Answers Exam Program Correctness, April, 1st, 2016.

Problem 1 (20 pt). Declared are the variables $a,\ b,\ n : \mathbb{N}$. Design an annotated command S that satisfies the Hoare triple:

$$\{\ b \cdot a^n = X\ \wedge\ 2 \cdot Y \le n < 2 \cdot (Y+1)\ \}\ \ S\ \ \{\ b \cdot a^n = X\ \wedge\ n = Y\ \}$$

You are not allowed to use a loop.

Answer:

$\{\ b \cdot a^n = X\ \wedge\ 2 \cdot Y \le n < 2 \cdot (Y+1)\ \}$
(∗ calculus; $n = 2 \cdot (n\ \mathbf{div}\ 2) + n\ \mathbf{mod}\ 2$ ∗)
$\{\ b \cdot a^{2 \cdot (n\ \mathbf{div}\ 2) + n\ \mathbf{mod}\ 2} = X\ \wedge\ 2 \cdot Y \le n < 2 \cdot (Y+1)\ \}$
(∗ calculus ∗)
$\{\ b \cdot a^{n\ \mathbf{mod}\ 2} \cdot a^{2 \cdot (n\ \mathbf{div}\ 2)} = X\ \wedge\ 2 \cdot Y \le n < 2 \cdot (Y+1)\ \}$
(∗ calculus ∗)
$\{\ b \cdot (n\ \mathbf{mod}\ 2 = 0\ ?\ 1 : a) \cdot a^{2 \cdot (n\ \mathbf{div}\ 2)} = X\ \wedge\ 2 \cdot Y \le n < 2 \cdot (Y+1)\ \}$
$b := b * (n\ \mathbf{mod}\ 2 = 0\ ?\ 1 : a);$
$\{\ b \cdot a^{2 \cdot (n\ \mathbf{div}\ 2)} = X\ \wedge\ 2 \cdot Y \le n < 2 \cdot (Y+1)\ \}$
(∗ calculus ∗)
$\{\ b \cdot (a \cdot a)^{n\ \mathbf{div}\ 2} = X\ \wedge\ 2 \cdot Y \le n < 2 \cdot (Y+1)\ \}$
$a := a * a;$
$\{\ b \cdot a^{n\ \mathbf{div}\ 2} = X\ \wedge\ 2 \cdot Y \le n < 2 \cdot (Y+1)\ \}$
(∗ calculus ∗)
$\{\ b \cdot a^{n\ \mathbf{div}\ 2} = X\ \wedge\ Y \le n\ \mathbf{div}\ 2 < Y+1\ \}$
$n := n\ \mathbf{div}\ 2;$
$\{\ b \cdot a^n = X\ \wedge\ y \le n < Y+1\ \}$
(∗ calculus ∗)
$\{\ b \cdot a^n = X\ \wedge\ n = Y\ \}$

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

const $n : \mathbb{Z},\ a :$ **array** $[0..n)$ **of** $\mathbb{Z}$;
var $z : \mathbb{Z}$;
$\{\ P : n > 0\ \}$
T
$\{\ Q : z = \mathsf{Max}\ (\mathsf{Min}\ (a[i] + a[j] \mid i,j : 0 \le i \le j \le k) \mid k : 0 \le k < n)\ \}$.

The time complexity of the command S must be linear in n. You are not allowed to use the values $\pm\infty$ in the program. Start by defining one or more suitable helper functions with corresponding recurrences.

Answer: We introduce $F(x) = \mathsf{Max}\ (\mathsf{Min}\ (a[i] + a[j] \mid i,j : 0 \le i \le j \le k) \mid k : 0 \le k < x)$ such that we can rewrite the postcondition as

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

Since the values $\pm\infty$ are not allowed in the program, we use the base case

$$F(1) = \mathsf{Min}\ (a[i] + a[j] \mid i,j : 0 \le i \le j \le 0) = a[0] + a[0] = 2 \cdot a[0]$$

In a loop, we will increment x, so we are interested in a recurrence for $F(x+1)$.

$F(x+1)$
$= \{$ definition $F\}$
$\mathsf{Max}\ (\mathsf{Min}\ (a[i]+a[j] \mid i,j : 0 \le i \le j \le k) \mid k : 0 \le k < x+1)$
$= \{$ **assume** $0 \le x < n$; split $k < x$ or $k = x$ $\}$
$\mathsf{Max}\ (\mathsf{Min}\ (a[i]+a[j] \mid i,j : 0 \le i \le j \le k) \mid k : 0 \le k < x)$ **max** $\mathsf{Min}\ (a[i]+a[j] \mid i,j : 0 \le i \le j \le x)$
$= \{$ definition F; use half-open intervals $\}$
$F(x)$ **max** $\mathsf{Min}\ (a[i]+a[j] \mid i,j : 0 \le i \le j < x+1)$
$= \{$ **introduce** $G(x) = \mathsf{Min}\ (a[i]+a[j] \mid i,j : 0 \le i \le j < x)$ $\}$
$F(x)$ **max** $G(x+1)$

It is clear that $G(1) = 2 \cdot a[0]$ as well. We are interested in a recurrence for $G(x+1)$.

