HW 4 – Floyd-Hoare Logic

CS 477 – Spring 2014
Revision 1.1

Assigned March 6, 2014
Due March 13, 2014, 11:59 pm
Extension 48 hours (20% penalty)

1 Change Log

1.1 Corrected typos and the line overrun in the code in Problem 5.

1.0 Initial Release.

2 Objectives and Background

The purpose of this HW is to test your understanding of

- proving correctness of a program using Floyd-Hoare Logic

Another purpose of HWs is to provide you with experience answering non-programming written questions of the kind you may experience on the midterm and final.

3 Turn-In Procedure

The pdf for this assignment (`hw4.pdf`) should be found in the `assignments/hw4/` subdirectory of your svn directory for this course. Your solutions to the problems should be put in that same directory. Using your favorite tool(s), you should put your solution to the handwritten problems in a file named `hw4-submission.pdf`, and the solution to he Isabelle problem should go in a file named `hw4.thy`. A stub file for `hw4.thy` has already been put in your directory as well. If you have problems generating a pdf, please seek help from the course staff. Your answers to the following questions are to be submitted electronically from within `assignments/hw4/` subdirectory by committing the file as follows:

```
svn add hw4-submission.pdf
svn commit -m "Turning in hw4"
```

This will commit both your solution to the handwritten problems and the solution to Isabelle problem (because hw4.thy already exists in your directory).

4 Handwritten Problems

Give a proof in Floyd-Hoare Logic of each of the following Hoare triples. You should state clearly which rule you are using at each step.

1. (10pts) $\{x > 1 \wedge y > 0\}\ if\ y > 1\ then\ z := x * y\ else\ z := x/y\ fi\ \ \{z \geq x \wedge z > y\}$
   In this problem, the variables range over real numbers.

Solution:
Let $ThenTree =$

$$\dfrac{(1)(x>1\wedge y>0\wedge y>1)\Rightarrow((x*y)\geq x\wedge(x*y)>y) \qquad \dfrac{}{\{(x*y)\geq x\wedge(x*y)>y\}\ z:=x*y\ \{z\geq x\wedge z>y\}}\,\text{AsignAx}}{\{x>1\wedge y>0\wedge y>1\}\ z:=x*y\ \{z\geq x\wedge z>y\}}\,\text{PrecondStr}$$

where (1) holds because $(x>1\wedge y>0)\Rightarrow(x*y)>y$ and $(x>1\wedge y>1)\Rightarrow(x>0\wedge y>1)\Rightarrow(x*y)\geq x$

Let $ElseTree =$

$$\dfrac{(2)(x>1\wedge y>0\wedge\neg(y>1))\Rightarrow((x/y)\geq x\wedge(x/y)>y) \qquad \dfrac{}{\{(x/y)\geq x\wedge(x/y)>y\}\ z:=x*y\ \{z\geq x\wedge z>y\}}\,\text{AsignAx}}{\{x>1\wedge y>0\wedge\neg(y>1)\}\ z:=x/y\ \{z\geq x\wedge z>y\}}\,\text{PrecondStr}$$

where (2) holds because $\neg(y>1)\Rightarrow y\leq 1$, and $x>1\Rightarrow x>0$ and $(x>0\wedge y>0\wedge y\leq 1)\Rightarrow(x/y)\geq x$, and $(x/y)\geq x\wedge x>1\Rightarrow(x/y)>1$ and $((x/y)>1\wedge y\leq 1)\Rightarrow(x/y)>y$, and thus $(x>1\wedge y>0\wedge\neg(y>1))\Rightarrow((x/y)\geq x\wedge(x/y)>y)$.

Then

$$\dfrac{ThenTree \qquad\qquad ElseTree}{\{x>1\wedge y>0\}\ if\ y>1\ then\ z:=x*y\ else\ z:=x/y\ fi\ \{z\geq x\wedge z>y\}}\,\text{IfThenElseRule}$$

2. (15 pts) $\{n>0\}\ i:=n;\ j:=0;\ while\ i\geq 0\ do\ (j:=j+i; i:=i-1)\ od\ \{j=(n\times(n+1))/2\}$
In this problem, the variables range over the integers.

