A. Why

• A formal proof lets us write out in detail the reasons for believing that something is valid.
• Proof outlines condense the same information as a proof.

B. Objectives

At the end of this class you should

• Know how the invariant, initialization, test, body, and postcondition of a loop are interrelated.
• Know how to write and check a formal proof of partial correctness.

C. Alternative Invariants Yield Different Programs and Proofs

• The invariant, test, initialization code, and body of a loop are all interconnected: Changing one can change them all. For example, we used $s = \text{sum}(0, k)$ in our earlier summation loop invariant, so we had the loop terminate with $k = n$.

Example 1: (Original summation loop)

• Using $s = \text{sum}(0, k)$

  $$\{n \geq 0\}$$
  $$k := 0; \ s := 0;$$
  $$\{\text{inv } p_1 = 0 \leq k \leq n \land s = \text{sum}(0, k)\}$$
  $$\text{while } k < n \text{ do}$$
  $$\quad s := s+k+1; \ k := k+1$$
  $$\text{od}$$
  $$\{s = \text{sum}(0, n)\}$$

• If instead we use $s = \text{sum}(0, k+1)$ or $s = \text{sum}(0, k–1)$ in our invariant, we must terminate with $k+1 = n$ or $k–1 = n$ respectively, and we change what we should increment $s$ by.

Example 2: Using $s = \text{sum}(0, k+1)$

• Using $s = \text{sum}(0, k+1)$ gives us

  $$\{n > 0\}$$
  $$k := 0; \ s := 1;$$
  $$\{\text{inv } p_2 = 0 \leq k < n \land s = \text{sum}(0, k+1)\}$$
  $$\text{while } k < n-1 \text{ do}$$
Example 3: Using \( s = \text{sum}(0, k-1) \)

- Using \( s = \text{sum}(0, k-1) \) gives us

\[
\begin{align*}
{n \geq 0} \\
k := 1; s := 0; \\
\{ \text{inv } p_2 = 1 \leq k \leq n+1 \land s = \text{sum}(0, k-1) \} \\
\text{while } k \leq n \text{ do} \\
\quad s := s+k; k := k+1 \\
\text{od} \\
\{s = \text{sum}(0, n)}
\end{align*}
\]

D. Formal Proofs of Partial Correctness

- As you’ve seen, the format of a formal proof is very rigid syntactically. The relationship between formal proofs and informal proofs is like the description of an algorithm in a program (very rigid syntax) versus in pseudocode (much more informal syntax).

- Just as a reminder, we’re using Hilbert-style proofs: Each line’s assertion is an assumption, an axiom, or follows by some rule that appeals to earlier lines in the proof. In high-school geometry, we might have used

1. \( \text{Length of } AB = \text{length of } XY \) \hspace{1cm} \text{Assumption}
2. \( \text{Angle } ABC = \text{Angle } XYZ \) \hspace{1cm} \text{Assumption}
3. \( \text{Length of } BC = \text{length of } YZ \) \hspace{1cm} \text{Assumption}
4. \( \text{Triangles } ABC, XYZ \text{ are congruent} \) \hspace{1cm} \text{Side-Angle-Side, lines 1, 2, 3}

E. Sample Formal Proofs

- We can write out the reasoning for the sample summation loop we looked at. We've seen formal proofs of the loop body's correctness; all we really have to do is attach the proof of loop initialization correctness:

Example 1 (repeated):

\[
\begin{align*}
{n \geq 0} \\
k := 0; s := 0; \\
\{ \text{inv } p_1 = 0 \leq k \leq n \land s = \text{sum}(0, k) \} \\
\text{while } k < n \text{ do} \\
\quad s := s+k+1; k := k+1 \\
\text{od} \\
\{s = \text{sum}(0, n)}
\end{align*}
\]

- Below, let \( S_1 = s := s+k+1; k := k+1 \) (the loop body) and let \( W = \text{while } k < n \text{ do } S_1 \text{ od} \) (the loop).
To review, the order of the lines in the proof is somewhat arbitrary — you can only refer to lines above you in the proof, but they can be anywhere above you.

