

Week 10 Tutorial Solutions
Weakest Precondition Calculus

Question 1

- a) $wp(y:=tmp, (x = 3 \wedge y = 5))$
 $\equiv (x = 3 \wedge tmp = 5)$
- b) $wp(x:=y; y:=tmp, (x = 3 \wedge y = 5))$
 $\equiv wp(x:=y, wp(y:=tmp, (x = 3 \wedge y = 5)))$
 $\equiv wp(x:=y, (x = 3 \wedge tmp = 5))$
 $\equiv (y = 3 \wedge tmp = 5)$
- c) $wp(tmp:=x; x:=y; y:=tmp, (x = 3 \wedge y = 5))$
 $\equiv wp(tmp:=x, wp(x:=y; y:=tmp, (x = 3 \wedge y = 5)))$
 $\equiv wp(tmp:=x, (y = 3 \wedge tmp = 5))$
 $\equiv (y = 3 \wedge x = 5)$
- d) $wp(\text{if } x>y \text{ then } (tmp:=x; x:=y; y:=tmp), (x = 3 \wedge y = 5))$
 $\equiv ((x > y) \wedge wp(tmp:=x; x:=y; y:=tmp, (x = 3 \wedge y = 5)))$
 $\vee (\neg(x > y) \wedge (x = 3 \wedge y = 5))$
 $\equiv ((x > y) \wedge (y = 3 \wedge x = 5)) \vee (\neg(x > y) \wedge (x = 3 \wedge y = 5))$
 $\equiv ((y = 3 \wedge x = 5) \vee (x = 3 \wedge y = 5))$

Question 2

B is the code `if k>0 then begin k:=k-1; n:=n*2 end`. Let B' be the conditionally executed code, that is: `k:=k-1; n:=n*2`.

- a) $wp(B', (k = 0 \wedge n = 0)) \equiv wp(k:=k-1, k = 0 \wedge 2n = 0)$
 $\equiv (k = 1 \wedge n = 0)$
 $wp(B, (k = 0 \wedge n = 0)) \equiv (k > 0 \wedge wp(B', (k = 0 \wedge n = 0))) \vee (\neg(k > 0) \wedge (k = 0 \wedge n = 0))$
 $\equiv (k = 1 \vee k = 0) \wedge n = 0$
- b) $wp(B', (n = 2^{j-k})) \equiv wp(k:=k-1, (2n = 2^{j-k}))$
 $\equiv n = 2^{j-k}$
 $wp(B, (n = 2^{j-k})) \equiv (k > 0 \wedge wp(B', (n = 2^{j-k}))) \vee (\neg(k > 0) \wedge (n = 2^{j-k}))$
 $\equiv (k > 0 \wedge n = 2^{j-k}) \vee (k \leq 0 \wedge n = 2^{j-k})$
 $\equiv (n = 2^{j-k})$

Question 3

Note that the normal mathematical definition of factorial is as a partial function; that is, $k!$ is undefined for negative values of k . This means, for example, that $(-1)! = n$ is not true for any integer n . In the program at hand $Post$ can only be true if $n \geq 0$.

$$\begin{aligned}
 1) \quad P_0 &\equiv (i \geq n \wedge Q) \equiv (fac = i! \wedge 0 \leq i = n) \\
 P_1 &\equiv (i < n \wedge wp(B, P_0)) \equiv (fac * (i + 1) = (i + 1)! \wedge 0 \leq (i + 1) = n) \\
 &\equiv (fac = i! \wedge 0 \leq i = (n - 1))
 \end{aligned}$$

[Note: At first sight the last step consists of dividing both sides of an equality by $(i + 1)$. That is not a legal move since $i + 1$ may be zero. Instead, we do case analysis on whether $i + 1 = 0$ or not; if it is, then P_1 is false (because $fac * 0 \neq 0!$) and so we *are* able to make the simplification. Moreover, as a side benefit we use this information to simplify the other part of the condition to $0 \leq (i + 1) = n$.]

$$\begin{aligned}
 P_2 &\equiv (i < n \wedge wp(B, P_1)) \equiv (fac * (i + 1) = (i + 1)! \wedge 0 \leq (i + 1) = (n - 1)) \\
 &\equiv (fac = i! \wedge 0 \leq i = (n - 2))
 \end{aligned}$$

$$\begin{aligned}
 2) \quad P_k &\equiv (i < n \wedge wp(B, P_{k-1})) \\
 &\equiv (fac = i! \wedge 0 \leq i = (n - k))
 \end{aligned}$$

[Note 1: Strictly, an inductive argument is required to support this claim about P_k . Since it is a straightforward one, it is left to the reader to construct it.]

[Note 2: P_k is false for $k > n$.]

[Note 3: The predicate $Post$ tells all about the ‘answer’ of this program whereas Q documents the final state more completely. The (similar) predicate $(fac = i! \wedge 0 \leq i \leq n)$ is the loop invariant and is precise documentation for the loop. Although it’s not well illustrated by this program, $Post$ isn’t as useful for reasoning about the workings of the program.]

$$\begin{aligned}
 3) \quad wp(W, Q) &\equiv \exists k.(k \geq 0 \wedge P_k) \\
 &\equiv \exists k.(k \geq 0 \wedge fac = i! \wedge 0 \leq i = (n - k)) \\
 &\equiv (fac = i! \wedge 0 \leq i \wedge i \leq n)
 \end{aligned}$$

$$\begin{aligned}
 4) \quad wp(Facto, Q) &\equiv wp(i := 0; fac := 1, wp(W, Q)) \\
 &\equiv 1 = 0! \wedge 0 \leq 0 \wedge 0 \leq n \\
 &\equiv n \geq 0
 \end{aligned}$$

5) Suppose that $n \geq 0$ holds in some state S_i in which $Facto$ computes. The above result says that it will terminate in a state (call it S_f) in which Q holds.

Since $Q \implies Post$ is obviously true, $Post$ holds in S_f , and so we have proven the program specification, $\{n \geq 0\} Facto \{fac = n!\}$.