$G(x+1)$
$= \{$ definition $G\}$
$\mathsf{Min}\ (a[i]+a[j] \mid i,j : 0 \le i \le j < x+1)$
$= \{$ **assume** $0 \le x < n$; split $j < x$ or $j = x$ $\}$
$\mathsf{Min}\ (a[i]+a[j] \mid i,j : 0 \le i \le j < x)$ **min** $\mathsf{Min}\ (a[i]+a[x] \mid i : 0 \le i \le x)$
$= \{$ definition G; calculus; use half-open intervals$\}$
$G(x)$ **min** $(a[x] + \mathsf{Min}\ (a[i] \mid i : 0 \le i < x+1))$
$= \{$ **introduce** $H(x) = \mathsf{Min}\ (a[i] \mid i : 0 \le i < x)$ $\}$
$G(x)$ **min** $(a[x] + H(x+1))$

It is clear that $H(1) = a[0]$. We are interested in a recurrence for $H(x+1)$.

$H(x+1)$
$= \{$ definition $H\}$
$\mathsf{Min}\ (a[i] \mid i : 0 \le i < x+1)$
$= \{$ **assume** $0 \le x < n$; split $i < x$ or $i = x$ $\}$
$\mathsf{Min}\ (a[i] \mid i : 0 \le i < x)$ **min** $a[x]$
$= \{$ definition H $\}$
$H(x)$ **min** $a[x]$

We can now introduce the invariant: $J:\ z = F(x) \ \wedge\ g = G(x) \ \wedge\ h = H(x) \ \wedge\ 1 \le x \le n$.
Clearly, we choose the guard $B:\ x \ne n$, such that $J \ \wedge\ \neg B \Rightarrow Q$
For the variant function we choose $\mathsf{vf} = n - x \in \mathbb{Z}$. Clearly $J \ \wedge\ B \Rightarrow J \Rightarrow \mathsf{vf} \ge 0$.
Initialization of the invariant is easy:

$\{\ n > 0\}$
(∗ base cases recurrences; $n > 0$ so $a[0]$ exists ∗)
$\{\ 2 \cdot a[0] = F(1) \ \wedge\ 2 \cdot a[0] = G(1) \ \wedge\ a[0] = H(1) \ \wedge\ 1 \le 1 \le n\ \}$.
$z := 2 * a[0];\ g := 2 * a[0];\ h := a[0];\ x := 1;$
$\{J:\ z = F(x) \ \wedge\ g = G(x) \ \wedge\ h = H(x) \ \wedge\ 1 \le x \le n\ \}$

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 ∗)
$\{z = F(x) \ \wedge\ g = G(x) \ \wedge\ h = H(x) \ \wedge\ 1 \le x < n \ \wedge\ n - x = V\}$
(∗ recurrence $H(x+1)$; substitution ∗)
$\{z = F(x) \ \wedge\ g = G(x) \ \wedge\ h \ \mathbf{min}\ a[x] = H(x+1) \ \wedge\ 1 \le x < n \ \wedge\ n - x = V\}$
$h := h\ \mathbf{min}\ a[x];$
$\{z = F(x) \ \wedge\ g = G(x) \ \wedge\ h = H(x+1) \ \wedge\ 1 \le x < n \ \wedge\ n - x = V\}$
(∗ recurrence $G(x+1)$; substitution ∗)
$\{z = F(x) \ \wedge\ g\ \mathbf{min}\ (a[x] + h) = G(x+1) \ \wedge\ h = H(x+1) \ \wedge\ 1 \le x < n \ \wedge\ n - x = V\}$
$g := g\ \mathbf{min}\ (a[x] + h);$
$\{z = F(x) \ \wedge\ g = G(x+1) \ \wedge\ h = H(x+1) \ \wedge\ 1 \le x < n \ \wedge\ n - x = V\}$
(∗ recurrence $F(x+1)$; substitution ∗)

$\{z \ \mathbf{max}\ g = F(x+1)\ \wedge\ g = G(x+1)\ \wedge\ h = H(x+1)\ \wedge\ 1 \le x < n\ \wedge\ n - x = V\}$

