Section 5.5

Sterling Jeppson

November 26, 2020

Exercises 1-5 contain a while loop and a predicate. In each case show that if the predicate is true before entry to the loop, then it is also true after exit from the loop.

Problem 1

Algorithm 1

```
1: while ($m \geq 0$ and $m \leq 100$) do
2:     $m := m + 1$
3:     $n := n - 1$
4: end while
```

predicate: $m + n = 100$

Solution

Proof. Suppose that the predicate $m + n = 100$ is true before entry to the loop. Then

$$m_{\text{old}} + n_{\text{old}} = 100$$

After execution of the loop,

$$m_{\text{new}} = m_{\text{old}} + 1 \quad \text{and} \quad n_{\text{new}} = n_{\text{old}} - 1$$

so

$$\begin{aligned} m_{\text{new}} + n_{\text{new}} &= (m_{\text{old}} + 1) + (n_{\text{old}} - 1) \\ &= m_{\text{old}} + n_{\text{old}} = 100 \end{aligned}$$ □

Problem 2

Algorithm 2

```
1: while ($m \geq 0$ and $m \leq 100$) do
2:     $m := m + 4$
3:     $n := n - 2$
4: end while
```

predicate: $m + n$ is odd

Solution

Proof. Suppose that the predicate $m + n$ is odd is true before entry to the loop. Then

$$m_{\text{old}} + n_{\text{old}} = \text{odd}$$

After execution of the loop,

$$m_{\text{new}} = m_{\text{old}} + 4 \quad \text{and} \quad n_{\text{new}} = n_{\text{old}} - 2$$

so

$$\begin{aligned} m_{\text{new}} + n_{\text{new}} &= (m_{\text{old}} + 4) + (n_{\text{old}} - 2) \\ &= (m_{\text{old}} + n_{\text{old}}) + (4 - 2) \\ &= \text{odd} + 2 \\ &= \text{odd} + \text{even} = \text{odd} \end{aligned}$$

□

Problem 3

Algorithm 3

```
while ($m \geq 0$ and $m \leq 100$) do
    $m := 3 \cdot m$
    $n := 5 \cdot n$
end while
```

predicate: $m^3 > n^2$

Solution

Proof. Suppose that the predicate $m^3 > n^2$ is true before entry to the loop. Then

$$m_{\text{old}}^3 > n_{\text{old}}^2$$

After execution of the loop,

$$m_{\text{new}} = 3 \cdot m_{\text{old}} \quad \text{and} \quad n_{\text{new}} = 5 \cdot n_{\text{old}}$$

so

$$m_{\text{new}}^3 = (3 \cdot m_{\text{old}})^3 = 27 \cdot m_{\text{old}}^3 \quad \text{and} \quad n_{\text{new}}^2 = (5 \cdot n_{\text{old}})^2 = 25 \cdot n_{\text{old}}^2$$

Therefore

$$\begin{aligned} m_{\text{old}}^3 &> n_{\text{old}}^2 \\ 27 \cdot m_{\text{old}}^3 &> 25 \cdot n_{\text{old}}^2 \\ m_{\text{new}}^3 &> n_{\text{new}}^2 \end{aligned}$$

□

Problem 4

Algorithm 4

```
while ($n \geq 0$ and $n \leq 100$) do
    $n := n + 1$
end while
```

predicate: $2^n < (n+2)!$

Solution

Proof. Suppose that the predicate $2^n < (n+2)!$ is true before entry to the loop. Then

$$2^{n_{\text{old}}} < (n_{\text{old}} + 2)!$$

After execution of the loop,

$$n_{\text{new}} = n_{\text{old}} + 1$$

so,

$$2^{n_{\text{new}}} = 2^{n_{\text{old}}+1} = 2 \cdot 2^{n_{\text{old}}} \quad \text{and} \quad (n_{\text{new}}+2)! = (n_{\text{old}}+3)! = (n_{\text{old}}+3)(n_{\text{old}}+2)!$$

Therefore

$$\begin{aligned} 2^{n_{\text{old}}} &< (n_{\text{old}} + 2)! \\ 2 \cdot 2^{n_{\text{old}}} &< (n_{\text{old}} + 3)(n_{\text{old}} + 2)! \qquad n \geq 0 \\ 2^{n_{\text{new}}} &< (n_{\text{new}} + 2)! \end{aligned}$$

