COS4892 Assignment 2 - 547106

Christopher Deon Steenkamp - 36398934

Question 1

1.1

1.2

1.3

1.4

Question 2

2.1

The program statement is a sequential composition of the form $\{P\}S1; S2; S3\{Q\}$

We are given that $\{Q\}$: $\{i = j\}$ $S1$: $i := 1 + 1$ $S2$: $k := k * 2$ $S3$: $j := j - 1$

Working backwards from the postcondition $\{i = j\}$, we apply the assignment axiom to $S3$ by replacing j with $j - 1$. From this we get the intermediate assertion R: $\{i = j - 1\}$.

Using R as the postconditon of $S2$, we again work backwards and apply the assignment axiom to $S2$. The variable assignment to k has no effect on our postcondition so the assertion R is invariant over $S2$.

We then use R: $\{i = j - 1\}$ as the postcondition of $S1$, and applying the assignment axiom by replacing i with $i + 1$ gives us the assertion P: $\{i + 1 = j - 1\}$, which is the precondition for the program statement.

2.2

I think there is an error in the question because incrementing x after you have multiplied y by x will cause the loop invariant to fail ($y \neq x!$). For my solution I have swapped the order of the two assignment statements (and provided verification of the correctness of the updated program).

The corrected program statement in guarded command form is thus:

$\{n = 4\}$
$x, y := 0, 1;$
$\{$Invariant: $0 \leq x \leq n \wedge y = x!$
Bound function: $n - x\}$
do $x < n \longrightarrow x := x + 1;\ y := y * x$
od
$\{x = 4 \wedge y = n!\}$

$\neg done$ is defined as $x < n$, so the loop body runs when $x < n$.

To verify the termination condtion we show that the conjunction of the invariant and *done* implies the postcondition. So $x \geq n \wedge 0 \leq x \leq n \wedge y = x!$. Also, $x \geq 0 \wedge x \leq n$ together imply that $x = n$ so we have that the loop terminates correctly.

To verify initialization assignments, we show that the loop invariant is a postcondition of assignment. We use the assignment axiom and the initial assignment values to show that:

$n \geq 0 \equiv 0 \leq 0 \leq n \wedge 1 = 0!$

We then check the loop body using $P = \{invariant \wedge \neg done \wedge bf\}$, working backwards from the post-condition using the assignment axiom. Note that we do two verfications because there are two sequential statements.

1. $\{Q\} \Rightarrow \{R\}$ where $R = Q[y := y * x]$
   $\{0 \leq x \leq n \wedge y * x = x! \wedge x < n \wedge n - x < C\}$
2. $\{R\} \Rightarrow \{R2\}$ where $R2 = R[x := x + 1]$
   $\{0 \leq x + 1 \leq n \wedge y * (x + 1) = (x + 1) * x! \wedge n - (x + 1) < C\}$

Which satisfies the final condition, with $n = 4, x = 4, y = n! = 24$

2.3

The given program statement in guarded-command format is as follows:

$\{P\}$
if $even(x) \longrightarrow \{P \wedge even(x)\}\ x := x - 1\ \{x \geq 0 \wedge z + y * x = a * b\}$
$\Box\ odd(x) \longrightarrow \{P \wedge odd(x)\}\ z := z + yx\ \{x \geq 0 \wedge z + y * x = a * b\}$
fi
$\{x \geq 0 \wedge z + y * x = a * b\}$

We also note that $[P \Rightarrow even(x) \vee odd(x)]$

Taking the first statement from the conditional, (if $even(x)$), and applying the assignment axiom, we get the following precondition:

$\{x - 1 \geq 0 \wedge z + y(x - 1) = a * b \wedge even(x)\}$

Applying the assignment axiom to the second statement of the conditional gives us the following precondition:

$\{x \geq 0 \wedge (z + yx) + yx = a * b \wedge odd(x)\}$

Combining the two preconditions we get $\{P\} = \{x \geq 1 \wedge z + yx - y = a * b \wedge z + 2yx = a * b\}$

2.4

$\{P\}$ do $0 \geq x \geq -2 \longrightarrow x := x - 1$ od $\{x = -3\}$

Starting with the postcondition Q, we iteratively use the assignment axiom until we get to the point that the loop is out of bounds.