For example, lines 1 and 2 don’t have to be in that order, they just have to be before we use them in the sequence rule at line 3 (which in turn has to be somewhere before line 5, and so on).

### F. Full Proof Outlines

- Formal proofs are long and contain repetitive information (we keep copying the same conditions over and over).
- In a proof outline, we add conditions to the inside of the program, not just the ends.
- A full proof outline is a way to write out all the information that you would need to generate a formal proof of partial correctness.
• Each triple in the proof must appear in the outline. Said another way, every statement must be part of a triple (including sequences, conditionals, and loops), and every triple must be provable using the proof rules.
• Precondition strengthening appears as a triple with an extra precondition; postcondition weakening appears as a triple with an extra postcondition.
• If two conditions sit next to each other, as in \{p_1 \land p_2\}, it stands for a predicate calculus obligation of \(p_1 \rightarrow p_2\).
• A proof outline does not stand for a unique proof. Aside from permuting line orderings, the timing of precondition strengthening and postcondition weakening may not be unique.

**Example 1**

• Here we form the sequence \(k := 0; x := 1\) and then weaken its postcondition.

\[
\begin{align*}
1. & \quad \{T\} k := 0 \{k = 0\} & \text{Assignment} \\
2. & \quad \{k = 0\} x := 1 \{k = 0 \land x = 1\} & \text{Assignment} \\
3. & \quad \{T\} k := 0; x := 1 \{k = 0 \land x = 1\} & \text{Sequence 1, 2} \\
4. & \quad k = 0 \land x = 1 \rightarrow k \geq 0 \land x = 2^k & \text{Predicate logic} \\
5. & \quad \{T\} k := 0; x := 1 \{k \geq 0 \land x = 2^k\} & \text{Postcond. weak. 3, 4}
\end{align*}
\]

• The full proof outline for this is \(\{T\} k := 0; \{k = 0\} x := 1 \{k = 0 \land x = 1\} \{k \geq 0 \land x = 2^k\}\)

**Example 2**

• This is like Example 1 but uses precondition strengthening instead of postcondition weakening.

\[
\begin{align*}
1. & \quad \{k \geq 0 \land 1 = 2^k\} \ x := 1 \{k \geq 0 \land x = 2^k\} & \text{Assignment} \\
2. & \quad \{0 \geq 0 \land 1 = 2^0\} \ k := 0 \{k \geq 0 \land 1 = 2^k\} & \text{Assignment} \\
3. & \quad \{0 \geq 0 \land 1 = 2^0\} \ k := 0; \ x := 1 \{k \geq 0 \land x = 2^k\} & \text{Sequence 2, 1} \\
4. & \quad k \geq 0 \land 1 = 2^0 & \text{Predicate logic} \\
5. & \quad \{T\} k := 0; \ x := 1 \{k \geq 0 \land x = 2^k\} & \text{Precond Str. 4, 3}
\end{align*}
\]

• The full proof outline is \(\{T\} \{0 \geq 0 \land 1 = 2^0\} \ k := 0; \{k \geq 0 \land 1 = 2^k\} \ x := 1 \{k \geq 0 \land x = 2^k\}\)

**Example 3**

• Here’s a full proof outline for the summation loop; note how the structure of the outline follows the partial correctness proof, which is shown below.

\[
\begin{align*}
\{n \geq 0\} & \ k := 0; \{n \geq 0 \land k = 0\} \ s := 0; \{n \geq 0 \land k = 0 \land s = 0\} \\
\{\text{inv} \ p_1 = 0 \leq k \leq n \land s = \text{sum}(0, k)\} \\
\text{while} \ k < n \ \text{do} & \\
\{p_1 \land k < n\} & \{p_1[k+1/k][s+k+1/s]\} \\
\ s := s+k+1; \{p_1[k+1/k]\} & \\
\ k := k+1 \{p_1\} & \text{od} \\
\{p_1 \land k \geq n\} & \\
\{s = \text{sum}(0, n)\} &
\end{align*}
\]