□

Problem 5

Algorithm 5

```
while ($n \geq 3$ and $n \leq 100$) do
    $n := n + 1$
end while
```

predicate: $2n + 1 \leq 2^n$

Solution

Proof. Suppose that the predicate $2n + 1 \leq 2^n$ is true before entry to the loop. Then

$$2n_{\text{old}} + 1 \leq 2^{n_{\text{old}}}$$

After execution of the loop,

$$n_{\text{new}} = n_{\text{old}} + 1$$

so

$$2n_{\text{new}} + 1 = 2(n_{\text{old}} + 1) + 1 = 2n_{\text{old}} + 3 \quad \text{and} \quad 2^{n_{\text{new}}} = 2^{n_{\text{old}}+1} = 2 \cdot 2^{n_{\text{old}}}$$

Therefore

$$\begin{aligned} 2n_{\text{old}} + 1 &\leq 2^{n_{\text{old}}} \\ 2(2n_{\text{old}} + 1) &\leq 2(2^{n_{\text{old}}}) \\ 4n_{\text{old}} + 2 &\leq 2^{n_{\text{new}}} \\ 2n_{\text{old}} + 3 &\leq 2^{n_{\text{new}}} && n \geq 3 \text{ so } 2n > 1 \\ 2n_{\text{new}} + 1 &\leq 2^{n_{\text{new}}} \end{aligned}$$

□

Exercises 6-9 each contain a while loop annotated with a pre-condition and a post-condition and also a loop invariant. In each case, use the loop invariant theorem to prove the correctness of the loop with respect to the pre-conditions and post-conditions.

Problem 6

Algorithm 6

Precondition: $m \in \mathbb{Z}^{nonneg}$, $x \in \mathbb{R}$, $i = 0$, and $exp = 1$.

```
1: while ($i \neq m$) do
2:     $exp := exp \cdot x$
3:     $i := i + 1$
4: end while
```

Postcondition: $exp = x^m$

Loop invariant: $I(n)$ is "$exp = x^n$ and $i = n$."

Solution

Proof.

I. Basis Property: $I(0)$ is "$exp = x^0 = 1$ and $i = 0$." But by the preconditions $exp = 1$ and $i = 0$. Thus $I(0)$ is true before the first iteration of the loop.

II. Inductive Property: Suppose k is a nonnegative integer such that $G \wedge I(k)$ is true before an iteration of the loop. Since G is true, $i \neq m$ and the loop is entered. Before execution of the loop,

$$exp_{\text{old}} = x^k \quad \text{and} \quad i_{\text{old}} = k$$

After execution of the loop,

$$exp_{\text{new}} = exp_{\text{old}} \cdot x = x^k \cdot x = x^{k+1} \quad \text{and} \quad i_{\text{new}} = i_{\text{old}} + 1 = k + 1$$

Hence $I(k+1)$ is true.

III. Eventual Falsity of the Guard: The guard G is the condition $i \neq m$, and $m \in \mathbb{Z}^{nonneg}$. By I and II, it is known that for all integers $n \geq 0$, if the loop is iterated through n times, then $i = n$. So after m iterations of the loop, $i = m$. Thus G becomes false after m iterations of the loop.

IV. Correctness of the Post-Condition: According to the post-conditions, the value of exp after execution of the loop should be x^m. But if G becomes false after N iterations, $i = m$. And if $I(N)$ is true, $i = N$ and $exp = x^N$. Since G is false and $I(N)$ is true, $m = i = N$ and $exp = x^m$ as required. □

Problem 7

Algorithm 7

Precondition: $m \in \mathbb{Z}^+$, $largest = A[1]$, and $i = 1$

```
1: while ($i \neq m$) do
2:     $i := i + 1$
3:     if $A[i] > largest$ then
4:         $largest := A[i]$
5:     end if
6: end while
```

Postcondition: $largest = \max(A[1], A[2], ..., A[m])$

Loop invariant: $I(n)$ is "$largest = \max(A[1], A[2], ..., A[n+1])$ and $i = n + 1$."

Solution

Proof.

I. Basis Property: $I(0)$ is "$largest = \max(A[1]) = A[1]$ and $i = 0 + 1 = 1$." But by the preconditions $largest = A[1]$ and $i = 1$. Thus $I(0)$ is true before the first iteration of the loop.