$z := z\ \mathbf{max}\ g;$

$\{z = F(x+1)\ \wedge\ g = G(x+1)\ \wedge\ h = H(x+1)\ \wedge\ 1 \le x < n\ \wedge\ n - x = V\}$

(∗ prepare $x := x + 1$; calculus ∗)

$\{z = F(x+1)\ \wedge\ g = G(x+1)\ \wedge\ h = H(x+1)\ \wedge\ 1 \le x < n\ \wedge\ n - (x+1) < V\}$

$x := x + 1;$

$\{J\ \wedge\ \mathsf{vf} < V:\ z = F(x)\ \wedge\ g = G(x)\ \wedge\ h = H(x)\ \wedge\ 1 \le x < n\ \wedge\ n - x < V\}$

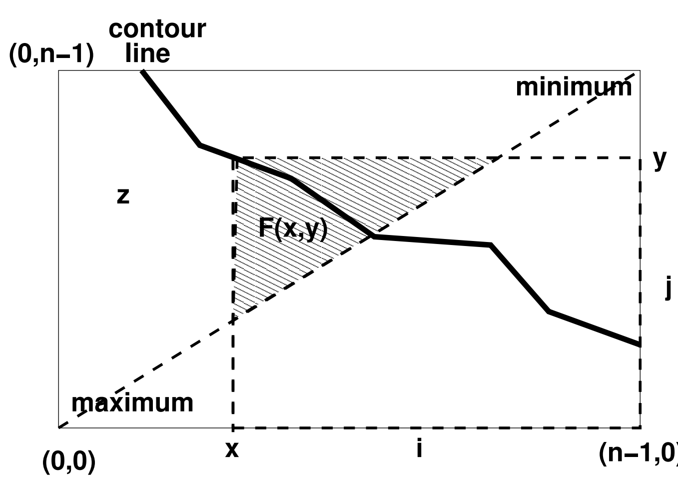
We completed the proof. We found the following program fragment:

```
const n : ℤ, a : array [0..n) of ℤ;
var z, s, t, x : ℤ;
  { P : n > 0 }
  z := 2 ∗ a[0];
  g := 2 ∗ a[0];
  h := a[0];
  x := 1;
    {J : z = F(x) ∧ g = G(x) ∧ h = H(x) ∧ 1 ≤ x ≤ n }
      (∗ vf = n − x ∗)
  while x ≠ n do
    h := h min a[x];
    g := g min (a[x] + h);
    z := z max g;
    x := x + 1;
  end;
    {Q : z = F(n)}
```

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

```
const n, w : ℕ, a : array [0..n) of ℕ;
  var z : ℕ;
    {P : Z = #{(i, j) | i, j : 0 ≤ i ≤ j < n ∧ 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 $F(x,y) = \#\{(i,j) \mid i,j : x \le i \le j < y \le n \ \wedge \ a[i,j] = w\}$.
It is clear that $x \ge y \Rightarrow F(x,y) = 0$. To reduce the triangular area (see sketch), we need to increment x or decrement y. We first have a look at an increment of x:

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

Next we investigate a decrement of y:

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

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

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

(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: $P: \ Z = F(0,n)$. We introduce the invariant, guard, and variant function:

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

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:

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

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 ∧ y − x = V}
if a[x, y − 1] ≥ w then
      {a[x, y − 1] ≥ w ∧ Z = z + F(x, y) ∧ x < y ∧ y − x = V}
        (* recurrence F(x + 1, y); logic; calculus *)
      {Z = z + ord(a[x, y − 1] = w) + F(x + 1, y) ∧ y − (x + 1) < V}
    z := z + ord(a[x, y] = w);
      {Z = z + F(x + 1) ∧ y − (x + 1) < V}
    x := x + 1;
      {Z = z + F(x, y) ∧ y − x < V}
else
      {a[x, y − 1] < w ∧ Z = z + F(x, y) ∧ x < y ∧ y − x = V}
        (* recurrence F(x, y − 1); logic; calculus *)
      {Z = z + F(x, y − 1) ∧ (y − 1) − x < V}
    y := y − 1;
      {Z = z + F(x, y) ∧ y − x < V}
end; (* collect branches *)
    {J ∧ vf < V : Z = z + F(x, y) ∧ y − x < 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 ≤ j < n ∧ a[i, j] = w} }
x := 0;
y := n;
z := 0;
  {J : Z = z + #{(i, j) | i, j : x ≤ i ≤ j < n ∧ a[i, j] = w} }
    (* vf = y − x *)
while x < y do
  if a[x, y − 1] ≥ w then
    z := z + ord(a[x, y − 1] = w);
    x := x + 1;
  else
    y := y − 1;
  end;
end;
  {Q : Z = z}
```