• The triples and predicate obligations from this outline are below. They have been listed in the same order as above, in the formal proof of the summation program.
1. \(\{n \geq 0\} \ k := 0 \ {n \geq 0 \land k = 0}\)
2. \(\{n \geq 0 \land k = 0\} \ s := 0 \ {n \geq 0 \land k = 0 \land s = 0}\)
3. \(\{n \geq 0\} \ k := 0; s := 0 \ {n \geq 0 \land k = 0 \land s = 0}\)
4. \(n \geq 0 \land k = 0 \land s = 0 \rightarrow p_1\)
5. \(\{n \geq 0\} \ k := 0; s := 0 \ {p_1}\)
6. \(\{p_1[k+1/k]\} \ k := k+1 \ {p_1}\)
7. \(\{p_1[k+1/k][s+k+1/s]\} \ s := s+k+1 \ {p_1[k+1/k]}\)
8. \(\{p_1[k+1/k][s+k+1/s]\} \ s := s+k+1; k := k+1 \ {p_1}\)
9. \(p_1 \land k < n \rightarrow p_1[k+1/k][s+k+1/s]\)
10. \(\{p_1 \land k < n\} \ s := s+k+1; k := k+1 \ {p_1}\)
11. \(\{inv \ p_1\} \ W \{p_1 \land k \geq n\} \mbox{ where } W = \mbox{ while } k < n \ do \ s := s+k+1; k := k+1 \ od\)
12. \(\{n \geq 0\} \ k := 0; s := 0; \{inv \ p_1\} \ W \{p_1 \land k \geq n\}\)
13. \(p_1 \land k \geq n \rightarrow s = \mbox{sum}(0, n)\)
14. \(\{n \geq 0\} \ k := 0; s := 0; \{inv \ p_1\} \ W \{s = \mbox{sum}(0, n)\}\)
Proofs and Proof Outlines for Partial Correctness

Part 1: Full Proofs and Proof Outlines of Partial Correctness

CS 536: Science of Programming, Spring 2021

3/22 Added full proof outline problems

A. Why

• A formal proof lets us write out in detail the reasons for believing that something is valid.

B. Objectives

At the end of this activity assignment you should be able to

• Write and check formal proofs of partial correctness.

C. Problems

Formal Proofs

1. The formal proof below is incomplete; fill in the missing rule names (and line references, where needed).

   1. \( T \rightarrow 0 \geq 0 \land 1 = 2^0 \)
   2. \( \{0 \geq 0 \land 1 = 2^0\} k := 0 \{k \geq 0 \land 1 = 2^k\} \)
   3. \( \{T\} k := 0 \{k \geq 0 \land 1 = 2^k\} \)
   4. \( \{k \geq 0 \land 1 = 2^k\} x := 1 \{k \geq 0 \land x = 2^k\} \)
   5. \( \{T\} k := 0; x := 1 \{k \geq 0 \land x = 2^k\} \)

   Here's an alternate version of the proof that uses forward assignments:

   1. \( \{T\} k := 0 \{k = 0\} \)
   2. \( \{k = 0\} x := 1 \{k = 0 \land x = 1\} \)
   3. \( \{T\} k := 0; x := 1 \{k = 0 \land x = 1\} \)
   4. \( k = 0 \land x = 1 \rightarrow k \geq 0 \land x = 2^k \)
   5. \( \{T\} k := 0; x := 1 \{k \geq 0 \land x = 2^k\} \)

2. Repeat Problem 1 on the incomplete proof below.

   1. \( \{-x = \text{abs}(x)\} y := -x \{y = \text{abs}(x)\} \)
   2. \( y = x \land x < 0 \rightarrow -x = \text{abs}(x) \)
   3. \( \{y = x \land x < 0\} y := -x \{y = \text{abs}(x)\} \)
   4. \( \{y = \text{abs}(x)\} \text{skip} \{y = \text{abs}(x)\} \)
   5. \( y = x \land x \geq 0 \rightarrow y = \text{abs}(x) \)
   6. \( \{y = x \land x \geq 0\} \text{skip} \{y = \text{abs}(x)\} \)
   7. \( \{y = x\} \text{if } x < 0 \text{ then } y := -x \text{ fi } \{y = \text{abs}(x)\} \)
3. Repeat Problem 1 on the incomplete proof below.