Solution:
Let $Setup =$

$$\dfrac{\dfrac{\begin{array}{c}(1)\\ n>0\Rightarrow\\ (2\times 0=\\ (n\times(n+1))\\ -(n\times(n+1)))\\ \wedge\,(n\geq -1)\end{array} \qquad \dfrac{}{\begin{array}{c}\{(2\times 0=(n\times(n+1))\\ -(n\times(n+1)))\\ \wedge\,(n\geq -1)\}\\ i:=n\\ \{(2\times 0=(n\times(n+1))\\ -(i\times(i+1)))\\ \wedge\,(i\geq -1)\}\end{array}}\,\text{AssignAx}}{\begin{array}{c}\{n>0\}\\ i:=n\\ \{(2\times 0=(n\times(n+1))-(i\times(i+1)))\\ \wedge\,(i\geq -1)\}\end{array}}\,\text{PrecondStr} \qquad \dfrac{}{\begin{array}{c}\{(2\times 0=(n\times(n+1))-(i\times(i+1)))\\ \wedge\,(i\geq -1)\}\\ j:=0\\ \{(2\times j=(n\times(n+1))-(i\times(i+1)))\\ \wedge\,(i\geq -1)\}\end{array}}\,\text{AssignAx}}{\{n>0\}\ i:=n;\ j:=0\ \{(2\times j=(n\times(n+1))-(i\times(i+1)))\wedge(i\geq -1)\}}\,\text{SeqRule}$$

where (1) is true because $2\times 0=0$ and $(n\times(n+1))-(n\times(n+1)))=0$ so $2\times 0=(n\times(n+1))-(n\times(n+1)))$, and $n>0\Rightarrow n\geq -1$.

Let $WhileLoop =$

$$
\dfrac{
\dfrac{
\dfrac{
\begin{array}{c} (2) \\ (2\times j = \\ (n\times(n+1)) \\ -(i\times(i+1))) \\ \wedge\,(i\geq -1) \\ \wedge\,(i\geq 0) \\ \Rightarrow \\ (2\times(j+i) = \\ (n\times(n+1)) \\ -((i-1)\times((i-1)+1))) \\ \wedge\,((i-1)\geq -1) \end{array}
\qquad
\dfrac{}{\begin{array}{c} \{(2\times(j+i) = \\ (n\times(n+1)) \\ -((i-1)\times((i-1)+1))) \\ \wedge\,((i-1)\geq -1)\} \\ j := j+i \\ \{(2\times j = \\ (n\times(n+1)) \\ -((i-1)\times((i-1)+1))) \\ \wedge\,((i-1)\geq -1)\} \end{array}}\;\textsf{AssignAx}
}{
\begin{array}{c} \{(2\times j = \\ (n\times(n+1)) \\ -(i\times(i+1))) \\ \wedge\,(i\geq -1) \\ \wedge\,(i\geq 0)\} \\ j := j+i \\ \{(2\times j = \\ (n\times(n+1)) \\ -((i-1)\times((i-1)+1))) \\ \wedge\,((i-1)\geq -1)\} \end{array}
}\;\textsf{PrecondStr}
\qquad
\dfrac{}{\begin{array}{c} \{(2\times j = \\ (n\times(n+1)) \\ -((i-1)\times((i-1)+1))) \\ \wedge\,((i-1)\geq -1)\} \\ i := i-1 \\ \{(2\times j = \\ (n\times(n+1)) \\ -(i\times(i+1))) \\ \wedge\,(i\geq -1)\} \end{array}}\;\textsf{AssignAx}
}{
\begin{array}{c} \{(2\times j = (n\times(n+1)) - (i\times(i+1))) \wedge (i\geq -1) \wedge (i\geq 0)\} \\ j := j+i; i := i-1 \\ \{(2\times j = (n\times(n+1)) - (i\times(i+1))) \wedge (i\geq -1)\} \end{array}
}\;\textsf{SeqRule}
}{
\begin{array}{c} \{(2\times j = (n\times(n+1)) - (i\times(i+1))) \wedge (i\geq -1)\} \\ \mathit{while}\ i\geq 0\ \mathit{do}\ (j := j+i; i := i-1)\ \mathit{od} \\ \{(2\times j = (n\times(n+1)) - (i\times(i+1))) \wedge (i\geq -1) \wedge \neg(i>0)\} \end{array}
}\;\textsf{WhileRule}
$$

