CS536, Spring 2022
Practice Midterm Exam #2
SOLUTIONS

Name _____

IIT Email _____

Important notes:

- This exam has 15 pages. Make sure you have them all.
- You have 75 minutes to complete the exam. We suggest looking through the questions first to see where to focus your time.
- Use only **blue or black pen** to complete this exam. If you don't have one, ask.
- Write your answers in the space provided. If you need more space for answers or scrap, you can use the back of the page, but clearly mark where your answers are.
- The last 5 pages of this exam are reference material. You may (carefully) tear them off and use them durng the exam. We do not need to collect these pages.
- You are permitted to refer to **two** double-sided 8.5" × 11" sheets of notes. We will collect your sheets of notes at the end of the exam, so if you want them back, please make sure your full name is clearly written on both pages (or the first one if they're attached). **No other outside aids (including electronics) or notes are permitted.**
- Sign the statement below:
  I have not used any unauthorized resources or received or given help during this exam.

Signed_____ Date_____

1 If Rule (8 points)

In class, we've seen two different rules for proving Hoare triples involving if, and we've seen how each can be proven using the other one and weakening, e.g., we can prove If' using If:

$$\dfrac{\dfrac{\vdash \{p \wedge e\}\ s_1\ \{q\} \qquad \vdash \{p \wedge \neg e\}\ s_2\ \{q\}}{\vdash \{p\}\ \mathsf{if}\ e\ \mathsf{then}\ \{s_1\}\ \mathsf{else}\ \{s_2\}\ \{q \vee q\}}\ (\textsc{If}) \qquad \dfrac{}{q \vee q \Rightarrow q}}{\vdash \{p\}\ \mathsf{if}\ e\ \mathsf{then}\ \{s_1\}\ \mathsf{else}\ \{s_2\}\ \{q\}}\ (\textsc{Weakening})$$

Here's another rule for if:

$$\dfrac{\vdash \{p_1\}\ s_1\ \{q\} \qquad \vdash \{p_2\}\ s_2\ \{q\}}{\vdash \{(e \to p_1) \wedge (\neg e \to p_2)\}\ \mathsf{if}\ e\ \mathsf{then}\ \{s_1\}\ \mathsf{else}\ \{s_2\}\ \{q\}}\ (\textsc{If"})$$

Show that this rule can be proven in terms of (one of) our existing if rules by writing a proof tree whose conclusion is $\vdash \{(e \to p_1) \wedge (\neg e \to p_2)\}\ \mathsf{if}\ e\ \mathsf{then}\ \{s_1\}\ \mathsf{else}\ \{s_2\}\ \{q\}$ and which uses $\vdash \{p_1\}\ s_1\ \{q\}$ and $\vdash \{p_2\}\ s_2\ \{q\}$ as axioms.

$$\dfrac{\dfrac{\vdash \{p_1\}\ s_1\ \{q\} \qquad (e \to p_1) \wedge (\neg e \to p_2) \wedge e \Rightarrow p_1}{\vdash \{(e \to p_1) \wedge (\neg e \to p_2) \wedge e\}\ s_1\ \{q\}}\ (\textsc{Weakening}) \qquad \dfrac{\vdash \{p_2\}\ s_2\ \{q\} \qquad (e \to p_1) \wedge (\neg e \to p_2) \wedge \neg e \Rightarrow p_2}{\vdash \{(e \to p_1) \wedge (\neg e \to p_2) \wedge \neg e\}\ s_2\ \{q\}}\ (\textsc{Weakening})}{\vdash \{(e \to p_1) \wedge (\neg e \to p_2)\}\ \mathsf{if}\ e\ \mathsf{then}\ \{s_1\}\ \mathsf{else}\ \{s_2\}\ \{q\}}\ (\textsc{If})$$

2 Concepts (18 points)

Answer each question in 2-5 English sentences. Be as clear and to-the-point as possible.

(a) (6 points) When we defined syntactic equality, we said that we would consider, e.g., $\forall x.P(x) \not\equiv \forall y.P(y)$, but that we might revisit this choice later. Give one good reason to consider $\forall x.P(x) \not\equiv \forall y.P(y)$ and one good reason to consider $\forall x.P(x) \equiv \forall y.P(y)$ (think about, e.g., what would happen if we tried to substitute $x + 2$ for x in both).