Below, let \( W = \text{while } j > 0 \text{ do } j := j - 1; \ s := s + j \text{ od} \)

1. \( \{n \geq 0\} j := n \ \{n \geq 0 \land j = n\} \)
2. \( \{n \geq 0 \land j = n\} s := n \ \{n \geq 0 \land j = n \land s = n\} \)
3. \( \{n \geq 0\} j := n; \ s := n \ \{n \geq 0 \land j = n \land s = n\} \)
4. \( n \geq 0 \land j = n \land s = n \rightarrow p \)
5. \( \{n \geq 0\} j := n; \ s := n \ \{p\} \)
6. \( \{p[s+j/s]\} s := s + j \ \{p\} \)
7. \( \{p[s+j/s][j-1/j]\} j := j - 1 \ \{p[s+j/s]\} \)
8. \( p \land j > 0 \rightarrow p[s+j/s][j-1/j] \)
9. \( \{p \land j > 0\} j := j - 1 \ \{p[s+j/s]\} \)
10. \( \{p \land j > 0\} j := j - 1; \ s := s + j \ \{p\} \)
11. \( \{\text{inv } p\} W \ \{p \land j \leq 0\} \)
12. \( p \land j \leq 0 \rightarrow s = \text{sum}(0, n) \)
13. \( \{\text{inv } p\} W \ \{s = \text{sum}(0, n)\} \)
14. \( \{n \geq 0\} j := n; \ s := n; \ \{\text{inv } p\} W \ \{s = \text{sum}(0, n)\} \)

4. Write a formal proof of partial correctness for \( \{n > 1\} x := n; \ x := x \times x \ \{x \geq 4\} \) that uses \( wp \) and precondition strengthening.

5. Repeat Problem 4 but use \( sp \) and postcondition weakening.

**Full Proof Outlines**

For Problems 1–3, you are given a full proof outline; write a corresponding proof of partial correctness from it. There are multiple right answers.

1. \( \{T\} \{0 \geq 0 \land 1 = 2^0\} k := 0; \ \{k \geq 0 \land 1 = 2^k\} x := 1 \ \{k \geq 0 \land x = 2^k\} \)

2a. \( \{y = x\} \text{if } x < 0 \text{then} \)

   \( \{y = x \land x < 0\} \{-x = \text{abs}(x)\} y := -x \ \{y = \text{abs}(x)\} \)

   \text{else}

   \( \{y = x \land x \geq 0\} \ \{y = \text{abs}(x)\} \ \text{skip} \ \{y = \text{abs}(x)\} \)

   \text{fi} \ \{y = \text{abs}(x)\} \)
2b. \{ y = x \} \textbf{if} x < 0 \textbf{then} \{ y = x \land x < 0 \} y := -x \{ y_0 = x \land x < 0 \land y = -x \} \textbf{else} \{ y = x \land x \geq 0 \} \textbf{skip} \{ y = x \land x \geq 0 \} \textbf{fi} \{ (y = x \land x < 0 \land y = -x) \lor (y = x \land x \geq 0) \} \{ y = \text{abs}(x) \} \\

2c. \{ y = x \} \{ (x < 0 \rightarrow -x = \text{abs}(x)) \land (x \geq 0 \rightarrow y = \text{abs}(x)) \} \textbf{if} x < 0 \textbf{then} \{-x = \text{abs}(x)\} y := -x \{ y = \text{abs}(x) \} \textbf{else} \{ y = \text{abs}(x) \} \textbf{skip} \{ y = \text{abs}(x) \} \textbf{fi} \{ y = \text{abs}(x) \} \\

3. Hint: Use \texttt{sp} for the two loop initialization assignments. \\
\{ n \geq 0 \} j := n; \{ n \geq 0 \land j = n \} s := n; \{ n \geq 0 \land j = n \land s = n \} \textbf{inv} p = 0 \leq j \leq n \land s = \text{sum}(j, n) \textbf{while} j > 0 \textbf{do} \{ p \land j > 0 \} \{ p[s+j/s][j-1/j] \} j := j-1; \{ p[s+j/s] \} s := s+j \{ p \} \textbf{od} \{ p \land j \leq 0 \} \{ s = \text{sum}(0, n) \}
Solution to Practice 16 (Formal Proofs and Full Proof Outlines)