1. $Q[x := x - 1]$ gives us $x - 1 = -3$ so $R1 = \{x = -2\}$
2. $R1[x := x - 1]$ gives us $x - 1 = -2$ so $R2 = \{x = -1\}$
3. $R2[x := x - 1]$ gives us $x - 1 = -1$ so $R3 = \{x = 0\}$
4. $R3[x := x - 1]$ gives us $x - 1 = 0$ so $R4 = \{x = 1\}$ which is out of bounds so the last valid state was $R3$.

This gives us that the precondition P is $\{x = 0\}$

Question 3

To verify a Hoare triple $\{P\}S\{Q\}$ we need to show that, given a precondition P, after executing some statement S, we have the postcondition Q. To this, we make use of the assignment axiom. The axiom is applied by working backwards from the postcondition, using the fact that, given a postcondition Q which is the state after execution of a statement S, we can “create” a precondition R by replacing all occurrences of variables in Q with their respective assignments from S.

For example, given the postcondition $Q = \{x = 2\}$ and the assignment statement $S = x := x * 2$ we can create a precondition R such that $R = Q[x := x * 2]$ which means that we replace occurrences of x in Q with $x * 2$. This gives us $R = \{x * 2 = 2\}$, which we simplify using arithmetic to $\{x = 1\}$.

Clearly, by starting at R, with $x = 1$, after setting $x := x * 2$ $(x := 1 * 2)$, we have Q, $\{x = 2\}$.

From here, if P and R can be shown equivalent, then we have verified the Hoare triple $\{P\}S\{Q\}$.

Question 4

4.1

$\{x = y\}$
if $x = 0 \longrightarrow \{x = y \wedge x = 0\}x := y + 1\{(x = y + 1) \vee (z = x + 1)\}$
$\Box$ $x \neq 0 \longrightarrow \{x = y \wedge x \neq 0\}z := y + 1\{(x = y + 1) \vee (z = x + 1)\}$
fi
$\{(x = y + 1) \vee (z = x + 1)\}$

We have that either $x = 0$ or $x \neq 0$ so we can verify correctness of the program by verifying the correctness of each statement.

We start by using the assignment axiom on the postcondition $\{(x = y + 1) \vee (z = x + 1)\}$ using the statement $x := y + 1$.

This gives us $\{(y + 1 = y + 1) \vee (z = (y + 1) + 1)\}$ (which is *true* by propositional calculus) so we have that:

$\{(y + 1 = y + 1) \vee (z = (y + 1) + 1)\}$
$=$ {replace equals for equals, $true \equiv p \equiv p$, with $p = y + 1$}
$\{(true) \vee (z = (y + 1) + 1)\}$
$=$ {$true$ is a zero of disjunction}
$true$

So we have $x = y \wedge x = 0 \equiv true$.

Applying the assignment axiom on the postcondition $\{(x = y+1) \vee (z = x+1)\}$ using the statement $z := y+1$ yields:

$\{(x = y + 1) \vee (y + 1 = x + 1)\}$
$=$ {we have that x = y from P, replacing equals for equals}
$\{(y = y + 1) \vee (y + 1 = y + 1)\}$
$=$ {replace equals for equals, $true \equiv p \equiv p$, with $p = y + 1$}
$\{(y = y + 1) \vee (true)\}$
$=$ {$true$ is a zero of disjunction}
$true$

So we have $\{x = y \wedge x \neq 0\} \equiv true$

This proves that the program correctly arrives at the postcondition from the given precondition.

