Answers Exam Program Correctness, April, 7th 2015.

Problem 1 (20 pt). Design an annotated command S that satisfies the Hoare triple:

$$\{ \ P : \ X \geq 0 \ \wedge \ (p-2 = X \ \vee \ p = -X) \ \wedge \ p^2 + q = Y \ \} \ \ S \ \ \{ \ Q : \ p = X \ \wedge \ p^2 + q = Y \ \}$$

Answer:

$\{ \ P : \ X \geq 0 \ \wedge \ (p-2 = X \ \vee \ p = -X) \ \wedge \ p^2 + q = Y \ \}$
  (∗ logic ∗)
$\{ \ (p-2 = X \geq 0 \ \vee \ -p = X \geq 0) \ \wedge \ p^2 + q = Y \ \}$
if $p \geq 2$ **then**
  $\{ \ p \geq 2 \ \wedge \ (p-2 = X \geq 0 \ \vee \ -p = X \geq 0) \ \wedge \ p^2 + q = Y \ \}$
    (∗ $p \geq 2 \Rightarrow -p < 0$; logic ∗)
  $\{ \ p-2 = X \ \wedge \ p^2 + q = Y \ \}$
    (∗ prepare $p := p-2$ ∗)
  $\{ \ p-2 = X \ \wedge \ (p-2+2)^2 + q = Y \ \}$
 $p := p-2;$
  $\{ \ p = X \ \wedge \ (p+2)^2 + q = Y \ \}$
    (∗ calculus ∗)
  $\{ \ p = X \ \wedge \ p^2 + 4 \cdot p + 4 + q = Y \ \}$
 $q := 4 * p + 4 + q;$
  $\{ \ p = X \ \wedge \ p^2 + q = Y \ \}$
else
  $\{ \ p < 2 \ \wedge \ (p-2 = X \geq 0 \ \vee \ -p = X \geq 0) \ \wedge \ p^2 + q = Y \ \}$
    (∗ logic ∗)
  $\{ \ -p = X \ \wedge \ p^2 + q = Y \ \}$
    (∗ prepare $p := -p$; Note that $p^2 = (-p)^2$ ∗)
  $\{ \ -p = X \ \wedge \ (-p)^2 + q = Y \ \}$
 $p := -p;$
  $\{ \ p = X \ \wedge \ p^2 + q = Y \ \}$
end; (∗ collect branches ∗)
 $\{ \ Q : \ p = X \ \wedge \ p^2 + q = Y \ \}$

Problem 2 (30 pt). Design and prove the correctness of a command T that satisfies

const $n : \mathbb{N}$, a : **array** $[0..n)$ **of** $\mathbb{Z}$;
var $z : \mathbb{Z}$;
  $\{ \ P :$ **true** $\}$
T
  $\{ \ Q : z = \Pi(\Sigma(a[j] \cdot a[k] \mid j,k : 0 \leq j \leq k < i) \mid i : 0 \leq i < n) \ \}$.

The time complexity of the command S must be linear in n. Start by defining (a) suitable helper function(s) and the corresponding recurrence(s).

Answer: We start by introducing $F(x) = \Pi(\Sigma(a[j] \cdot a[k] \mid j,k : 0 \leq j \leq k < i) \mid i : 0 \leq i < x)$ such that we can rewrite the postcondition as

$$Q : \ z = F(n)$$

It is clear that $F(0) = 1$ (product over empty domain). In a loop, we will increment x, so we are interested in a recurrence for $F(x+1)$.

  $F(x+1)$
$= \{$ definition $F\}$
  $\Pi(\Sigma(a[j] \cdot a[k] \mid j,k : 0 \leq j \leq k < i) \mid i : 0 \leq i < x+1)$

$= \{$ **assume** $x \geq 0$; split $i < x$ or $i = x$ $\}$
$\Pi(\Sigma(a[j] \cdot a[k] \mid j,k : 0 \leq j \leq k < i) \mid i : 0 \leq i < x) \cdot \Sigma(a[j] \cdot a[k] \mid j,k : 0 \leq j \leq k < x)$
$= \{$ definition $F\}$
$F(x) \cdot \Sigma(a[j] \cdot a[k] \mid j,k : 0 \leq j \leq k < x)$
$= \{$ **introduce** $S(x) = \Sigma(a[j] \cdot a[k] \mid j,k : 0 \leq j \leq k < x)$ $\}$
$F(x) \cdot S(x)$

It is clear that $S(0) = 0$ (sum over empty domain). We are interested in a recurrence for $S(x+1)$.