Syntactic equality has always meant whether the two predicates are written the same, up to parentheses, so these would not be syntactically equal. On the other hand, if we wanted to do the substitution $[x + 2/x](\forall x.P(x))$, we might need to change $\forall x.P(x)$ into $\forall y.P(y)$ using the fact that the names of bound variables don't matter. So if they don't matter, then we should consider the two $\equiv$.

(b) (6 points) Would the following rule be correct for proving total correctness of sequences? Why or why not?

$$\frac{\vdash [\ p\ ]\ s_1\ [\ q_1\ ] \qquad \vdash [\ q_1\ ]\ s_2\ [\ q\ ]}{\vdash [\ p\ ]\ s_1; s_2\ [\ q\ ]}$$

Yes. If s_1 terminates in a state that satisfies q_1, and whenever q_1 holds, s_2 terminates in a state that satisfies q, then $s_1; s_2$ terminates in a state that satisfies q.

(c) (6 points) Point out which step in the following substitution is incorrect, and briefly explain why it is incorrect.

$$\begin{aligned} & [x + 2/y](\forall x.P(y, x)) \\ = \ & [z + 2/y](\forall z.[z/x]P(y, x)) \\ = \ & [z + 2/y](\forall z.P(y, z)) \\ = \ & \forall z.[z + 2/y]P(y, z) \\ = \ & \forall z.P(z + 2, z) \end{aligned}$$

The second step is incorrect. We shouldn't change x to z in $x + 2$ since this x is free, not bound.

3 Proofs (30 points)

Do **either** proof A or proof B. For whichever one you choose:

a) Give appropriate loop invariants for any loops.

b) Write a **full** proof for the program. You may write a full proof outline or a Hilbert-style proof. (We would not recommend using a proof tree.)

If you start both, tell us which one you want us to grade: ___ Proof A ___ Proof B

Note: Proof A is quite a bit easier than Proof B. On the exam, the two proofs will be closer in difficulty, and likely closer to B's difficulty than A.

Proof A

This code from Exam 1 multiplies x and y (as long as $y \geq 0$).

Precondition:	$\{y \geq 0\}$
	$i := \overline{0};$
	$n := \overline{0};$
Program:	$\mathsf{while}(i < y)\{$
	$\quad n := n + x;$
	$\quad i := i + \overline{1}$
	$\}$
Postcondition:	$\{i = y \wedge n = x * y\}$

	$\{y \geq 0\}$
$i := \overline{0};$	$\{y \geq 0 \wedge i = 0\}$
$n := \overline{0};$	$\{y \geq 0 \wedge i = 0 \wedge n = 0\}$
$\{\mathbf{inv}\ i \leq y \wedge n = x * i\}$	
$\mathsf{while}(i < y)\{$	$\{i \leq y \wedge n = x * i \wedge i < y\} \Rightarrow \{i + 1 \leq y \wedge n + x = x * (i + 1)\}$
$\quad n := n + x;$	$\{i + 1 \leq y \wedge n + x = x * (i + 1)\}$
$\quad i := i + \overline{1}$	$\{i \leq y \wedge n = x * i\}$
$\}$	$\{i \leq y \wedge n = x * i \wedge i \geq y\}$
	$\Rightarrow \{i = y \wedge n = x * y\}$

Proof B

We're searching array a for value x; the precondition guarantees we'll succeed. **Hint:** include the precondition (maybe give it a shorthand like p_0) in the loop invariant. What else do you need to put in the loop invariant so that this, the precondition, and the negation of the loop condition all together imply the postcondition?

Precondition:	$\{\exists k \in [0, \lvert a\rvert - 1].a[k] = x\}$
	$i := \overline{0}$;
	$j := \overline{0}$;
Program:	while$(i < size(a))\{$
	if $a[i] = x$ then $\{j := i\}$ else $\{$skip$\}$;
	$i := i + \overline{1}$
	$\}$
Postcondition:	$\{a[j] = x\}$

Let $p_0 = \exists k \in [0, |a| - 1].a[i] = x$.