4.2

$\{x = y\}$
if $x = 0 \longrightarrow \{x = y \wedge x = 0\}x := y + 1\{(z = 1) \rightarrow (x = 1)\}$
$\Box$ $x \neq 0 \longrightarrow \{x = y \wedge x \neq 0\}z := y + 1\{(z = 1) \rightarrow (x = 1)\}$
fi
$\{(z = 1) \rightarrow (x = 1)\}$

As is the case in q4.1, we have that either $x = 0$ or $x \neq 0$ so we can verify correctness of the program by verifying the correctness of each statement.

We start by using the assignment axiom on the postcondition $\{(z = 1) \rightarrow (x = 1)\}$ using the statement $x := y + 1$.

This gives us $\{(z = 1) \rightarrow (y + 1 = 1)\}$
$=$ {arithmetic}
$\{(z = 1) \rightarrow (y = 0)\}$
$=$ {assume $x = y = 0$, definition of only-if, with $p = (z = 1)$ and $q = (y = 0)$, $p \Rightarrow q \equiv q \equiv p \vee q$}

$\{(z = 1) \vee (y = 0)\}$
$= \{x = y, x = 0$, equals for equals$\}$
$\{(z = 1) \vee (true)\}$
$= \{true$ is a zero of disjunction$\}$
$true$

So we have $x = y \wedge x = 0 \equiv true$.

Applying the assignment axiom on the postcondition $\{(z = 1) \rightarrow (x = 1)\}$ using the statement $z := y + 1$ yields:

$\{(y + 1 = 1) \rightarrow (x = 1)\}$
$= \{$arithmetic$\}$
$\{(y = 0) \rightarrow (x = 1)\}$
$= \{$Logical equivalence $p \Rightarrow q \equiv \neg p \vee q\}$
$\{(y \neq 0) \vee (x = 1)\}$
$= \{x = y, x \neq 0$, equals for equals$\}$
$\{(true) \vee (x = 1)\}$
$= \{true$ is a zero of disjunction$\}$
$true$

So we have $\{x = y \wedge x \neq 0\} \equiv true$

This proves that the program correctly arrives at the postcondition from the given precondition.

4.3

$\{3 \leq |x| \leq 4\}$
if $x < 0 \longrightarrow \{3 \leq |x| \leq 4 \wedge x < 0\} y := -x \{2 \leq y \leq 4\}$
$\square$ $x \geq 0 \longrightarrow \{3 \leq |x| \leq 4 \wedge x \geq 0\} y := x \{2 \leq y \leq 4\}$
fi
$\{2 \leq y \leq 4\}$

We have that either $x < 0$ or $x \geq 0$ so we can verify correctness of the program by verifying the correctness of each statement.

We start by using the assignment axiom on the postcondition $\{2 \leq y \leq 4\}$ using the statement $y := -x$.

This gives us $\{2 \leq -x \leq 4\}$
$= \{$we have that $x < 0$ so we know that -x is positive$\}$
$\{2 \leq x \leq 4\}$
$= \{$equals for equals and arithmetic $x = |x|\}$
$\{2 \leq |x| \leq 4\}$
$= \{$Given that $3 \leq |x| \leq 4\}$
$\{2 \leq 3 \leq |x| \leq 4\}$
$= \{$arithmetic$\}$
$\{3 \leq |x| \leq 4\}$

So we have $3 \leq |x| \leq 4$ as required.

Applying the assignment axiom on the postcondition $\{2 \leq y \leq 4\}$ using the statement $y := x$ yields:

$\{2 \leq x \leq 4\}$
$= \{$we have that $x \geq 0$ so we know that x is positive, so equals for equals and arithmetic $x = |x|\}$
$\{2 \leq |x| \leq 4\}$
$= \{$Given that $3 \leq |x| \leq 4\}$
$\{2 \leq 3 \leq |x| \leq 4\}$
$= \{$arithmetic$\}$
$\{3 \leq |x| \leq 4\}$

So we have $3 \leq |x| \leq 4$ as required.

This proves that the program correctly arrives at the postcondition from the given precondition.

Question 5

The two required steps for proving some property P over natural numbers n using simple induction are the basis step and the inductive step.

