**Computer Science 81, Spring 2011**
Assignment 6
Due Wed. Mar. 2

1. Prove the total correctness of the program below. The program is given as a Hoare triple in the "Japeish" language. **All variables are integers**. ***You are not required to use JAPE***, although you are free to do so. (It will be instructive, but not trivial.)

   Here ÷ is the integer division function (quotient is always an integer), *mod* is the modulus or remainder function, that is, mod(n, d) is the remainder of dividing n by d, and *pow* is the power function, that is pow(b, n0) = $b^{n0}$.

```
{n = n0 ∧ n0 ≥ 0 ∧ b > 0}

(s := b; r := 1)
while n > 0
      do
      if mod(n, 2) = 1
            then r := r × s
            else skip
      fi;
      n := n ÷ 2;
      s := s × s
      od

{r = pow(b, n0)}
```

The first task is to determine how this program works. Devising an appropriate loop invariant should help. Note that the invariant will usually involve *all* of the variables used in the loop. To prove termination, you will need to establish a variant as well.

In case you use JAPE, below is the incantation that should be entered as a conjecture, except you will have to insert your invariant and this will all be on one line:

```
WHERE DISTINCT b, n, n0, pow, r, s IS
{n=n0 ∧ n0 ≥ 0 ∧ b>0} (s:=b; r:=1)
{… your invariant here …}
while n > 0 do if mod(n,2)=1 then r:=r×s else skip fi; n:=n÷2; s := s×s od
{r = pow(b, n0)}
```

Note that you would start the proof using JAPE's Ntuple rule since the invariant is present as an intermediate assertion.

Any facts that you use that are not built into JAPE's Hoare logic should be expressed using Lemmas (the proofs of which can use the *Obviously* rule). List all lemmas that you used. Here are most of the lemmas I used, and you are welcome to use these. You will need to enter them in the Useful Lemmas window, and provide "proofs", most of which

you can declare as obvious for now. Generally, your proof will be more elegant with fewer lemmas.

```
(A, B, C, X are arbitrary integer variables)

 a.  pow(A, B) defined
 b.  mod(n,2) computes
 c.  2 ≠ 0
 d.  A×B×C = A×(B×C)
 e.                A = X  |— 1×A = X
 f.                B = 1  |— A×B = A
 f.                B = C  |— A×B = A×C
 h.         A = B, B = C  |— A = C
 i.      A ≥ 0, ¬(A > 0)  |— A = 0
 j.                n ≥ 0  |— n ÷ 2 ≥ 0
 k.         n = A, n > 0  |— n ÷ 2 < A
 l.                n = 0  |— pow(A, n) = 1
 m.      ¬(mod(n,2) = 1)  |— mod(n, 2) = 0
 n.         mod(n,2) = 0  |— pow(s × s, n ÷ 2) = pow(s, n)
 o.         mod(n,2) = 1  |— s × pow(s × s, n ÷ 2) = pow(s, n)
```

If using JAPE, *save your work often*, especially your lemmas. (If you get an error dialog from JAPE asking you whether to Continue or Quit, it is better to Continue, because if you quit, you may lose your current proof.)

To reload your work into JAPE, you must load the Theory first, then open your file.

**Submit**: Your proof on paper, along with a commentary showing what the various parts are for, and how they inter-relate. Don't simply turn a print out unannotated. The annotations could be something along the lines of slides presented in class. For example, you could use the JAPE line numbers and have your commentary refer to ranges of them. Or you could embed your assertions into code, using what Huth and Ryan call the "tableau" representation.

Note that you might have to repeat the proof one or more times, so you can see how to construct the commentary. It should go faster the second time. You might want to make notes along the way, because the rationale and order is not always evident in the final proof.

For reference, my proof was about 87 lines of JAPE, not counting the proofs of lemmas. If you need more hints, please ask.

2. The following program, also in Japeish, is designed to compute the index `m` of the minimum value in a non-empty array. (If there is more than one value having the minimum, then this program computes the last one, but this is not part of the stated expectation.) All variables are integer. Note that the array is not modified during the program, so it can be treated as a constant.

```
WHERE DISTINCT a, i, m IS

{0 < length(a) }
( m := 0; i := 1 )

{ … your invariant here … }

while i < length(a)
   do
      if a[i] ≤ a[m]
            then m := i
            else skip
      fi;
      i := i+1
   od

{∀j.(((0 ≤ j) ∧ (j < length(a))) → a[m] ≤ a[j])}
```

1. Devise an invariant that will be sufficiently strong to enable a total correctness proof of the program using Hoare logic.

2. Prove termination using an appropriate variant.

3. **You do not need to use JAPE**, although it is acceptable to do so. The program given is ready to enter once you've added the invariant.

4. State any lemmas that you use.

5. I found it helpful at one point to bifurcate and use ∨-Elimination to deal with the verification conditions related to the conditional. The specific lemma I used to create the bifurcation was:

   $A < B+1 \vdash A < B \vee A = B$