1. pre-condition: \( \{ N>0 \} \)
   post-condition: \( \{(NPRIME<=>[N>1) AND (for every 1<i<N, there exists no q such that iXq=N)] AND UNCH(N)\} \)

2. \( f = (X>10 \rightarrow X,Y := 0,Y+X \mid X>5 \rightarrow X,Y := Y,Y-X \mid true \rightarrow I) \)


4. a

5. G, F, H, E, G, B, A, D, E, C

6. e

7. h

8. b

9. In large projects, team members spend a smaller proportion of their time in development activities (supported by development tools) and more time communicating (with one another) and understanding other parts of the system. Development tools, Sommerville argues, make no difference to this.

10. **rapidity vs. visibility**: making a process visible requires the people involved to produce info about the process itself; this may slow down software production because of the time it takes to produce these documents.

    **maintainability, standardization vs. acceptability**: To make a process more maintainable, you may have to adopt procedures and tools that reflect broader organizational practice and that are used in different parts of the company; such standardization may conflict with non-standard procedures and tools introduced to support locally preferred ways of working. The result can be a reduction in process acceptability.

11. true, true, true, true, false

12. The WP-based method requires showing \( P \Rightarrow \text{wp}(S,Q) \)

    \[
    \text{wp}(S,Q) = (x\neq 0 \land \text{wp}(y := -x, y>0)) \lor (x=0 \land y>0)
    = (x\neq 0 \land -x>0) \lor (x=0 \land y>0)
    = (x\neq 0 \land x<0) \lor (x=0 \land y>0)
    = (x<0 \lor (x=0 \land y>0))
    \]

    Does \( x=-5 \Rightarrow (x<0 \lor (x=0 \land y>0)) \) ?

    \[
    x=-5 \Rightarrow [(x<0 \lor (x=0 \land y>0)) = (true \lor (x\geq 0 \land y>0))
    = true]
    \]

    Therefore, \( P \Rightarrow \text{wp}(S,Q) \). Thus, the assertion holds.
13. \( P \Rightarrow I, \{I \land b\} S \{I, (I \land \neg b) \Rightarrow Q\} \) \\
\{P\} while \( b \) do \( S \{Q\}\)

b. vii

14. Proof: It is sufficient to show: 
\[ f = (x,y := y+2,y) = [(x,y := y+2,y); (x,y := x,x-2)] = (x,y := x,x-2) o (x,y := y+2,y) = (x,y := y+2,(y+2)-2) = (x,y := y+2,y) = f \]

Therefore, \( f = [P] \).

15. a. \( y+z = y_0+z_0 \)

b. i. \( X_0 = (17,-4), X_1 = (16,-3), X_2 = (15,-2), X_3 = (14,-1), X_4 = (13,0) \)

ii. “Agrees with \( q(x) \)” in this case just means: \( y+z = y_0+z_0 \).

\( (17,-4): 17-4 = 17-4 \quad \checkmark \quad \text{(tautology)} \), \( (16,-3): 16-3 = 17-4 \quad \checkmark \), \( (15,-2): 15-2 = 17-4 \quad \checkmark \), \( (14,-1): 14-1 = 17-4 \quad \checkmark \), \( (13,0): 13-0 = 17-4 \quad \checkmark \)

iii. \( X_k \), of course, is NOT one of the states in the sequence of states from \( X_0 \) to \( X_4 \) identified in part (i), but it DOES agree with \( q(X) \): \( (-5,18): -5+18 = 17-4 \quad \checkmark \)

The **implication** is that while \( P1 \) does not produce the intermediate state \( X_k \) while mapping \( (17,-4) \) to \( (13,0) \), some loop that computes \( f \) could do so.

iv. Like \( X_k \), \( X_t \) is not one of the states in the sequence of states from \( X_0 \) to \( X_4 \) identified in part (i), and neither does it agree with \( q(X) \): \( (0,0): 0+0 \neq 17-4 \)

The **implication** is that no while loop that computes \( f \) could produce \( \{0,0\} \) as an intermediate state from the input \( \{17,-4\} \).

c. i. \( X_0 = (3,4), X_1 = (7,0) \)

ii. \( (3,4): 3+4 = 3+4 \quad \checkmark \quad \text{(tautology)} \), \( (16,-3): 7+0 = 3+4 \quad \checkmark \)

d. i. \( (-6,3): -6+3 = -6+3 \quad \checkmark \quad \text{(tautology)} \), \( (-3,0): -3+0 = -6+3 \quad \checkmark \)

ii. Yes, by observation, the nature of \( f \) (in this particular case) is such that \( f(X_0) = f(X_n) \) where \( X_0 \) is the initial state and \( X_n \) is the final state. Thus, the initial and final states of any program computing \( f \) (again, in this particular case) must agree with \( q(X) \).

16. a. \( (y+1 = y_0+1) \equiv (y = y_0) \)

b. \( y_0: y_0 = y_0 \quad \checkmark \quad \text{(tautology)} \), \( y_1: y_0+1 \neq y_0 \quad (q(X) \text{ does not agree with the final state}) \)

c. **There is no such program.** A fundamental property of all functions, \( f \), computed by a while loop is that \( f(X) = f(X_0) \) for all execution states, \( X \), that hold when the loop predicate is evaluated. This implies that the final state, \( X_n = f(X_0) = f(X_1) = \ldots = f(X_n) \). If \( f' \) were computed by some while loop, it would therefore follow that \( y_0+1 = f'(y_0) = f'(y_0+1) \). But in fact, \( f'(y_0+1) = y_0+2 \), by definition of \( f' \). Therefore \( X_0 = f'(X_0) \neq f'(X_n) \), and we conclude that there is no program of the form “while \( b \) do s” that computes function \( f' \).
Histogram of Raw Scores

123456789111111112222222222333333334444444445555555555566666666666666777777777777778888888889
0123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890

^|
mean
(63 pts)