where (2) is true because $(n\times(n+1))-(i\times(i+1))) = n^2+n-i^2-i$ and $(n\times(n+1))-((i-1)\times((i-1)+1))) = n^2+n-i^2+i = (n^2+n-i^2-i)+(2\times i)$ and since $2\times j = (n\times(n+1))$ $-(i\times(i+1)))$ we have

$$
\begin{aligned}
2\times(j+i) &= (2\times j)+(2\times i) \\
&= (n^2+n-i^2-i)+(2\times i) \\
&= (n\times(n+1)) - ((i-1)\times((i-1)+1)))
\end{aligned}
$$

For the second conjunct,

Then

$$
\dfrac{
\dfrac{Setup \qquad WhileLoop}{
\begin{array}{c} \{n>0\} \\ i := n;\ j := 0;\ \mathit{while}\ i\geq 0\ \mathit{do}\ (j := j+i; i := i-1)\ \mathit{od} \\ \{(2\times j = (n\times(n+1)) - (i\times(i+1))) \wedge (i\geq -1) \wedge \neg(i>0)\} \end{array}
}\;\textsf{SeqRule}
\qquad SideCond1
}{
\{n>0\}\ i := n;\ j := 0;\ \mathit{while}\ i\geq 0\ \mathit{do}\ (j := j+i; i := i-1)\ \mathit{od}\ \{j = (n\times(n+1))/2\}
}\;\textsf{PostcondWk}
$$

where $SideCond1 = ((2\times j = (n\times(n+1)) - (i\times(i+1))) \wedge (i\geq -1) \wedge \neg(i>0)) \Rightarrow (j = (n\times(n+1))/2)$, and $SideCond1$ is true because $((i \geq -1) \wedge \neg(i > 0)) \Rightarrow i = -1$ and $(i = -1) \Rightarrow (i+1) = 0$ and $((i+1) = 0) \Rightarrow (i \times (i+1) = 0)$ and $(i \times (i+1) = 0) \Rightarrow ((n \times (n+1)) - (i \times (i+1)) = (n \times (n+1)))$ and thus $(2 \times j = (n \times (n+1)) - (i \times (i+1))) \Rightarrow (2 \times j = (n \times (n+1)))$ and since one of n or $n+1$ must be even, $(2 \times j = (n \times (n+1))) \Rightarrow (j = (n \times (n+1))/2)$.

5 Extra Credit

3. (5 pts) $\{a > 0 \wedge b > 0\}$
   $m := a;$
   $n := b;$
   $while\ n \neq m\ do (if\ m < n\ then\ n := n - m\ else\ m := m - n\ fi)\ od$
   $\{a\ mod\ m = 0 \wedge b\ mod\ m = 0\}$

   In this problem, the variables range over the integers.