	$\{\exists k \in [0, \lvert a\rvert - 1].a[i] = x\}$
$i := \overline{0}$;	$\{(\exists k \in [0, \lvert a\rvert - 1].a[i] = x) \wedge i = 0\}$
$j := \overline{0}$;	$\{(\exists k \in [0, \lvert a\rvert - 1].a[i] = x) \wedge i = 0 \wedge j = 0\}$
{**inv** $p = p_0 \wedge i \leq \lvert a\rvert$	$\wedge(a[j] = x \vee \forall k \in [0, i-1].a[k] \neq x)\}$
while$(i < size(a))\{$	$\{p \wedge i < \lvert a\rvert\}$
if $a[i] = x$ then $\{$	$\{p \wedge i < \lvert a\rvert \wedge a[i] = x\} \Rightarrow \{p_0 \wedge i + 1 \leq \lvert a\rvert \wedge (a[i] = x \vee \forall k \in [0, i].a[k] \neq x)\}$
$j := i$	$\{p_0 \wedge i + 1 \leq \lvert a\rvert \wedge (a[j] = x \vee \forall k \in [0, i].a[k] \neq x)\}$
$\}$ else $\{$	$\{p \wedge i < \lvert a\rvert \wedge a[i] \neq x\}$
skip	$\{p \wedge i < \lvert a\rvert \wedge a[i] \neq x\} \Rightarrow \{p_0 \wedge i + 1 \leq \lvert a\rvert \wedge (a[j] = x \vee \forall k \in [0, i].a[k] \neq x)\}$
$\}$;	$\{p_0 \wedge i + 1 \leq \lvert a\rvert \wedge (a[j] = x \vee \forall k \in [0, i].a[k] \neq x)\}$
$i := i + \overline{1}$	$\{p\}$
$\}$	$\{p \wedge i \geq \lvert a\rvert\} \Rightarrow \{a[j] = x\}$

4 Weakest Preconditions and Strongest Postconditions (24 points)

Calculate the following. Be sure to note whether we're asking for wp, wlp or sp. For full credit, **show your work** (this will also let us potentially give you partial credit if your final answer is wrong). You **do not need to** simplify your answers further once you have calculated a valid wp, wlp or sp.

(a) (6 points) $wlp(a[j] := -1, \forall i \in [0, |a| - 1].a[i] > 0)$

$$\begin{aligned} & [-1/a[j]](\forall i \in [0, |a| - 1].a[i] > 0) \\ = \; & \forall i \in [0, |a| - 1], i = j \; ? \; -1 > 0 : a[i] > 0 \\ (\Leftrightarrow \; & j \notin [0, |a| - 1]) \end{aligned}$$

(b) (12 points) $wp(x := y; z := sqrtx, z \leq 4)$

$$\begin{aligned} & wlp(x := y; z := sqrtx, z \leq 4) \wedge D(x := y; z := sqrtx) \\ = \; & wlp(x := y, wlp(z := sqrtx, z \leq 4)) \wedge D(x := y) \wedge wp(x := y, D(sqrtx)) \\ = \; & wlp(x := y, \sqrt{x} \leq 4) \wedge T \wedge wlp(x := y, x \geq 0) \wedge T \\ = \; & \sqrt{y} \leq 4 \wedge y \geq 0 \\ (\Leftrightarrow \; & 0 \leq y \leq 16) \end{aligned}$$

(c) (6 points) $sp(T, \textsf{if } x < y \textsf{ then } \{\textsf{skip}\} \textsf{ else } \{x := y + 1\})$

$$\begin{aligned} & sp(x < y, \textsf{skip}) \vee sp(x \geq y, x := y + 1) \\ = \; & x < y \vee (x_0 \geq y \wedge x = y + 1) \end{aligned}$$

5 Filtering an Array (20 points)

The precondition for this program is T and the postcondition is $(\forall k \in [0, j-1].b[k] > 0) \wedge i = |a|$.

$i := \overline{0};$
$j := \overline{0};$
$\{\textbf{inv}\ i \leq |a| \wedge \forall k \in [0, j-1].b[k] > 0\}$
$\mathsf{while}(i < size(a))\{$
  $\mathsf{if}\ a[i] > 0\ \mathsf{then}\ \{b[j] := a[i]; j := j + \overline{1}\}$
  $\mathsf{else}\ \{\mathsf{skip}\}$
  $i := i + \overline{1};$
$\}$

Let $p_1 = i \leq |a| \wedge \forall k \in [0, j-1].b[k] > 0$
and $p_2 = i < |a| \wedge \forall k \in [0, j-1].b[k] > 0$
and $p_3 = i < |a| \wedge (\forall k \in [0, j].b[k] > 0)$
and $s_0 = \mathsf{if}\ a[i] > 0\ \mathsf{then}\ \{b[j] := a[i]; j := j + \overline{1}\}\ \mathsf{else}\ \{\mathsf{skip}\}$
and $s = s_0; i := i + \overline{1}$.