1. Proof:
   1. \(T \rightarrow 0 \geq 0 \land 1 = 2^0\)  
      \(\text{Predicate logic}\)
   2. \(\{0 \geq 0 \land 1 = 2^0\} \ k := 0 \ \{k \geq 0 \land 1 = 2^k\}\)  
      \(\text{Assignment (backward)}\)
   3. \(\{T\} \ k := 0 \ \{k \geq 0 \land 1 = 2^k\}\)  
      \(\text{Precond strengthening 1, 2}\)
   4. \(\{k \geq 0 \land 1 = 2^k\} \ x := 1 \ \{k \geq 0 \land x = 2^k\}\)  
      \(\text{Assignment (backward)}\)
   5. \(\{T\} \ k := 0; x := 1 \ \{k \geq 0 \land x = 2^k\}\)  
      \(\text{Sequence 3, 4}\)
   [Alternate version]
   1. \(\{T\} \ k := 0 \ \{k = 0\}\)  
      \(\text{Assignment (forward)}\)
   2. \(\{k = 0\} \ x := 1 \ \{k = 0 \land x = 1\}\)  
      \(\text{Assignment (forward)}\)
   3. \(\{T\} \ k := 0; x := 1 \ \{k = 0 \land x = 1\}\)  
      \(\text{Sequence 1, 2}\)
   4. \(k = 0 \land x = 1 \rightarrow k \geq 0 \land x = 2^k\)  
      \(\text{Predicate logic}\)
   5. \(\{T\} \ k := 0; x := 1 \ \{k \geq 0 \land x = 2^k\}\)  
      \(\text{Postcond weakening 3, 4}\)

2. Proof:
   1. \(\{-x = \text{abs}(x)\} \ y := -x \ \{y = \text{abs}(x)\}\)  
      \(\text{Assignment}\)
   2. \(y = x \land x < 0 \rightarrow -x = \text{abs}(x)\)  
      \(\text{Predicate logic}\)
   3. \(\{y = x \land x < 0\} \ y := -x \ \{y = \text{abs}(x)\}\)  
      \(\text{Precond str 2, 1}\)
   4. \(y = \text{abs}(x)\)  
      \(\text{Skip}\)
   5. \(y = x \land x \geq 0 \rightarrow y = \text{abs}(x)\)  
      \(\text{Predicate logic}\)
   6. \(\{y = x \land x \geq 0\} \ \text{skip} \ \{y = \text{abs}(x)\}\)  
      \(\text{Precond str 5, 4}\)
   7. \(\{y = x\} \ \text{if } x < 0 \ \text{then } y := -x \ \text{fi} \ \{y = \text{abs}(x)\}\)  
      \(\text{Conditional 3, 6}\)