$S(x+1)$
$= \{$ definition $S\}$
$\Sigma(a[j] \cdot a[k] \mid j,k : 0 \leq j \leq k < x+1)$
$= \{$ **assume** $x \geq 0$; split $k < x$ or $k = x$ $\}$
$\Sigma(a[j] \cdot a[k] \mid j,k : 0 \leq j \leq k < x) \ + \ \Sigma(a[j] \cdot a[x] \mid j : 0 \leq j \leq x)$
$= \{$ definition S; calculus$\}$
$S(x) \ + \ a[x] \cdot \Sigma(a[j] \mid j : 0 \leq j \leq x)$
$= \{$ **introduce** $T(x) = \Sigma(a[j] \mid j : 0 \leq j < x)$ $\}$
$S(x) \ + \ a[x] \cdot T(x+1)$

It is clear that $T(0) = 0$ (sum over empty domain). We are interested in a recurrence for $T(x+1)$.

$T(x+1)$
$= \{$ definition $T\}$
$\Sigma(a[j] \mid j,k : 0 \leq j \leq k < x+1)$
$= \{$ **assume** $x \geq 0$; split $k < x$ or $k = x$ $\}$
$\Sigma(a[j] \mid j : 0 \leq j \leq k < x) \ + \ a[x]$
$= \{$ definition $S\}$
$T(x) \ + \ a[x]$

We can now introduce the invariant: $J: \ z = F(x) \ \wedge \ s = S(x) \ \wedge \ t = T(x) \ \wedge \ 0 \leq x \leq n$.
Clearly, we choose the guard $B: \ x \neq n$, such that $J \ \wedge \ \neg B \Rightarrow Q$
For the variant function we choose $\mathsf{vf} = n - x \in \mathbb{Z}$. Clearly $J \Rightarrow \mathsf{vf} \geq 0$.
Initialization of the invariant is easy:

$\{$ **true** $\}$
(∗ base cases recurrences; $n \in \mathbb{N}$ ∗)
$\{ 1 = F(0) \ \wedge \ 0 = S(0) \ \wedge \ 0 = T(0) \ \wedge \ 0 \leq 0 \leq n \}$.
$z := 1;\ s := 0;\ t := 0;\ x := 0;$
$\{J: \ z = F(x) \ \wedge \ s = S(x) \ \wedge \ t = T(x) \ \wedge \ 0 \leq x \leq n \}$

We now turn to the derivation of the body of the while-loop.

$\{J \ \wedge \ B \ \wedge \ \mathsf{vf} = V\}$
(∗ definitions J, B, and vf ∗)
$\{z = F(x) \ \wedge \ s = S(x) \ \wedge \ t = T(x) \ \wedge \ 0 \leq x < n \ \wedge \ n - x = V\}$
(∗ recurrence $F(x+1)$; substitution ∗)
$\{z \cdot s = F(x+1) \ \wedge \ s = S(x) \ \wedge \ t = T(x) \ \wedge \ 0 \leq x < n \ \wedge \ n - x = V\}$
$z := z * s;$
$\{z = F(x+1) \ \wedge \ s = S(x) \ \wedge \ t = T(x) \ \wedge \ 0 \leq x < n \ \wedge \ n - x = V\}$
(∗ recurrence $T(x+1)$; substitution ∗)
$\{z = F(x+1) \ \wedge \ s = S(x) \ \wedge \ t + a[x] = T(x+1) \ \wedge \ 0 \leq x < n \ \wedge \ n - x = V\}$
$t := t + a[x];$
$\{z = F(x+1) \ \wedge \ s = S(x) \ \wedge \ t = T(x+1) \ \wedge \ 0 \leq x < n \ \wedge \ n - x = V\}$
(∗ recurrence $S(x+1)$; substitution ∗)
$\{z = F(x+1) \ \wedge \ s + a[x] \cdot t = S(x+1) \ \wedge \ t = T(x+1) \ \wedge \ 0 \leq x < n \ \wedge \ n - x = V\}$
$s := s + a[x] * t;$

$\{z = F(x+1) \ \wedge\ s = S(x+1) \ \wedge\ t = T(x+1) \ \wedge\ 0 \le x < n \ \wedge\ n - x = V\}$
(∗ prepare $x := x + 1$; calculus ∗)
$\{z = F(x+1) \ \wedge\ s = S(x+1) \ \wedge\ t = T(x+1) \ \wedge\ 0 \le x+1 \le n \ \wedge\ n - (x+1) < V\}$
$x := x + 1;$
$\{J \ \wedge\ \mathsf{vf} < V : \ z = F(x) \ \wedge\ s = S(x) \ \wedge\ t = T(x) \ \wedge\ 0 \le x \le n \ \wedge\ n - x < V\}$