(a) (12 points) Fill in the blanks in the following Hilbert-style proof for the body of the loop.

1	$\{p_3\}\ j := j + \overline{1}\ \{p_2\}$	AssignBwd
2	$\{i < \|a\| \wedge i \leq \|a\| \wedge (\forall k \in [0, j].(j = k\ ?\ a[i] : b[k]) > 0)\}\ b[j] := a[i]\ \{p_3\}$	AssignBwd
3	$\{i < \|a\| \wedge a[i] > 0 \wedge p_1\}\ b[j] := a[i]\ \{p_3\}$	Weakening 2
4	$\{i < \|a\| \wedge a[i] > 0 \wedge p_1\}\ b[j] := a[i]; j := j + \overline{1}\ \{p_2\}$	Sequence 3, 2
5	$\{i < \|a\| \wedge a[i] \leq 0 \wedge p_1\}\ \mathsf{skip}\ \{i < \|a\| \wedge a[i] \leq 0 \wedge p_1\}$	Skip
6	$\{i < \|a\| \wedge a[i] \leq 0 \wedge p_1\}\ \mathsf{skip}\ \{p_2\}$	Weakening 5
7	$\{i < \|a\| \wedge p_1\}\ s_0\ \{p_2\}$	If' 4, 6
8	$\{i + 1 \leq \|a\| \wedge (\forall k \in [0, j-1].b[k] > 0)\}\ i := i + \overline{1}\ \{p_1\}$	AssignBwd
9	$\{p_2\}\ i := i + \overline{1}\ \{p_1\}$	Weaken 8
10	$\{i < \|a\| \wedge p_1\}\ s\ \{p_1\}$	Sequence 7, 9

(b) (4 points) For the loop invariant given, is the given postcondition the strongest postcondition for the program? Explain why or why not.

Yes. The sp for a while loop is $p \wedge \neg e$.

(c) (4 points) If we change the loop invariant, we can get a stronger postcondition. Give a valid postcondition for the program that is stronger than the given postcondition. You **do not** need to give the loop invariant that would let us prove it.

For example,

$$(\forall k \in [0, j-1].b[k] > 0 \wedge \exists i.a[i] = b[k]) \wedge i = |a|$$

A Logic Laws

Name	Description
Simplify	$p \wedge q \Rightarrow p, q$
Modus Ponens	$(p \rightarrow q), p \Rightarrow q$
Conjunction	$p, q \Rightarrow p \wedge q$
Disjunction	$p \Rightarrow p \vee q, q \vee p$
Definition of Conditional	$p \rightarrow q \Leftrightarrow \neg p \vee q$
Definition of Biconditional	$p \leftrightarrow q \Leftrightarrow (p \rightarrow q) \wedge (q \rightarrow p)$
Law of the Excluded Middle (LEM)	$p \vee \neg p \Leftrightarrow T$
Double Negation Elimination (DNE)	$p \Leftrightarrow \neg\neg p$
Contradiction	$p \wedge \neg p \Leftrightarrow F$
Identity	$p \wedge T \Rightarrow p$, $p \vee F \Rightarrow p$
DeMorgan's Laws	$\neg(p \wedge q) \Leftrightarrow \neg p \vee \neg q$ $\neg(p \vee q) \Leftrightarrow \neg p \wedge \neg q$ $\neg(\forall x.p(x)) \Leftrightarrow \exists x.\neg p(x)$ $\neg(\exists x.p(x)) \Leftrightarrow \forall x.\neg p(x)$
Distributivity	$(p \wedge q) \vee r \Leftrightarrow (p \vee r) \wedge (q \vee r)$ $(p \vee q) \wedge r \Leftrightarrow (p \wedge r) \vee (q \wedge r)$
Commutativity	$p \wedge q \Leftrightarrow q \wedge p$, $p \vee q \Leftrightarrow q \vee p$
Associativity	$(p \wedge q) \wedge r \Leftrightarrow p \wedge (q \wedge r)$, $(p \vee q) \vee r \Leftrightarrow p \vee (q \vee r)$
Idempotency	$p \wedge p \Leftrightarrow p$, $p \vee p \Leftrightarrow p$
Domination	$p \vee T \Leftrightarrow T$, $p \wedge F \Leftrightarrow F$