3. Below, \(W = \text{while } j > 0 \ \text{do } j := j-1; s := s+j \ \text{od}\)
   1. \(\{n \geq 0\} \ j := n \ \{n \geq 0 \land j = n\}\)  
      \(\text{Assignment}\)
   2. \(\{n \geq 0 \land j = n\} \ s := n \ \{n \geq 0 \land j = n \land s = n\}\)  
      \(\text{Assignment}\)
   3. \(\{n \geq 0\} \ j := n; s := n \ \{n \geq 0 \land j = n \land s = n\}\)  
      \(\text{Sequence 1, 2}\)
   4. \(n \geq 0 \land j = n \land s = n \rightarrow p\)  
      \(\text{Predicate logic}\)
   5. \(\{n \geq 0\} \ j := n; s := n \ \{p\}\)  
      \(\text{Postcond. weak 3, 4}\)
   6. \(\{p[s+j/s]\} \ s := s+j \ \{p\}\)  
      \(\text{Assignment}\)
   7. \(\{p[s+j/s][j-1/j]\} \ j := j-1 \ \{p[s+j/s]\}\)  
      \(\text{Assignment}\)
   8. \(p \land j > 0 \rightarrow p[s+j/s][j-1/j]\)  
      \(\text{Predicate logic}\)
   9. \(p \land j > 0 \ \text{if } j = j-1 \ \text{then } p[s+j/s]\)  
      \(\text{Precond. str 8, 7}\)
   10. \(p \land j > 0 \ \text{if } j = j-1 \ \text{then } s := s+j \ \{p\}\)  
      \(\text{Sequence 9, 6}\)
   11. \(\{\text{inv } p\} \ W \ \{p \land j \leq 0\}\)  
      \(\text{While 10}\)
   12. \(p \land j \leq 0 \rightarrow s = \text{sum}(0, n)\)  
      \(\text{Predicate logic}\)
   13. \(\{\text{inv } p\} \ W \ \{s = \text{sum}(0, n)\}\)  
      \(\text{Postcond. weak 12, 11}\)
   14. \(\{n \geq 0\} \ j := n; s := n \ \{\text{inv } p\} \ W \ \{s = \text{sum}(0, n)\}\)  
      \(\text{Sequence 5, 13}\)
4. Proof using wp:
   1. \(\{x \times x \geq 4\} \ x := x \times x \{x \geq 4\}\) \hspace{1em} (Backward) Assignment
   2. \(\{n \times n \geq 4\} \ x := n; \{x \times x \geq 4\}\) \hspace{1em} (Backward) Assignment
   3. \(\{n \times n \geq 4\} \ x := n; \ x := x \times x \{x \geq 4\}\) \hspace{1em} Sequence 2, 1
   4. \(n > 1 \rightarrow n \times n \geq 4\) \hspace{1em} Predicate logic
   5. \(\{n > 1\} \ x := n; \ x := x \times x \{x \geq 4\}\) \hspace{1em} Precond. str. 4, 3

5. Proof using sp:
   1. \(\{n > 1\} \ x := n; \{n > 1 \land x = n\}\) \hspace{1em} (Forward) Assignment
   2. \(\{n > 1 \land x = n\} \ x := x \times x \{n > 1 \land x = n \land x = x \times x\}\) \hspace{1em} (Forward) Assignment
   3. \(\{n > 1\} \ x := n; \ x := x \times x \{n > 1 \land x = n \land x = x \times x\}\) \hspace{1em} Sequence 1, 2
   4. \(n > 1 \land x_0 = n \land x = x \times x \rightarrow x \geq 4\) \hspace{1em} Predicate logic
   5. \(\{n > 1\} \ x := n; \ x := x \times x \{x \geq 4\}\) \hspace{1em} Postcond. weak 3, 4

**Full Proof Outlines (Solution)**

1. (Full outline to proof):
   1. \(T \rightarrow 0 \geq 0 \land 1 = 2^0\) \hspace{1em} Predicate logic
   2. \(\{0 \geq 0 \land 1 = 2^0\} \ k := 0; \{k \geq 0 \land 1 = 2^k\}\) \hspace{1em} Assignment
   3. \(\{T\} \ k := 0; \{k \geq 0 \land 1 = 2^k\}\) \hspace{1em} Precond str 1, 2
   4. \(\{k \geq 0 \land 1 = 2^k\} \ x := 1 \{k \geq 0 \land x = 2^k\}\) \hspace{1em} Assignment
   5. \(\{T\} \ k := 0; \ x := 1 \{k \geq 0 \land x = 2^k\}\) \hspace{1em} Sequence 3, 4

2a. (Full outline to proof):
   1. \(-x = abs(x)\} \ y := -x \{y = abs(x)\}\) \hspace{1em} Assignment
   2. \(y = x \land x < 0 \rightarrow -x = abs(x)\) \hspace{1em} Predicate logic
   3. \(\{y = x \land x < 0\} \ y := -x \{y = abs(x)\}\) \hspace{1em} Precond str 2, 1
   4. \(\{y = abs(x)\} \ skip \{y = abs(x)\}\) \hspace{1em} Skip
   5. \(y = x \land x \geq 0 \rightarrow y = abs(x)\) \hspace{1em} Predicate logic
   6. \(\{y = x \land x \geq 0\} \ skip \{y = abs(x)\}\) \hspace{1em} Precond str 5, 4
   7. \(\{y = x\} \ if \ x < 0 \ then \ y := -x \ fi \{y = abs(x)\}\) \hspace{1em} Conditional 3, 6

