(y)$
    then begin y := $y/2$; x := $x * x$ end
    else begin y := $y - 1$; z := $z * x$ end
end
$\{z = A^B\}$

Program 6

```
{ x = A ∧ y = B }
begin
z := 1;
while  y ≠ 0 do  { z · x^y = A^B }
   begin
   while  even?(y) do  { z · x^y = A^B ∧ y ≠ 0 }
      begin
      y := y/2
      x := x * x
      end ;
   z := z * x;
   y := y − 1
   end
end
{ z = A^B }
```

Program 7

$\{x = A \wedge y = B\}$

```
begin
while  x ≠ y do
   { gcd(x, y) = gcd(A, B)}
   if  x  <  y
     then  y := y  −  x
     else  x := x  −  y
end
```

$\{x = \mathrm{gcd}(A, B)\}$

Program 8

Program taken from, Saud Alagic' and Michael A. Arbib, The Design of Well-structured and Correct Programs, Springer, New York, 1978, (Ch. 2, Sec. 2.7, p 44)

```
begin
{x > 0 ∧ y > 0}
r := x;
q := 0;
w := y;
while w ≤ r do
{∃ n (w = 2^n y) ∧ (x = qw + r) ∧ (0 ≤ r)}
   w := 2w;
while w ≠ y do
{∃ n (w = 2^n y) ∧ (x = qw + r) ∧ (0 ≤ r < w)}
  begin
  q := 2q;
  w := w ÷ 2;
  if w ≤ r then
    begin
    r := r − w;
    q := 1 + q;
    end
  else
    r := r;      [i.e., do nothing]
  end
{ (x = qy + r) ∧ (0 ≤ r < y) }
end
```