B Language Syntax and Semantics

Expression and Statement Syntax

$$
\begin{array}{rcl}
e & ::= & \overline{n} \mid \mathsf{true} \mid \mathsf{false} \mid x \mid a[e] \mid e \; op \; e \mid e \; ? \; e : e \mid size(a) \\
s & ::= & \mathsf{skip} \mid s; s \mid x := e \mid a[e] := e \mid \mathsf{if} \; e \; \mathsf{then} \; \{s\} \; \mathsf{else} \; \{s\} \mid \mathsf{while} \; e \; \{s\}
\end{array}
$$

Expression Semantics

$$
\begin{array}{rcll}
\sigma(\overline{n}) & = & n & \\
\sigma(\mathsf{true}) & = & T & \\
\sigma(\mathsf{false}) & = & F & \\
\sigma(x) & = & \sigma(x) & \\
\sigma(a[e]) & = & (\sigma(a))[\sigma(e)] & \sigma(e) \neq \bot_e \wedge 0 \leq \sigma(e) < |\sigma(a)| \\
\sigma(a[e]) & = & \bot_e & \text{otherwise} \\
\sigma(e_1 \; op \; e_2) & = & \sigma(e_1) \; op \; \sigma(e_2) & \sigma(e_1) \neq \bot_e \neq \sigma(e_2) \\
\sigma(e_1 \; op \; e_2) & = & \bot_e & \sigma(e_1) = \bot_e \vee \sigma(e_2) = \bot_e \\
\sigma(e_1 \; ? \; e_2 : e_3) & = & \sigma(e_2) & \sigma(e_1) = T \\
\sigma(e_1 \; ? \; e_2 : e_3) & = & \sigma(e_3) & \sigma(e_1) = F \\
\sigma(e_1 \; ? \; e_2 : e_3) & = & \bot_e & \sigma(e_1) = \bot_e \\
\sigma(size(a)) & = & |\sigma(a)| &
\end{array}
$$

Statement Semantics - Small-step

$$\frac{\langle s_1, \sigma\rangle \to \langle s_1', \sigma\rangle}{\langle s_1; s_2, \sigma\rangle \to \langle s_1'; s_2, \sigma\rangle} \qquad \frac{\langle s_1, \sigma\rangle \to \langle \mathsf{skip}, \bot_e\rangle}{\langle s_1; s_2, \sigma\rangle \to \langle \mathsf{skip}, \bot_e\rangle} \qquad \frac{}{\langle \mathsf{skip}; s, \sigma\rangle \to \langle s, \sigma\rangle}$$

$$\frac{\sigma(e) \neq \bot_e}{\langle x := e, \sigma\rangle \to \langle \mathsf{skip}, \sigma[x \mapsto \sigma(e)]\rangle} \qquad \frac{\sigma(e) = \bot_e}{\langle x := e, \sigma\rangle \to \langle \mathsf{skip}, \bot_e\rangle}$$

$$\frac{\sigma(e_1) \neq \bot_e \qquad \sigma(e_2) \neq \bot_e \qquad 0 \leq \sigma(e_1) < |\sigma(a)|}{\langle a[e_1] := e_2, \sigma\rangle \to \langle \mathsf{skip}, \sigma[a[\sigma(e_1)] \mapsto \sigma(e_2)]\rangle} \qquad \frac{\sigma(e_1) = \bot_e \vee \sigma(e_2) = \bot_e}{\langle a[e_1] := e_2, \sigma\rangle \to \langle \mathsf{skip}, \bot_e\rangle}$$

$$\frac{\sigma(e_1) \geq |\sigma(a)| \vee \sigma(e_1) < 0}{\langle a[e_1] := e_2, \sigma\rangle \to \langle \mathsf{skip}, \bot_e\rangle} \qquad \frac{\sigma(e) = T}{\langle \mathsf{if} \; e \; \mathsf{then} \; \{s_1\} \; \mathsf{else} \; \{s_2\}, \sigma\rangle \to \langle s_1, \sigma\rangle}$$

$$\frac{\sigma(e) = F}{\langle \mathsf{if} \; e \; \mathsf{then} \; \{s_1\} \; \mathsf{else} \; \{s_2\}, \sigma\rangle \to \langle s_2, \sigma\rangle} \qquad \frac{\sigma(e) = \bot_e}{\langle \mathsf{if} \; e \; \mathsf{then} \; \{s_1\} \; \mathsf{else} \; \{s_2\}, \sigma\rangle \to \langle \mathsf{skip}, \bot_e\rangle}$$

$$\frac{}{\langle \mathsf{while} \; e \; \{s\}, \sigma\rangle \to \langle \mathsf{if} \; e \; \mathsf{then} \; \{s; \mathsf{while} \; e \; \{s\}\} \; \mathsf{else} \; \{\mathsf{skip}\}, \sigma\rangle}$$