II. Inductive Property: Suppose k is a nonnegative integer such that $G \wedge I(k)$ is true before an iteration of the loop. Since G is true, $i \neq m$ and the loop is entered. Before execution of the loop,

$$largest_{\text{old}} = \max(A[1], A[2], ..., A[k+1]) \quad \text{and} \quad i_{\text{old}} = k + 1$$

After execution of the loop,

$$largest_{\text{new}} = \max(largest_{\text{old}}, A[k+2]) \quad \text{and} \quad i_{\text{new}} = i_{\text{old}} + 1$$

It follows that

$$largest_{\text{new}} = \max(A[1], A[2], ..., A[k+2]) \quad \text{and} \quad i_{\text{new}} = k+2$$

Hence $I(k+1)$ is true.

III. Eventual Falsity of the Guard: The guard G is the condition $i \neq m$, and $m \in \mathbb{Z}^+$. By I and II, it is known that for all integers $n \geq 0$, if the loop is iterated through n times then $i = n+1$. So after $m-1$ iterations of the loop, $i = m$. Thus G becomes false after $m-1$ iterations of the loop.

IV. Correctness of the Post-Condition: According to the post-conditions, the value of $largest$ after execution of the loop should be $\max(A[1], A[2], ..., A[m])$. But if G becomes false after N iterations, $i = m$. And if $I(N)$ is true, $i = N+1$ and $largest = \max(A[1], A[2], ..., A[N+1])$. By the transitive property of equality on i, $m = N+1$. It follows that $largest = \max(A[1], A[2], ..., A[m])$. □

Problem 8

Algorithm 8

Precondition: $m \in \mathbb{Z}^+$, $sum = A[1]$, and $i = 1$

```
while ($i \neq m$) do
    $i := i + 1$
    $sum := sum + A[i]$
end while
```

Postcondition: $sum = A[1] + A[2] + ... + A[m]$

Loop invariant: $I(n)$ is "$sum = A[1] + A[2] + ... + A[n+1]$ and $i = n+1$."

Solution

Proof.

I. Basis Property: $I(0)$ is "$sum = A[1]$ and $i = 0 + 1 = 1$." But by the pre-conditions $sum = A[1]$ and $i = 1$. Thus $I(0)$ is true before the first iteration of the loop.

II. Inductive Property: Suppose k is a nonnegative integer such that $G \wedge I(k)$ is true before an iteration of the loop. Since G is true, $i \neq m$ and the loop is entered. Before execution of the loop,

$$sum_{\text{old}} = A[1] + A[2] + ... + A[k+1] \quad \text{and} \quad i_{\text{old}} = k+1$$

After execution of the loop,

$$sum_{\text{new}} = sum_{\text{old}} + A[k+2] \quad \text{and} \quad i_{\text{new}} = i_{\text{old}} + 1$$

It follows that

$$sum_{\text{new}} = A[1] + A[2] + ... + A[k+2] \quad \text{and} \quad i_{\text{new}} = k+2$$

Hence $I(k+1)$ is true.

III. Eventual Falsity of the Guard: The guard G is the condition $i \neq m$. It follows from I and II that for all integers $n \geq 0$ if the loop is iterated through n times then $i = n+1$. So after $m-1$ iterations of the loop, $i = m$. Thus G becomes false after $m-1$ iterations of the loop.

IV. Correctness of the Post-Condition: According to the post-conditions the value of sum after execution of the loop should be $sum = A[1] + A[2] + ... + A[m]$. But if G becomes false after N iterations, $i = m$. And if $I(N)$ is true, $i = N+1$ and $sum = A[1] + A[2] + ... + A[N+1]$. By the transitive property of equality on i, $m = N+1$. It follows that $sum = A[1] + A[2] + ... + A[m]$. □

Problem 9

Algorithm 9

Precondition: $a = A$ and $A \in \mathbb{Z}^+$

```
1: while ($a > 0$) do
2:     $a := a - 2$
3: end while
```

Postcondition: A is even $\implies a = 0$ and A is odd $\implies a = -1$.

Loop invariant: $I(n)$ is "Both a and A are even integers or both are odd integers and, in either case, $a \geq -1$."

Solution

Proof.

I. Basis Property: $I(0)$ is "Both a and A are even integers or both are odd integers and, in either case, $a \geq -1$." But by the pre-conditions $a = A$ which means that both have the same parity. Also $A \in \mathbb{Z}^+$ and so $a > -1$. Thus $I(0)$ is true before the first iteration of the loop.