   Solution:
   Let $IfThenElseProof =$

$$
\dfrac{
\dfrac{
\begin{array}{l}
(1) \\
(\forall d.(m\ mod\ d = 0 \\
\quad \wedge\ n\ mod\ d = 0) \\
\quad \Leftrightarrow (a\ mod\ d = 0 \\
\quad \wedge\ b\ mod\ d = 0)) \\
\Rightarrow \\
(\forall d.(m\ mod\ d = 0 \wedge \\
\quad (n - m)\ mod\ d = 0) \\
\quad \Leftrightarrow (a\ mod\ d = 0 \\
\quad \wedge\ b\ mod\ d = 0))
\end{array}
\quad
\dfrac{}{
\begin{array}{l}
\{(\forall d.(m\ mod\ d = 0 \wedge \\
\quad (n - m)\ mod\ d = 0) \\
\quad \Leftrightarrow (a\ mod\ d = 0 \\
\quad \wedge\ b\ mod\ d = 0))\} \\
n := n - m \\
\{(\forall d.(m\ mod\ d = 0 \\
\quad \wedge\ n\ mod\ d = 0) \\
\quad \Leftrightarrow (a\ mod\ d = 0 \\
\quad \wedge\ b\ mod\ d = 0))\}
\end{array}
}\text{AssignAx}
}{
\begin{array}{l}
\{(\forall d.(m\ mod\ d = 0 \wedge n\ mod\ d = 0) \\
\quad \Leftrightarrow (a\ mod\ d = 0 \wedge b\ mod\ d = 0)) \\
\wedge\ (n \neq m) \wedge (m < n)\} \\
n := n - m \\
\{(\forall d.(m\ mod\ d = 0 \wedge n\ mod\ d = 0) \\
\quad \Leftrightarrow (a\ mod\ d = 0 \wedge b\ mod\ d = 0))\}
\end{array}
}\text{PrecondStr}
\quad
\dfrac{
\begin{array}{l}
(2) \\
(\forall d.(m\ mod\ d = 0 \\
\quad \wedge\ n\ mod\ d = 0) \\
\quad \Leftrightarrow (a\ mod\ d = 0 \\
\quad \wedge\ b\ mod\ d = 0)) \\
\Rightarrow \\
(\forall d. \\
\quad ((m - n)\ mod\ d = 0 \\
\quad \wedge\ n\ mod\ d = 0) \\
\quad \Leftrightarrow (a\ mod\ d = 0 \\
\quad \wedge\ b\ mod\ d = 0))
\end{array}
\quad
\dfrac{}{
\begin{array}{l}
\{(\forall d. \\
\quad ((m - n)\ mod\ d = 0 \\
\quad \wedge\ n\ mod\ d = 0) \\
\quad \Leftrightarrow (a\ mod\ d = 0 \\
\quad \wedge\ b\ mod\ d = 0))\} \\
m := m - n \\
\{(\forall d.(m\ mod\ d = 0 \\
\quad \wedge\ n\ mod\ d = 0) \\
\quad \Leftrightarrow (a\ mod\ d = 0 \\
\quad \wedge\ b\ mod\ d = 0))\}
\end{array}
}\text{AssignAx}
}{
\begin{array}{l}
\{(\forall d.(m\ mod\ d = 0 \wedge n\ mod\ d = 0) \\
\quad \Leftrightarrow (a\ mod\ d = 0 \wedge b\ mod\ d = 0)) \\
\wedge\ (n \neq m) \wedge \neg(m < n)\} \\
m := m - n \\
\{(\forall d.(m\ mod\ d = 0 \wedge n\ mod\ d = 0) \\
\quad \Leftrightarrow (a\ mod\ d = 0 \wedge b\ mod\ d = 0))\}
\end{array}
}\text{PrecondStr}
}{
\begin{array}{l}
\{(\forall d.(m\ mod\ d = 0 \wedge n\ mod\ d = 0) \Leftrightarrow (a\ mod\ d = 0 \wedge b\ mod\ d = 0)) \\
\wedge\ (n \neq m)\} \\
if\ m < nthen\ n := n - m \\
else\ m := m - n\ fi)\ od \\
\{(\forall d.(m\ mod\ d = 0 \wedge n\ mod\ d = 0) \Leftrightarrow (a\ mod\ d = 0 \wedge b\ mod\ d = 0))\}
\end{array}
}\text{IfThenElseRule}
$$

where (1) and (2) are true because $(m\ mod\ d = 0 \wedge n\ mod\ d = 0)$ holds if and only if $(m\ mod\ d = 0 \wedge (n - m)\ mod\ d = 0)$ holds if and only if $((m - n)\ mod\ d = 0 \wedge n\ mod\ d = 0)$.