Statement Semantics - Big-step

$$
\begin{array}{rcll}
M(\mathsf{skip}, \sigma) & = & \{\sigma\} & \\
M(s_1; s_2, \sigma) & = & \bigcup_{\sigma' \in M(s_1,\sigma)} M(s_2, \sigma') & \\
M(x := e, \sigma) & = & \{\sigma[x \mapsto \sigma(e)]\} & \sigma(e) \neq \bot_e \\
M(x := e, \sigma) & = & \{\bot_e\} & \sigma(e) = \bot_e \\
M(a[e_1] := e_2, \sigma) & = & \{\sigma[a[\sigma(e_1)] \mapsto \sigma(e_2)]\} & \sigma(e_1) \neq \bot_e \wedge \sigma(e_2) \neq \bot_e \wedge 0 \leq \sigma(e_1) < |\sigma(a)| \\
M(a[e_1] := e_2, \sigma) & = & \{\bot_e\} & \text{otherwise} \\
M(\mathsf{if}\ e\ \mathsf{then}\ \{s_1\}\ \mathsf{else}\ \{s_2\}, \sigma) & = & M(s_1, \sigma) & \sigma(e) = T \\
M(\mathsf{if}\ e\ \mathsf{then}\ \{s_1\}\ \mathsf{else}\ \{s_2\}, \sigma) & = & M(s_2, \sigma) & \sigma(e) = F \\
M(\mathsf{if}\ e\ \mathsf{then}\ \{s_1\}\ \mathsf{else}\ \{s_2\}, \sigma) & = & \{\bot_e\} & \sigma(e) = \bot_e \\
M(\mathsf{while}\ e\ \{s\}, \sigma) & = & \Sigma_k & \Sigma_k \text{ is the lowest } k \text{ such that if} \\
 & & & \sigma \in \Sigma_k \text{ , then } \sigma(e) = F \\
M(\mathsf{while}\ e\ \{s\}, \sigma) & = & \{\bot_d\} & \text{no such } k \text{ exists}
\end{array}
$$

where

$$
\begin{array}{rcl}
\Sigma_0 & = & \{\sigma\} \\
\Sigma_k + 1 & = & \bigcup_{\sigma \in \Sigma_k} M(s, \sigma)
\end{array}
$$

C Hoare Triple Inference Rules

$$\frac{}{\vdash \{p\}\ \mathsf{skip}\ \{p\}}\ (\textsc{Skip}) \qquad \frac{}{\vdash \{[e/x]p\}\ x := e\ \{p\}}\ (\textsc{AssignBwd})$$

$$\frac{}{\vdash \{[e/a[i]]p\}\ a[i] := e\ \{p\}}\ (\textsc{ArrAssignBwd}) \qquad \frac{x_0\ \mathsf{fresh}}{\vdash \{p\}\ x := e\ \{[x_0/x]p\} \wedge x = [x_0/x]e}\ (\textsc{AssignFwd})$$

$$\frac{\vdash \{p\}\ s_1\ \{q'\} \qquad \vdash \{q'\}\ s_2\ \{q\}}{\vdash \{p\}\ s_1; s_2\ \{q\}}\ (\textsc{Seq}) \qquad \frac{\vdash \{p \wedge e\}\ s_1\ \{q_1\} \qquad \vdash \{p \wedge \neg e\}\ s_2\ \{q_2\}}{\vdash \{p\}\ \mathsf{if}\ e\ \mathsf{then}\ \{s_1\}\ \mathsf{else}\ \{s_2\}\ \{q_1 \vee q_2\}}\ (\textsc{If})$$

$$\frac{\vdash \{p \wedge e\}\ s_1\ \{q\} \qquad \vdash \{p \wedge \neg e\}\ s_2\ \{q\}}{\vdash \{p\}\ \mathsf{if}\ e\ \mathsf{then}\ \{s_1\}\ \mathsf{else}\ \{s_2\}\ \{q\}}\ (\textsc{If'}) \qquad \frac{\vdash \{p \wedge e\}\ s\ \{p\}}{\vdash \{p\}\ \mathsf{while}\ e\ \{s\}\ \{p \wedge \neg e\}}\ (\textsc{While})$$