II. Inductive Property: Suppose k is a nonnegative integer such that $G \wedge I(k)$ is true before an iteration of the loop. Since G is true, $a > 0$ and the loop is entered. Before execution of the loop,

$$a_{\text{old}} \text{ and } A \text{ are either both even or both odd and } a_{\text{old}} \geq 1$$

After the execution of the loop,

Case 1($\boldsymbol{A}$ odd): In this case a_{old} is odd and $a_{\text{old}} \geq 1$. Now after the execution of the loop

$$\begin{aligned} a_{\text{new}} &= a_{\text{old}} - 2 \\ &= \text{odd} - \text{even} \\ &= \text{odd} \end{aligned}$$

Furthermore since $a_{\text{old}} \geq 1$ it follows that $a_{\text{new}} \geq -1$.

Case 2($\boldsymbol{A}$ even): In this case a_{old} is even and $a_{\text{old}} \geq 2$. Now after the execution of the loop

$$\begin{aligned} a_{\text{new}} &= a_{\text{old}} - 2 \\ &= \text{even} - \text{even} \\ &= \text{even} \end{aligned}$$

Furthermore since $a_{\text{old}} \geq 2$ it follows that $a_{\text{new}} \geq 0$.

Thus $I(k+1)$ is true.

III. Eventual Falsity of the Guard: The guard G is the condition $a > 0$. Each iteration of the loop reduces the value of a by 2 and yet leaves $a \geq -1$. Thus the values of a form a decreasing sequence of integers all greater than -2 and so by the well-ordering principle there must be a smallest such a, say a_{min}. Then $a_{\text{min}} \leq 0$. If $a_{\text{min}} > 0$ then the loop would iterate another time, and a new value of a equal to $a_{\text{min}} - 2$ would be obtained. But this would contradict the fact that a_{min} is the smallest value of a obtained by repeated iteration of the loop. Hence as soon as the value $a = a_{\text{min}}$ is computed, the value of a becomes less than or equal to 0, and so the guard G is false.

IV. Correctness of the Post-Condition: Suppose that for some nonnegative integer N, G is false and $I(N)$ is true. Since G is false, $a \leq 0$ and since $I(N)$ is true, A and a have the same parity and $a \geq -1$. Now the only two possible values for a are $a = -1$ and $a = 0$. It follows by the truth of $I(N)$ that

$$A \text{ is odd} \implies a \text{ is odd} \implies a = -1$$

and that

$$A \text{ is even} \implies a \text{ is even} \implies a = 0$$

We now have that A is even $\implies a = 0$ and A is odd $\implies a = -1$. $\square$

Problem 10

Prove the correctness of the **while** loop of Algorithm 4.8.3

Algorithm 4.8.3

Precondition: $A, B \in \mathbb{Z}^+$, and $a = A$ and $b = B$

```
1: while ($a \neq 0$ and $b \neq 0$) do
2:     if $a \geq b$ then
3:         $a := a - b$
4:     else
5:         $b := b - a$
6:     end if
7: end while
```

Postcondition: Either $a = 0$ and $b = \gcd(A, B)$ or $b = 0$ and $a = \gcd(A, B)$

Loop invariant:

$$I(n) \text{ is } (1)\ a, b \in \mathbb{Z}^{nonneg} \text{ and } \gcd(a, b) = \gcd(A, B).$$
$$(2) \text{ at most one of } a \text{ and } b \text{ equals } 0,$$
$$(3)\ 0 \leq a + b \leq A + B - n.$$

Solution

Proof.

I. Basis Property: $I(0)$ is that $a, b \in \mathbb{Z}^{nonneg}$ and $\gcd(a, b) = \gcd(A, B)$, at most one of a and b equals 0, and $0 \leq a + b \leq A + B - n$.. But by the preconditions $a = A$ and $b = B$ and $A, B \in \mathbb{Z}^+$. It follows that a and b are both nonnegative and $\gcd(a, b) = \gcd(A, B)$. Since a and b are both positive it is true that at most one of a and b equals 0. Finally since a and b are both positive $0 \leq a + b$ and since $a = A$, $b = B$, and $n = 0$ it follows that $a + b \leq A + B - 0$. Thus $0 \leq a+b \leq A+B-0$ and so $I(0)$ is true before the first iteration of the loop.