The basis step invoves showing that P is *true* for $n = 0$, i.e. $P(0)$. Then, the inductive step is used to show that $P(n+1)$ follows from the assumption that $P(n)$ is true (where $P(n)$ is known as the inductive hypothesis).

Thus, from $P(0)$ it can be shown that $P(1)$, then from $P(1)$, you can prove $P(2)$, ... $P(n) \to P(n+1)$.

Question 6

We have the following assertions.

$P = \{n \in Z \wedge 0 \leq n\}$
$bf = \{n - k\}$
$Q = \{n \in Z \wedge k \in Z \wedge 1 \leq k \leq n+1 \wedge_{i=1}^{k-1} A(i) \neq x \wedge ((k \leq n \wedge A(k) = x) \vee k = n+1)\}$
$\neg done = \{k \leq n \wedge A(k) \neq x\}$
$inv = \{n \in Z \wedge k \in Z \wedge 1 \leq k \leq n+1 \wedge_{i=1}^{k-1} A(i) \neq x\}$

1. We are given the loop invariant inv.
2. Show that $\{P\}k := 1\{inv\}$

   Using the assignment axiom of the initialization step on the invariant, $inv[k := 1]$, gives:

   $\{n \in Z \wedge 1 \in Z \wedge 1 \leq 1 \leq n+1 \wedge_{i=1}^{1-1} A(i) \neq x\}$
   $= \{\text{arithmetic, 1 is an integer}\}$
   $\{n \in Z \wedge true \wedge 0 \leq n \wedge_{i=1}^{0} A(i) \neq x\}$
   $= \{\text{iteration over empty range with } true \text{ as the unit of conjunction}\}$
   $\{n \in Z \wedge true \wedge 0 \leq n \wedge true\}$

   So we have $\{n \in Z \wedge 0 \leq n\} \equiv P$ as required.
3. Show that $\{inv \wedge \neg done\}k := k+1\{inv\}$

   Using the assignment axiom of the loop body on the conjunction of the invariant and the termination condition, $inv[k := k+1] \wedge \neg done[k := k+1]$, gives:

   $\{n \in Z \wedge k+1 \in Z \wedge 1 \leq k+1 \leq n+1 \wedge_{i=1}^{k+1-1} A(i) \neq x \wedge k+1 \leq n \wedge A(k+1) \neq x\}$
   $= \{\text{arithmetic, 1 is an integer,if } k+1 \text{ is an integer, then } k \text{ must also be an integer}\}$
   $\{n \in Z \wedge k \in Z \wedge 1 \leq k+1 \leq n+1 \wedge_{i=1}^{k} A(i) \neq x \wedge k+1 \leq n \wedge A(k+1) \neq x\}$
   $= \{\text{arithmetic, } k+1 \leq n \leq n+1 \text{ implies } k+1 \leq n\}$
   $\{n \in Z \wedge k \in Z \wedge 1 \leq k+1 \leq n \wedge_{i=1}^{k} A(i) \neq x \wedge A(k+1) \neq x\}$
   $= \{\text{One-point rule with } e = k+1\}$
   $\{n \in Z \wedge k \in Z \wedge 1 \leq k+1 \leq n \wedge_{i=1}^{k} A(i) \neq x \wedge_{i=k+1}^{k+1} A(i) \neq x\}$
   $= \{\text{splitting on range } 1 \leq i \leq k \text{ and } k+1 \leq i \leq k+1\}$
   $\{n \in Z \wedge k \in Z \wedge 1 \leq k+1 \leq n \wedge_{i=1}^{k+1} A(i) \neq x\}$

   Thus we have $inv \equiv \{n \in Z \wedge k \in Z \wedge 1 \leq k+1 \leq n \wedge_{i=1}^{k+1} A(i) \neq x\}$.
4. Show that $[inv \wedge done \Rightarrow Q]$

   $done \equiv \neg\neg done \equiv \{k > n \vee A(k) = x\}$
   There are two conditions which can terminate the loop, we investigate both separately.