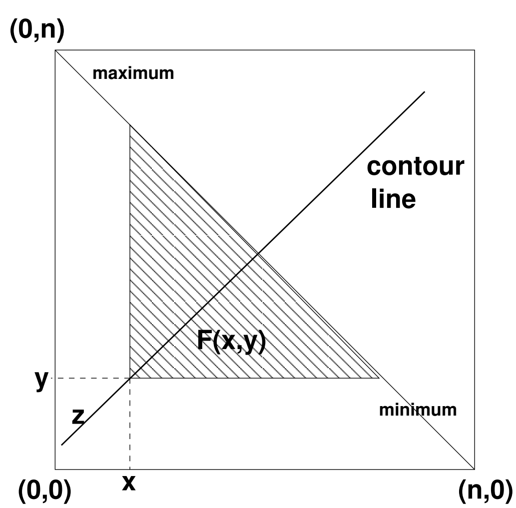
We completed the proof. We found the following program fragment:

const $n : \mathbb{N}, \quad a :$ **array** $[0..n)$ **of** $\mathbb{Z}$;
var $z, \ s, \ t, \ x : \mathbb{Z}$;
$\{ \ P :$ **true** $\}$
$z := 1;$
$s := 0;$
$t := 0;$
$x := 0;$
$\{J : \ z = F(x) \ \wedge\ s = S(x) \ \wedge\ t = T(x) \ \wedge\ 0 \le x \le n \ \}$
(∗ $\mathsf{vf} = n - x$ ∗)
while $x \neq n$ **do**
$z := z * s;$
$t := t + a[x];$
$s := s + a[x] * t;$
$x := x + 1;$
end;
$\{Q : \ z = F(n)\}$

Problem 3 (40 pt). Given is a two-dimensional array a that is *decreasing* in its first argument and *ascending* in its second argument. Consider the following specification:

const $n, \ w : \mathbb{N}, \quad a :$ **array** $[0..n)$ **of** $\mathbb{N}$;
var $z : \mathbb{N}$;
$\{P : \ Z = \#\{(i,j) \mid i,j : 0 \le i \ \wedge\ 0 \le j \ \wedge\ i + j < n \ \wedge\ a[i,j] = w\} \ \}$
U
$\{Q : \ Z = z\}$

(a) Make a sketch in which you clearly indicate where the array is high, low, and how a contour line goes.



(b) Define a function $F(x,y)$ that can be used to compute Z. Determine the relevant recurrences for $F(x,y)$, including the base cases.
Answer: We define the function $F(x,y) = \#\{(i,j) \mid i,j : x \le i \wedge y \le j \wedge i{+}j < n \wedge a[i,j] = w\}$. It is clear that $x+y \ge n \Rightarrow F(x,y) = 0$. To reduce the triangular area (see sketch), we need to increment x or increment y. We first have a look at an increment of x:

$$
\begin{array}{ll}
 & F(x,y) \\
= & \{ \text{ definition } F\} \\
 & \#\{(i,j) \mid i,j : x \le i \ \wedge\ y \le j \ \wedge\ i+j < n \ \wedge\ a[i,j] = w\} \\
= & \{ \textbf{ assume } x+y<n\text{; so domain non-empty; split } i = x \text{ or } x+1 \le i \ \} \\
 & \#\{(i,j) \mid i,j : x+1 \le i \ \wedge\ y \le j \ \wedge\ i+j < n \ \wedge\ a[i,j] = w\} \ + \\
 & \#\{j \mid j : y \le j \ \wedge\ x+j < n \ \wedge\ a[x,j] = w\} \\
= & \{ \text{ definition } F\text{; calculus}\} \\
 & F(x+1,y) \ + \ \#\{j \mid j : y \le j < n-x \ \wedge\ a[x,j] = w\} \\
= & \{ \ a[x,j] \text{ is ascending in } j\text{; } a[x,y] \text{ is minimal; } \textbf{assume } a[x,y] > w\text{; then } a[x,j] > w \text{ for } j \ge y \ \} \\
 & F(x+1,y)
\end{array}
$$