Then

$$
\dfrac{
\dfrac{
\begin{array}{l}
(3)\\
(a > 0 \land b > 0) \Rightarrow\\
(\forall d.(a\ mod\ d = 0\\
\quad \land b\ mod\ d = 0)\\
\quad \Leftrightarrow (a\ mod\ d = 0\\
\quad \land b\ mod\ d = 0))
\end{array}
\quad
\dfrac{}{
\begin{array}{l}
\{(\forall d.(a\ mod\ d = 0\\
\quad \land b\ mod\ d = 0)\\
\quad \Leftrightarrow (a\ mod\ d = 0\\
\quad \land b\ mod\ d = 0))\}\\
m := a;\\
\{(\forall d.(m\ mod\ d = 0\\
\quad \land b\ mod\ d = 0)\\
\quad \Leftrightarrow (a\ mod\ d = 0\\
\quad \land b\ mod\ d = 0))\}
\end{array}
}\ \textsf{AssignAx}
}{
\begin{array}{l}
\{a > 0 \land b > 0\}\\
m := a;\\
\{(\forall d.(m\ mod\ d = 0 \land b\ mod\ d = 0)\\
\quad \Leftrightarrow (a\ mod\ d = 0 \land b\ mod\ d = 0))\}
\end{array}
}\ \textsf{PrecondStr}
\quad
\dfrac{}{
\begin{array}{l}
\{(\forall d.(m\ mod\ d = 0\\
\quad \land b\ mod\ d = 0)\\
\quad \Leftrightarrow (a\ mod\ d = 0\\
\quad \land b\ mod\ d = 0))\}\\
n := b;\\
\{(\forall d.(m\ mod\ d = 0\\
\quad \land n\ mod\ d = 0)\\
\quad \Leftrightarrow (a\ mod\ d = 0\\
\quad \land b\ mod\ d = 0))\}
\end{array}
}\ \textsf{AssignAx}
}{
\begin{array}{l}
\{a > 0 \land b > 0\}\\
m := a;\ n := b;\\
\{(\forall d.(m\ mod\ d = 0 \land n\ mod\ d = 0)\\
\quad \Leftrightarrow (a\ mod\ d = 0 \land b\ mod\ d = 0))\}
\end{array}
}\ \textsf{SeqRule}
$$

$$
\dfrac{
\dfrac{
\vdots
}{
\begin{array}{l}
\{a > 0 \land b > 0\}\\
m := a;\ n := b;\\
\{(\forall d.(m\ mod\ d = 0 \land n\ mod\ d = 0)\\
\quad \Leftrightarrow (a\ mod\ d = 0 \land b\ mod\ d = 0))\}
\end{array}
}\ \textsf{SeqRule}
\quad
\dfrac{IfThenElseProof}{
\begin{array}{l}
\{(\forall d.(m\ mod\ d = 0\\
\quad \land n\ mod\ d = 0)\\
\quad \Leftrightarrow (a\ mod\ d = 0\\
\quad \land b\ mod\ d = 0))\}\\
while\ n\ \neq\ m\ do\\
(if\ m < n\\
then\ n := n - m\\
else\ m := m - n\ fi)\ od\\
\{(\forall d.(m\ mod\ d = 0\\
\quad \land n\ mod\ d = 0)\\
\quad \Leftrightarrow (a\ mod\ d = 0\\
\quad \land b\ mod\ d = 0))\\
\land \neg(n \neq m)\}
\end{array}
}\ \textsf{WhileRule}
}{
\begin{array}{l}
\{a > 0 \land b > 0\}\\
m := a;\ n := b;\\
while\ n\ \neq\ m\ do\\
(if\ m < n\ then\ n := n - m\\
else\ m := m - n\ fi)\ od\\
\{(\forall d.(m\ mod\ d = 0 \land n\ mod\ d = 0)\\
\quad \Leftrightarrow (a\ mod\ d = 0 \land b\ mod\ d = 0))\\
\land \neg(n \neq m)\}
\end{array}
}\ \textsf{SeqRule}
$$