With $done = \{k > n\}$ we have:

$\{n \in Z \wedge k \in Z \wedge 1 \leq k \leq n+1 \wedge_{i=1}^{k-1} A(i) \neq x \wedge k > n\}$
$= \{$Together, $k > n$ and $k \leq n+1$ imply $k = n+1\}$
$\{n \in Z \wedge k \in Z \wedge 1 \leq k \leq n+1 \wedge_{i=1}^{k-1} A(i) \neq x \wedge k = n+1\}$
$\Rightarrow$
Q

For the second case, with $done = \{A(k) = x\}$ we have:

$\{n \in Z \wedge k \in Z \wedge 1 \leq k \leq n+1 \wedge_{i=1}^{k-1} A(i) \neq x \wedge A(k) = x\}$
$= \{$We know that $k \neq n+1$, so with $k \leq n+1$ we know that $k \leq n\}$
$\{n \in Z \wedge k \in Z \wedge 1 \leq k \leq n+1 \wedge_{i=1}^{k-1} A(i) \neq x \wedge k \leq n \wedge A(k) = x\}$
$\Rightarrow$
Q

5. We have bf defined as $n - k$. The loop body executes while $k \leq n$, with k incremented each iteration by the statement $k := k+1$. As k approaches n, bf approaches 0. This means that the loop will complete in at most n iterations.

Question 7

Program statement:

$\{0 \leq n\}$
$k, c := 0, 0;$
do $k < n \wedge A(k) = 0 \longrightarrow k, c := k+1, c+1$
$\Box\ k < n \wedge A(k) \neq 0 \longrightarrow k := k+1$
od
$\{c = \langle \Sigma i : 0 \leq i < n \wedge A(i) = 0 : 1 \rangle\}$

Assertions:

$P = \{0 \leq n\}$
$bf = \{n - k\}$
$Q = \{c = \langle \Sigma i : 0 \leq i < n \wedge A(i) = 0 : 1 \rangle\}$
$\neg done = \{k < n\}$

Verification of correctness:

1. The loop invariant is identified as $inv = \{0 \leq k \leq n \wedge c = \langle \Sigma i : 0 \leq i < k \wedge A(i) = 0 : 1 \rangle\}$
2. Show that $\{P\}k, c := 0, 0\{inv\}$

   Using the assignment axiom of the initialization step on the invariant, $inv[k, c := 0, 0]$, gives:

   $\{0 \leq 0 \leq n \wedge 0 = \langle \Sigma i : 0 \leq i < 0 \wedge A(i) = 0 : 1 \rangle\}$
   $= \{$iteration over empty range with 0 as the unit of summation$\}$
   $\{0 \leq n \wedge 0 = 0\}$
   $= \{$reflexivity of equality, true is the unit of conjunction$\}$
   $0 \leq n$

   So we have $P \Rightarrow inv$ as required.
3. For each statement S in the loop body, show that $\{inv \wedge \neg done\}S\{inv\}$
   For the case where $A(k) = 0$, using the assignment axiom of the loop body on the conjunction of the invariant and the termination condition, $inv[k, c := k+1, c+1] \wedge \neg done[k, c := k+1, c+1]$, gives:

   $\{0 \leq k+1 \leq n \wedge c+1 = \langle \Sigma i : 0 \leq i < k+1 \wedge A(i) = 0 : 1 \rangle \wedge k+1 < n\}$
   $= \{$arithmetic$\}$

$\{0 \leq k \leq n \wedge c + 1 = \langle \Sigma i : 0 \leq i < k + 1 \wedge A(i) = 0 : 1 \rangle\}$
$= \{$splitting over range and one-point rule$\}$
$\{0 \leq k \leq n \wedge c + 1 = \langle \Sigma i : 0 \leq i < k \wedge A(i) = 0 : 1 \rangle + \langle \Sigma i : k \leq i < k + 1 \wedge A(i) = 0 : 1 \rangle\}$
$= \{A(k) = 0$ from loop branch, so the split out summation is 1$\}$
$\{0 \leq k \leq n \wedge c + 1 = \langle \Sigma i : 0 \leq i < k \wedge A(i) = 0 : 1 \rangle + 1\}$
$= \{$arithmetic$\}$
$\{0 \leq k \leq n \wedge c = \langle \Sigma i : 0 \leq i < k \wedge A(i) = 0 : 1 \rangle\}$
$\equiv inv$