II. Inductive Property: Suppose k is a nonnegative integer such that $G \wedge I(k)$ is true before an iteration of the loop. Since G is true, $a \neq 0$ and $b \neq 0$ and the loop is entered. Before execution of the loop a_{old}, $b_{\text{old}} \in \mathbb{Z}^+$, at most one of a_{old} and b_{old} equals 0 and

$$\gcd(a_{\text{old}}, b_{\text{old}}) = \gcd(A, B) \quad \text{and} \quad 0 \leq a_{\text{old}} + b_{\text{old}} \leq A + B - k$$

After execution of the loop,

$$a_{\text{new}} = \begin{cases} a_{\text{old}} - b_{\text{old}} & \text{if } a_{\text{old}} \geq b_{\text{old}} \\ a_{\text{old}} & \text{if } a_{\text{old}} < b_{\text{old}} \end{cases}$$

$$b_{\text{new}} = \begin{cases} b_{\text{old}} & \text{if } a_{\text{old}} \geq b_{\text{old}} \\ b_{\text{old}} - a_{\text{old}} & \text{if } a_{\text{old}} < b_{\text{old}} \end{cases}$$

To show (1), observe that since a_{old}, $b_{\text{old}} \in \mathbb{Z}^+$, it follows from definition that $a_{\text{new}}, b_{\text{new}} \in \mathbb{Z}^{nonneg}$. From problem 4.8.24a we have that

$$a \geq b > 0 \implies \gcd(a, b) = \gcd(a - b, b)$$

It now follows from definition that $\gcd(a_{\text{new}}, b_{\text{new}}) = \gcd(a_{\text{old}}, b_{\text{old}}) = \gcd(A, B)$.

To show (2), note that $a_{\text{old}}, b_{\text{old}} \in \mathbb{Z}^+$. Furthermore, either $a_{\text{new}} = a_{\text{old}}$ or $b_{\text{new}} = b_{\text{old}}$. Thus at most one of a_{new} or b_{new} equals 0.

To show (3), observe that

$$a_{\text{new}} + b_{\text{new}} = \begin{cases} a_{\text{old}} - b_{\text{old}} + b_{\text{old}} & \text{if } a_{\text{old}} \geq b_{\text{old}} \\ a_{\text{old}} + b_{\text{old}} - a_{\text{old}} & \text{if } a_{\text{old}} < b_{\text{old}} \end{cases}$$

Thus

$$a_{\text{new}} + b_{\text{new}} = \begin{cases} a_{\text{old}} & \text{if } a_{\text{old}} \geq b_{\text{old}} \\ b_{\text{old}} & \text{if } a_{\text{old}} < b_{\text{old}} \end{cases}$$

Since $a_{\text{old}}, b_{\text{old}} \in \mathbb{Z}^+$ it follows that $a_{\text{old}} \geq 1$ and $b_{\text{old}} \geq 1$. Thus

$$\begin{aligned} a_{\text{new}} + b_{\text{new}} + 1 &= a_{\text{old}} + 1 \quad \text{or} \quad 1 + b_{\text{old}} \\ &\leq a_{\text{old}} + b_{\text{old}} \end{aligned}$$

Now from the assumed truth of part (3) of $I(k)$ we have that

$$\begin{aligned} &0 \leq a_{\text{old}} + b_{\text{old}} \leq A + B - k \\ &0 \leq a_{\text{new}} + b_{\text{new}} + 1 \leq A + B - k \\ &0 \leq a_{\text{new}} + b_{\text{new}} \leq A + B - k - 1 \\ &0 \leq a_{\text{new}} + b_{\text{new}} \leq A + B - (k + 1) \end{aligned}$$

We have shown the truth of part (1), (2), and (3) and hence $I(k + 1)$ is true.

III. Eventual Falsity of the Guard: The guard G is the condition $a \neq 0$ and $b \neq 0$. By I and II, it is known that

(1) $a, b \in \mathbb{Z}^{nonneg}$ and $\gcd(a, b) = \gcd(A, B)$.
(2) at most one of a and b equals 0,
(3) $0 \leq a + b \leq A + B - n$.

It follows that after $A + B - 1$ iterations,

$$\begin{aligned} &0 \leq a + b \leq A + B - n \\ &0 \leq a + b \leq A + B - (A + B - 1) \\ &0 \leq a + b \leq 1 \end{aligned}$$

Now since a and b are both nonnegative and only one of a and b can equal 0, it must be that either $a = 0$ or $b = 0$. Thus G becomes false after at most $A + B - 1$ iterations of the loop.