$$
\dfrac{
\begin{array}{l}
\{a > 0 \land b > 0\}\\
m := a;\ n := b;\\
while\ n\ \neq\ m\ do\\
(if\ m < n\ then\ n := n - m\\
else\ m := m - n\ fi)\ od\\
\{(\forall d.(m\ mod\ d = 0 \land n\ mod\ d = 0)\\
\quad \Leftrightarrow (a\ mod\ d = 0 \land b\ mod\ d = 0))\\
\land \neg(n \neq m)\}
\end{array}
\quad
\begin{array}{l}
(4)\\
(\forall d.(m\ mod\ d = 0 \land n\ mod\ d = 0)\\
\quad \Leftrightarrow (a\ mod\ d = 0 \land b\ mod\ d = 0))\\
\land \neg(n \neq m) \Rightarrow\\
(a\ mod\ m = 0 \land b\ mod\ m = 0)
\end{array}
}{
\begin{array}{l}
\{a > 0 \land b > 0\}\\
m := a;\ n := b;\ while\ n\ \neq\ m\ do\ (if\ m < n\ then\ n := n - m\ else\ m := m - n\ fi)\ od\\
\{a\ mod\ m = 0 \land b\ mod\ m = 0\}
\end{array}
}\ \textsf{PostcondWk}
$$

Condition (3) holds trivially. For condition (4), first $\neg(n \neq m) \Rightarrow (n = m)$. Thus, if we specialize the antecednet to m, it reduces to $(m\ mod\ m = 0 \land m\ mod\ m = 0) \Leftrightarrow (a\ mod\ m = 0 \land b\ mod\ m = 0)$. But we always have $m\ mod\ m = 0$. Therefore, we have $(a\ mod\ m = 0 \land b\ mod\ m = 0)$.

6 Hoare Logic Proofs in Isabelle/HOL

In the directory `assignments/hw4` where the pdf for this file is located, there is a file `Hoare_SIMP.thy` where there is Hoare Logic for a simple imperative programming language. The theory file `Hoare_SIMP.thy` contains definitions for embedding predicate logic as suited to Hoare Logic into Isabelle/HOL. The type `'data` allows us to give what we want as our basic form of values for our programming language, in this instance `int`. Our expressions are represented as functions from states to `data`. We encapsulated this with the type abbreviation:

```
type_synonym exp = "state ⇒ data"
```

Similarly, boolean expressions are represented as functions from states to `bool`, and encapsulated with the type abbreviation:

```
type_synonym bool_exp = "state ⇒ bool"
```

Variables are represented by strings (which in turn are encoded as lists of characters) using the abbreviation `var_name`. We can use `$::var_name ⇒ exp` to convert variables into expressions. We can use `k::int⇒ exp` to convert integers and reals into expressions. Likewise, we can use `Bool::bool⇒ bool_exp` to convert booleans into boolean expressions. The first order predicates and logical connectives have been "lifted" to take arguments taking a state as an argument, and returning results also parametrized by states. An example of this is:

```
definition plus_e :: "exp ⇒ exp ⇒ exp" (infixl "[+]" 150) where
"(p [+] q) ≡ (λ s. p s + q s)"
```

The theorems stating the definitions (those introduced by the keyword `definition` are named by adding `_def` to the basic name of the defined constant. For example, the above definition may be accessed through the name `plus_e_def`.

Using this essentially shallow embedding of predicate logic in Isabelle/HOL, we can define substitution of expressions for variables in any construct encoded as a mapping from state by:

```
definition substitute :: "(state ⇒ 'a) ⇒ var_name ⇒ exp ⇒ (state ⇒ 'a)"
      ("_/[_/⇐_/]" [120,120,120]60)
 where "p[x⇐ e] ≡ λ s. p(λ v. if v = x then e(s) else s(v))"
```

The programming language itself is encoded using a data type for its abstract syntax trees, and augmenting the constructs with mixfix notation so that terms can be typed in and pretty-printed back roughly in the expected concrete syntax. The data type is as follows:

```
datatype command =
   AssignCom "var_name" "exp"              (infix "::=" 110)
 | SeqCom "command" "command"              (infixl ";" 109)
 | CondCom "bool_exp" "command" "command"
      ("IF _/ THEN _/ ELSE _/ FI" [120,120,120]60)
 | WhileCom "bool_exp" "command"           ("WHILE _/ DO _/ OD" [120,120]60)
```