For the case where $A(k) \neq 0$, using the assignment axiom of the loop body on the conjunction of the invariant and the termination condition, $inv[k := k + 1] \wedge \neg done[k = k + 1]$, gives:

$\{0 \leq k + 1 \leq n \wedge c = \langle \Sigma i : 0 \leq i < k + 1 \wedge A(i) = 0 : 1 \rangle \wedge k + 1 < n\}$
$= \{$arithmetic$\}$
$\{0 \leq k \leq n \wedge c = \langle \Sigma i : 0 \leq i < k + 1 \wedge A(i) = 0 : 1 \rangle\}$
$= \{$splitting over range and one-point rule$\}$
$\{0 \leq k \leq n \wedge c = \langle \Sigma i : 0 \leq i < k \wedge A(i) = 0 : 1 \rangle + \langle \Sigma i : k \leq i < k + 1 \wedge A(i) = 0 : 1 \rangle\}$
$= \{A(k) \neq 0$ from loop branch, 0 is the unit of summation over empty range$\}$
$\{0 \leq k \leq n \wedge c = \langle \Sigma i : 0 \leq i < k \wedge A(i) = 0 : 1 \rangle + 0\}$
$\equiv inv$

4. Show that $[inv \wedge done \Rightarrow Q]$

   $done \equiv \neg\neg done \equiv \{k \geq n\}$

   $\{0 \leq k \leq n \wedge c = \langle \Sigma i : 0 \leq i < k \wedge A(i) = 0 : 1 \rangle \wedge k \geq n\}$
   $= \{$Together, $k \geq n$ and $k \leq n$ imply $k = n\}$
   $\{c = \langle \Sigma i : 0 \leq i < n \wedge A(i) = 0 : 1 \rangle\}$
   $\Rightarrow$
   Q
5. We have bf defined as $n - k$. The loop body executes while $k < n$, with k incremented each iteration by the statement $k := k + 1$. As k approaches n, bf approaches 0. This means that the loop will complete in at most n iterations.

Question 8

We will prove by simple induction that $P \equiv \langle \Sigma k : 0 \leq k < n : 2^k \rangle = 2^n - 1$ for all $n \in N$

We start by showing $P(0)$:

$P(0)$
$= \{$definition$\}$
$\langle \Sigma k : 0 \leq k < 0 : 2^k \rangle = 2^0 - 1$
$= \{$0 is the unit of summation over the empty range$\}$
$0 = 2^0 - 1$
$= \{$arithmetic$\}$
$0 = 0$
$= \{$reflexivity of equality$\}$
$true$

We now show via the induction step that $P(n + 1)$:

$P(n + 1)$
$= \{$definition$\}$
$\langle \Sigma k : 0 \leq k < n + 1 : 2^k \rangle = 2^{n+1} - 1$
$= \{$split out the last element of the summation$\}$
$\langle \Sigma k : 0 \leq k < n : 2^k \rangle + 2^n = 2^{n+1} - 1$

$= \{\text{assume } P(n) \text{ (induction hypothesis)}\}$
$(2^n - 1) + 2^n = 2^{n+1} - 1$
$= \{\text{arithmetic}\}$
$(2^n - 1) + 2^n = (2^n * 2) - 1$
$= \{\text{arithmetic}\}$
$2(2^n) - 1 = (2^n * 2) - 1$
$= \{\text{arithmetic}\}$
$true$

Therefore, by the principle of simple mathematical induction, we conclude that $P(n)$ for all natural numbers n.