D Calculating wp, wlp, sp for loop-free programs

$\oplus$ stands for any binary operator that doesn't itself cause errors, e.g., +, -...

$$\begin{array}{rcl@{\qquad\qquad}rcl} wlp(\mathsf{skip}, q) & = & q & wlp(x := e, q) & = & [e/x]q \\ wlp(a[e_1] := e_2, q) & = & [e_2/a[e_1]]q & wlp(s_1; s_2, q) & = & wlp(s_1, wlp(s_2, q)) \end{array}$$

$$wlp(\mathsf{if}\ e\ \mathsf{then}\ \{s_1\}\ \mathsf{else}\ \{s_2\}, q) = (e \to wlp(s_1, q)) \wedge (\neg e \to wlp(s_2, q))$$

$$\begin{array}{rcl@{\qquad\qquad}rcl} sp(p, \mathsf{skip}) & = & p & sp(p, x := e) & = & [x_0/x]p \wedge x = [x_0/x]e \\ sp(p, s_1; s_2) & = & sp(sp(p, s_1), s_2) & & & \end{array}$$

$$sp(p, \mathsf{if}\ e\ \mathsf{then}\ \{s_1\}\ \mathsf{else}\ \{s_2\}) = sp(p \wedge e, s_1) \vee sp(p \wedge \neg e, s_2)$$

$$\begin{array}{rcl@{\qquad\qquad}rcl} D(c) & = & T & D(x) & = & T \\ D(a[e]) & = & D(e) \wedge 0 \leq e < |a| & D(e_1/e_2) & = & D(e_1) \wedge D(e_2) \wedge e_2 \neq 0 \\ D(sqrt(e)) & = & D(e) \wedge e \geq 0 & D(e_1 \oplus e_2) & = & D(e_1) \wedge D(e_2) \end{array}$$

$$D(e_1\ ?\ e_2 : e_3) = D(e_1) \wedge (e_1 \to D(e_2)) \wedge (\neg e_1 \to D(e_2))$$

$$\begin{array}{rcl@{\qquad\qquad}rcl} D(\mathsf{skip}) & = & T & D(x := e) & = & D(e) \\ D(a[e_1] := e_2) & = & D(a[e_1]) \wedge D(e_2) & D(s_1; s_2) & = & D(s_1) \wedge wp(s_1, D(s_2)) \\ D(\mathsf{while}\ e\ \{s\}) & = & D(e) \wedge (e \to D(s_1)) & & & \end{array}$$

$$D(\mathsf{if}\ e\ \mathsf{then}\ \{s_1\}\ \mathsf{else}\ \{s_2\}) = D(e) \wedge (e \to D(s_1)) \wedge (\neg e \to D(s_2))$$

$$wp(s, q) = wlp(s, q) \wedge D(s)$$

E Simplifying Conditional Expressions

$$\begin{array}{rcll@{\qquad\qquad}rcll} T\ ?\ e_1 : e_2 & \Rrightarrow & e_1 & \text{Always} & F\ ?\ e_1 : e_2 & \Rrightarrow & e_2 & \text{Always} \\ e_0\ ?\ e : e & \Rrightarrow & e & \text{Always} & & & & \\ e_0\ ?\ e_1 : e_2 & \Rrightarrow & e_2 & \text{If } e_0 \Rightarrow e_1 = e_2 & e_0\ ?\ e_1 : e_2 & \Rrightarrow & e_1 & \text{If } \neg e_0 \Rightarrow e_1 = e_2 \end{array}$$

Let Θ be a unary operator, $\oplus$ be a binary operator or relation and f be any function.