2b. (Full outline to proof):
   1. \(\{y = x \land x < 0\} \ y := -x \{y_0 = x \land x < 0 \land y = -x\}\) \hspace{1em} Assignment
   2. \(\{y = x \land x \geq 0\} \ skip \{y = x \land x \geq 0\}\) \hspace{1em} Assignment
   3. \(\{y = x\} \ if \ x < 0 \ then \ y := -x \ fi\)
   \(\{y_0 = x \land x < 0 \land y = -x\} \lor (y = x \land x \geq 0)\) \hspace{1em} Conditional 1, 2
   4. \(\{y_0 = x \land x < 0 \land y = -x\} \lor (y = x \land x \geq 0) \rightarrow y = abs(x)\) \hspace{1em} Predicate logic
   5. \(\{y = x\} \ if \ x < 0 \ then \ y := -x \ fi\)
   \(\{y = abs(x)\}\) \hspace{1em} Postcond. weak., 3, 4
2c. (Full outline to proof):

1. \( \{ -x = \text{abs}(x) \} y := -x \{ y = \text{abs}(x) \} \)  
   Assignment
2. \( \{ y = \text{abs}(x) \} \text{skip} \{ y = \text{abs}(x) \} \)  
   Skip
3. \( \{ p \} \text{if } x < 0 \text{ then } y := -x \{ y = \text{abs}(x) \} \)  
   Conditional 1, 2
   where \( p = (x < 0 \rightarrow -x = \text{abs}(x)) \wedge (x \geq 0 \rightarrow y = \text{abs}(x)) \)
4. \( y = x \rightarrow p \)  
   Predicate Logic
5. \( \{ y = x \} \text{if } x < 0 \text{ then } y := -x \{ y = \text{abs}(x) \} \)  
   Precond. str. 4, 3

3. Below, let \( W = \texttt{while } k > 0 \texttt{ do } k := k-1; s := s+k \texttt{ od} \)

1. \( \{ n \geq 0 \} k := n \{ n \geq 0 \wedge k = n \} \)  
   Assignment
2. \( \{ n \geq 0 \wedge k = n \} s := n \)  
   \( \{ n \geq 0 \wedge k = n \wedge s = n \} \)  
   Assignment
3. \( \{ n \geq 0 \} k := n; s := n \)  
   \( \{ n \geq 0 \wedge k = n \wedge s = n \} \)  
   Sequence 1, 2
4. \( n \geq 0 \wedge k = n \wedge s = n \rightarrow p \)  
   Predicate logic
5. \( \{ n \geq 0 \} k := n; s := n \{ p \} \)  
   Postcond. weak 3, 4
6. \( \{ p[s+k/s] \} s := s+k \{ p \} \)  
   Assignment
7. \( \{ p[s+k/s][k-1/k] \} k := k-1 \{ p[s+k/s] \} \)  
   Assignment
8. \( p \wedge k > 0 \rightarrow p[s+k/s][k-1/k] \)  
   Predicate logic
9. \( \{ p \wedge k > 0 \} k := k-1 \{ p[s+k/s] \} \)  
   Precond. str 8, 7
10. \( \{ p \wedge k > 0 \} k := k-1; s := s+k \{ p \} \)  
    Sequence 9, 6
11. \( \{ \text{inv } p \} W \{ p \wedge k \leq 0 \} \)  
    while 10
12. \( p \wedge k \leq 0 \rightarrow s = \text{sum}(0, n) \)  
    Predicate logic
13. \( \{ \text{inv } p \} W \{ s = \text{sum}(0, n) \} \)  
    Postcond. weak 12, 11
14. \( \{ n \geq 0 \} k := n; s := n; W \{ s = \text{sum}(0, n) \} \)  
    Sequence 5, 13