As an example use of all this, if we wanted to express

```
if x > 0 then y := 2 + x else y := 2 - x
```

we could enter into Isabelle/HOL

```
term "IF $''x'' [>] k 0 THEN ''y'' ::= k 2 [+] $''x''
                        ELSE ''y'' ::= k 2 [-] $''x'' FI"
```

The rules for the logic are given by the following rules (for the inductive relation `hvalid`):

`AssignmentAxiom:` $\{\{P[x \Leftarrow e])\}\}\ x ::= e\ \{\{P\}\}$

`SequenceRule:` $[\![\{\{P\}\}\ C\ \{\{Q\}\}\ ;\ \{\{Q\}\}\ C'\ \{\{R\}\}]\!]$
$\Longrightarrow \{\{P\}\}\ C\ ;\ C'\ \{\{R\}\}$

`RuleOfConsequence:` $[\![\ |\!\models (P[\longrightarrow]P')\ ;\ \{\{P'\}\}\ C\ \{\{Q'\}\}\ ;\ \ |\!\models (Q'[\longrightarrow]QP]\!] \Longrightarrow$
$\{\{P\}\}\ C\ \{\{Q\}\}$

`IfThenElseRule:` $[\![\{\{P[\wedge]B\}\}\ C\ \{\{Q\}\}\ ;\ \{\{P[\wedge]\ [\neg]B\}\}\ C'\ \{\{Q\}\}]\!] \Longrightarrow$
$\{\{P\}\}$ `IF` B `THEN` C `ELSE` C' `FI` $\{\{Q\}\}$

`WhileRule:` $[\![\{\{P[\wedge]B\}\}\ C\ \{\{P\}\}]\!] \Longrightarrow$
$\{\{P\}\}$ `WHILE` B `DO` C `OD` $\{\{P[\wedge]\ [\neg]B\}\}$

From the `RuleOfConsequence` there are two derived rules:

`PreconditionStrengthening:` $[\![\ |\!\models (P[\longrightarrow]P')\ ;\ \{\{P'\}\}\ C\ \{\{Q\}\}]\!] \Longrightarrow$
$\{\{P\}\}\ C\ \{\{Q\}\}$

`PostconditionWeakening:` $[\![\ |\!\models (Q'[\longrightarrow]Q)\ ;\ \{\{P\}\}\ C\ \{\{Q'\}\}]\!] \Longrightarrow$
$\{\{P\}\}\ C\ \{\{Q\}\}$

In addtion to using `rule`, `rule_tac`, `erule`, and `erule_tac`, particularly with the above rules, you will want to use `simp add:` $def_1 \ldots def_n$ and `clarsimp simp add:` $def_1 \ldots def_n$ to expand out the definitions and apply previously proven simplifications. If you wish to give a intermediate result that you feel will help, for example from the given hypotheses, you wish to both use and show a result, you may use `subgoal_tac` *result*. You may also want to use the built-in tool Sledgehammer to have Isabelle suggest possible proofs to you (using `metis`).

7 Isabelle Problem

4. (20pts) Prove in Isabelle/HOL

```
{{$''n'' [>] k 0}}
''i'' ::= $''n'' ; ''j'' ::= k 0 ;
WHILE $''i'' [>] k 0 DO ''j'' ::= $''j'' [+] $''i'' ; ''i'' ::= $''i'' [−] k 1 OD
{{k 2 [×] $''j'' [=] $''n'' [×] $''n'' [+] k 1}}
```

5. (Extra Credit 5 pts) Prove in Isabelle/HOL

```
{{$''a'' [>] k0 [∧] $''b'' [>] k0}}
''m'' ::= $''a'' ; ''n'' ::= $''b'' ;
(WHILE ([¬]($''n'' [=] $''m''))
 DO IF $''m'' [<] $''n'' THEN ''n'' ::= $''n'' [−] $''m''
   ELSE ''m'' ::= $''m'' [−] $''n'' FI
 OD)
{{$''a'' [mod] $''m'' [=] k0 [∧] $''b'' [mod] $''m'' [=] k0}}"
```