$$\begin{array}{rcl@{\qquad\qquad}rcl} \Theta(e\ ?\ e_1 : e_2) & \Rrightarrow & e\ ?\ \Theta(e_1) : \Theta(e_2) & (e\ ?\ e_1 : e_2) \oplus e_3 & \Rrightarrow & e\ ?\ e_1 \oplus e_3 : e_2 \oplus e_3 \\ a[e\ ?\ e_1 : e_2] & \Rrightarrow & e\ ?\ a[e_1] : a[e_2] & f(e\ ?\ e_1 : e_2) & \Rrightarrow & e\ ?\ f(e_1) : f(e_2) \end{array}$$

If e, e_1, and e_2 are Boolean expressions, then

$$\begin{array}{rcl@{\qquad\qquad}rcl} (e\ ?\ e_1 : F) & \Leftrightarrow & (e \wedge e_1) & (e\ ?\ F : e_2) & \Leftrightarrow & (\neg e \wedge e_2) \\ (e\ ?\ e_1 : T) & \Leftrightarrow & (e \to e_1) \Leftrightarrow (\neg e \vee e_1) & (e\ ?\ T : e_2) & \Leftrightarrow & (\neg e \to e_2) \Leftrightarrow (e \vee e_2) \end{array}$$

$$(e\ ?\ e_1 : e_2) \Leftrightarrow ((e \to e_1) \wedge (\neg e \to e_2)) \Leftrightarrow ((e \wedge e_1) \vee (\neg e \wedge e_2))$$

F Algorithm For Expanding Proof Outlines

A. Add a precondition:

(a) Prepend $\{wp(x := e, q)\}$ to $x := e$ $\{q\}$.

(b) Prepend $\{q\}$ to skip $\{q\}$.

(c) Prepend some p to s_2 in $s_1; s_2$ $\{q\}$ to get $s_1; \{p\}$ s_2 $\{q\}$.

(d) Add preconditions to the branches of an if-else:
Turn $\{p\}$ if e then $\{s_1\}$ else $\{s_2\}$ into $\{p\}$ if e then $\{\{p \wedge e\}$ $s_1\}$ else $\{\{p \wedge \neg e\}$ $s_2\}$.

(e) Add a precondition to an if-else:
Prepend $\{(e \to p_1) \wedge (\neg e \to p_2)\}$ to if e then $\{\{p_1\}$ $s_1\}$ else $\{\{p_2\}$ $s_2\}$.

B. Or add a postcondition:

(a) Append $\{sp(p, x := e)\}$ to $\{p\}$ $x := e$.

(b) Append $\{p\}$ to $\{p\}$ skip.

(c) Append some q to s_1 in $\{p\}$ $s_1; s_2$ to get $\{p\}$ s_1 $\{q\}; s_2$.

(d) Add postconditions to the branches of an if-else:
Turn if e then $\{s_1\}$ else $\{s_2\}$ $\{q_1 \vee q_2\}$ into if e then $\{s_1$ $\{q_1\}\}$ else $\{s_2$ $\{q_2\}\}$ $\{q_1 \vee q_2\}$
or if e then $\{s_1\}$ else $\{s_2\}$ $\{q\}$ into if e then $\{s_1$ $\{q\}\}$ else $\{s_2$ $\{q\}\}$ $\{q\}$

(e) Add a postcondition to an if-else:
Append $\{q_1 \vee q_2\}$ to if e then $\{s_1$ $\{q_1\}\}$ else $\{s_2$ $\{q_2\}\}$.

C. Or add loop conditions:

(a) Take a loop and add pre- and post-conditions to the loop body; add a postcondition for the loop:
Turn $\{\mathbf{inv}\ p\}$ while e $\{s\}$ into:
$\{\mathbf{inv}\ p\}$ while e $\{\{p \wedge e\}$ s $\{p\}\}$ $\{p \wedge \neg e\}$

D. Or strengthen or weaken some condition:

(a) Turn $\{q\}$ into $\{p\} \Rightarrow \{q\}$ for some p where $p \Rightarrow q$.

(b) Turn $\{p\}$ into $\{p\} \Rightarrow \{q\}$ for some q where $p \Rightarrow q$.

G Substitution for Array Elements

If $a \neq b$: $[e/a[e_1]](b[e_2]) = b[e_2']$ where $e_2' = [e/a[e_1]]e_2$.
General case: $[e_0/a[e_1]](a[e_2]) = (e_2' = e_1\ ?\ e_0 : a[e_2'])$ where $e_2' = [e_0/a[e_1]](e_2)$.
Special case where k is constant or variable: $[e/a[e_1]](a[k]) = (k = e\ ?\ e_1 : a[k])$.