Next we investigate an increment of y:

$$
\begin{array}{ll}
 & F(x,y) \\
= & \{ \text{ definition } F\} \\
 & \#\{(i,j) \mid i,j : x \le i \ \wedge\ y \le j \ \wedge\ i+j < n \ \wedge\ a[i,j] = w\} \\
= & \{ \textbf{ assume } x+y<n\text{; so domain non-empty; split } j = y \text{ or } y+1 \le j \ \} \\
 & \#\{(i,j) \mid i,j : x \le i \ \wedge\ y+1 \le j \ \wedge\ i+j < n \ \wedge\ a[i,j] = w\} \ + \\
 & \#\{i \mid i : x \le i \ \wedge\ i+y < n \ \wedge\ a[i,y] = w\} \\
= & \{ \text{ definition } F\text{; calculus}\} \\
 & F(x,y+1) \ + \ \#\{i \mid i : x \le i < n-y \ \wedge\ a[i,y] = w\} \\
= & \{ \ a[i,y] \text{ is decreasing in } i\text{; } a[x,y] \text{ is maximal; } \textbf{assume } a[x,y] \le w\text{; then } a[i,y] < w \text{ for } i > x \ \} \\
 & F(x,y+1) + \mathsf{ord}(a[x,y] = w)
\end{array}
$$

In conclusion, we found the following recurrence relation for $F(x,y)$:

$$
\begin{array}{rcl}
x+y \ge n & \Rightarrow & F(x,y) = 0 \\
x+y < n \ \wedge\ a[x,y] > w & \Rightarrow & F(x,y) = F(x+1,y) \\
x+y < n \ \wedge\ a[x,y] \le w & \Rightarrow & F(x,y) = F(x,y+1) + \mathsf{ord}(a[x,y] = w)
\end{array}
$$

(c) Design a command U that has a linear time complexity in n. Prove the correctness of your solution.

Answer: The precondition can be rewritten as: $P: \ Z = F(0,0)$. We introduce the invariant, guard, and variant function:

$$
\begin{array}{rcl}
J & : & Z = z + F(x,y) \\
B & : & x+y < n \\
\mathsf{vf} & = & n-x-y \in \mathbb{Z}
\end{array}
$$

Clearly, $J \ \wedge\ \neg B \Rightarrow Z = z$. It is also clear that $B \Rightarrow \mathsf{vf} \ge 0$. The invariant is easy to initialize:

$\{P: \ Z = F(0,0) \ \}$
$(*\text{ calculus } *)$
$\{ \ Z = 0 + F(0,0) \ \}.$
$z := 0; \ x := 0; \ y := 0;$
$\{J: \ Z = z + F(x,y) \ \}$

We now turn to the derivation of the body of the while-loop.

```
    {J ∧ B ∧ vf = V}
      (* definitions J, B, and vf *)
    {Z = z + F(x,y) ∧ x + y < n ∧ n − x − y = V}
if a[x,y] > w then
      {a[x,y] > w ∧ Z = z + F(x,y) ∧ x + y < n ∧ n − x − y = V}
        (* recurrence F(x+1,y); logic; calculus *)
      {Z = z + F(x+1,y) ∧ n − (x+1) − y < V}
  x := x + 1;
      {Z = z + F(x,y) ∧ n − x − y < V}
else
      {a[x,y] ≤ w ∧ Z = z + F(x,y) ∧ x + y < n ∧ n − x − y = V}
        (* recurrence F(x,y+1); logic; calculus *)
      {Z = z + ord(a[x,y] = w) + F(x,y+1) ∧ n − x − (y+1) < V}
  z := z + ord(a[x,y] = w);
      {Z = z + F(x,y+1) ∧ n − x − (y+1) < V}
  y := y + 1;
      {Z = z + F(x,y) ∧ n − x − y < V}
end; (* collect branches *)
    {J ∧ vf < V : Z = z + F(x,y) ∧ n − x − y < V}
```

We completed the proof. We found the following program fragment:

```
const n, w : ℕ,   a : array [0..n) of ℕ;
var x, y, z : ℕ;
  {P : Z = #{(i,j) | i,j : 0 ≤ i ∧ 0 ≤ j ∧ i + j < n ∧ a[i,j] = w} }
x := 0;
y := 0;
z := 0;
  {J : Z = z + #{(i,j) | i,j : x ≤ i ∧ y ≤ j ∧ i + j < n ∧ a[i,j] = w} }
    (* vf = n − x − y *)
while x + y < n do
  if a[x,y] > w then
    x := x + 1;
  else
    z := z + ord(a[x,y] = w);
    y := y + 1;
  end;
end;
  {Q : Z = z}
```