IV. Correctness of the Post-Condition: Suppose that for some nonnegative integer N, G is false and $I(N)$ is true. Since G is false, $a = 0$ or $b = 0$, and since $I(N)$ is true only one of a and b equals 0 and,

$$\gcd(a, b) = \gcd(A, B)$$

By substitution we have

$$a = 0 \implies \gcd(0, b) = \gcd(A, B) \quad \text{and} \quad b = 0 \implies \gcd(a, 0) = \gcd(A, B)$$

But by Lemma 4.8.1,

$$\gcd(0, b) = \text{b} \quad \text{and} \quad \gcd(a, 0) = a$$

Hence either $a = 0$ and $b = \gcd(A, B)$ or $b = 0$ and $a = \gcd(A, B)$. □

Problem 11

Prove the correctness of the following **while** loop which implements a way to multiply two numbers and was developed by the ancient Egyptians.

Algorithm 11

Precondition: $A, B \in \mathbb{Z}^+$, $x = A$, $y = B$, and $product = 0$

```
while ($y \neq 0$) do
    $r := y \ mod \ 2$
    if $r = 0$ then
        $x := 2 \cdot x$
        $y := y \ div \ 2$
    end if
    if $r = 1$ then
        $product := product + x$
        $y := y - 1$
    end if
end while
```

Postcondition: $product = A \cdot B$

Loop invariant: $I(n)$ is "$xy + product = A \cdot B$"

Solution

Proof.

I. Basis Property: $I(0)$ is "$xy + product = A \cdot B$". But by the pre-conditions, $x = A$, $y = B$, and $product - 0$ and so $xy + product = AB + 0 = AB$. Thus $I(0)$ is true before the first iteration of the loop.

II. Inductive Property: Suppose k is some nonnegative integers such that after k iterations of the loop $G \wedge I(k)$. Since G is true $y \neq 0$ and the loop is entered. Also since $I(k)$ is true we have that before execution of the loop,

$$x_{\text{old}} y_{\text{old}} + product_{\text{old}} = A \cdot B$$

After execution of the loop,

Case 1 (y_{old} is even): In this case $r = 0$ since all even numbers are evenly divisible by 2. Now after execution,

$$x_{\text{new}} = 2 \cdot x_{\text{old}} \qquad y_{\text{new}} = \frac{y_{\text{old}}}{2} \qquad product_{\text{new}} = product_{\text{old}}$$

It follows that

$$\begin{aligned} x_{\text{new}} y_{\text{new}} + product_{\text{new}} &= 2 \cdot x_{\text{old}} \cdot \frac{y_{\text{old}}}{2} + product_{\text{old}} \\ &= x_{\text{old}} y_{\text{old}} + product_{\text{old}} \\ &= A \cdot B \end{aligned}$$

Case 1 (y_{old} is odd): In this case $r = 1$ since all odd numbers are equal to $2 \cdot \textit{some integer} + 1$. Now after execution,

$$x_{\text{new}} = x_{\text{old}} \qquad y_{\text{new}} = y_{\text{old}} - 1 \qquad product_{\text{new}} = product_{\text{old}} + x_{\text{old}}$$

It follows that

$$\begin{aligned} x_{\text{new}} y_{\text{new}} + product_{\text{new}} &= x_{\text{old}} \cdot (y_{\text{old}} - 1) + product_{\text{old}} + x_{\text{old}} \\ &= x_{\text{old}} y_{\text{old}} - x_{\text{old}} + x_{\text{old}} + product_{\text{old}} \\ &= x_{\text{old}} y_{\text{old}} + product_{\text{old}} \\ &= A \cdot B \end{aligned}$$

Thus $I(k+1)$ is true.

III. Eventual Falsity of the Guard: The guard G is the condition $y \neq 0$ and each value of y obtained by repeated iteration of the loop is nonnegative and less than the previous value of y. Thus, by the well-ordering principle, there is a least value y_{min}. The fact is that $y_{\text{min}} = 0$. Suppose not. That is, suppose that $y_{\text{min}} \neq 0$. Then since $y \in \mathbb{Z}^{nonneg}$, it must be that $y_{\text{min}} > 0$ and so the loop will run again. But then a value of y will be calculated which will be smaller than y_{min} which is a contradiction of the fact that y_{min} is the least value of y obtained by repeated iteration of the loop. Hence $y_{\text{min}} = 0$. Since $y_{\text{min}} = 0$, the guard is false immediately following the loop iteration in which y_{min} is calculated.

IV. Correctness of the Post-Condition: Suppose that for some nonnegative integer N, G is false and $I(N)$ is true. Since G is false, $y = 0$, and since $I(N)$ is true,

$$xy + product = A \cdot B$$

Substituting $y = 0$ into the equation above gives

$$\begin{aligned} x(0) + product &= A \cdot B \\ product &= A \cdot B \end{aligned}$$ □

Problem 12

The following sentence could be added to the loop invariant for the Euclidean algorithm:

> There exist integers u, v, s, and t such that
> $a = uA + vB \quad \text{and} \quad b = sA + tB.$ 5.5.12

(a) Show that this sentence is a loop invariant for

Algorithm 12

```
1: while (b ≠ 0) do
2:     r = a mod b
3:     a := b
4:     b := r
5: end while
```

(b) Show that if initially $a = A$ and $b = B$, then sentence 5.5.12 is true before the first iteration of the loop.

(c) Explain how the correctness proof for the Euclidean algorithm together with the results of (a) and (b) above allow you to conclude that given any integers A and B with $A > B \geq 0$, there exist integers u and v so that $\gcd(A, B) = uA + vB$.

(d) By actually calculating u, v, s, and t at each stage of execution of the Euclidean algorithm, find integers u and v so that $\gcd(330, 156) = 330u + 156v$

Solution

(a) *Proof.* Suppose that the predicate $a = uA + vB \quad \text{and} \quad b = sA + tB$ is true before entry to the loop. Then,

$$a_{\text{old}} = u_{\text{old}}A + v_{\text{old}}B \quad \text{and} \quad b_{\text{old}} = s_{\text{old}}A + t_{\text{old}}B$$

After execution of the loop,

$$a_{\text{new}} = b_{\text{old}} = s_{\text{old}}A + t_{\text{old}}B \quad \text{and} \quad b_{\text{new}} = a_{\text{old}} \ \mathit{mod} \ b_{\text{old}}$$

It follows from the definition of *mod* that there exists an integer q such that

$$\begin{aligned} b_{\text{new}} &= a_{\text{old}} - b_{\text{old}}q \\ &= u_{\text{old}}A + v_{\text{old}}B - (s_{\text{old}}A + t_{\text{old}}B)q \\ &= u_{\text{old}}A + v_{\text{old}}B - s_{\text{old}}Aq - t_{\text{old}}Bq \\ &= u_{\text{old}}A - s_{\text{old}}Aq + v_{\text{old}}B - t_{\text{old}}Bq \\ &= (u_{\text{old}} - s_{\text{old}}q)A + (v_{\text{old}} - t_{\text{old}}q)B \end{aligned}$$ □

(b) Suppose that initially, $a = A$ and $b = B$. let $u = t = 1$ and let $v = s = 0$. Then,

$$a = 1 \cdot A + 0 \cdot B = A \quad \text{and} \quad b = 0 \cdot A = 1 \cdot B = B$$

Hence sentence 5.5.12 is true before the first iteration of the loop.

(c) From (a) we showed that if sentence 5.1.12 is true before the iteration of the loop then it will be true after the iteration of the loop. From (b) we showed that if algorithm 12 runs with the same relevant pre-conditions as the Euclidean algorithm then sentence 5.5.12 will be true before the first iteration of the loop. Now the proof of correctness of the Euclidean algorithm guarantees that after a finite number of iterations the guard G will be false and that when this occurs, $a = \gcd(A, B)$. We also know from the results of (a) and (b) that there exist integers u and v such that $a = uA + vB$. It now follows from transitivity of equality on a that $\gcd(A, B) = uA + vB$.

(d)

Euclidean Algorithm Trace Table with $a = A = 330$, and $b = B = 156$

	0	**1**	**2**	**3**	**4**	**end**
A	330					
B	156					
r	156	18	12	6	0	
a	330	156	18	12	6	
b	156	18	12	6	0	
u	1	0	1	-8	9	
v	0	1	-2	17	-19	
s	0	1	-8	9	-26	
t	1	-2	17	-19	55	
gcd						6

It follows from the trace table above that the $\gcd(330, 156) = 6$. Now let $u = 9$ and let $v = -19$ and then $6 = 330(9) + 156(-19)$. Thus

gcd(330, 156) = 330(9) + 156(-19)