Programming Language Qualifying Exam
Spring 2011

Answer the following five problems.

1. Languages and Compilation

   (a) Data that is passed to a function is called *boxed* if a reference is passed instead of a copy of the data. Give some advantages and disadvantages of boxing.

   (b) The language Fortran hard-coded the locations of all variables—including function parameters—during the compilation. What benefit would this bring? What would the disadvantages of such an approach be?

   (c) What is the difference between a compiler, and interpreter, and a byte-code compiler? Give an example of each.

   (d) What are the differences between the object-oriented, functional, and imperative languages? Give an example of each, and an advantage each language type has over the others.

2. Abstraction

   (a) What are abstract data-types? Why are they important?

   (b) One of the major changes from C++ to Java was the replacement of pointers with references. Give at two advantages of having made this change.

   (c) Give a situation in which using C-style pointers rather than Java-style references is preferable from a software engineering standpoint.

3. Grammars

   Consider the following grammar:

$$\begin{array}{lcl} S & \rightarrow & y\ E \\ E & \rightarrow & E\ y\ E \\ & | & x\ b \end{array}$$

   (a) Construct the Characteristic Finite State Machine for the above grammar.

   (b) Convert the above grammar to an LL grammar (or explain why it is already LL).

   (c) What advantage results from a grammar being LL?

   (d) Is the above grammar ambiguous? Give a proof with your answer.

4. Weakest Precondition

   (a) Define *weakest precondition* and *weakest liberal precondition*.

   (b) In English, explain what $WP(S, F) = F$ indicates. (Note, we say explain, not simply translate.)

   (c) Consider the following program S. Let the postcondition $R \equiv x = y$. Determine formally the conditions under which this program returns the correct answer.

```
if x > y then x := 2 * y;
if x < y then y := 2 * x;
```

5. Loop Verification

   (a) In order to verify the correct operation of a loop, you need to check five formulas. What are they?

   (b) Fix the bug in the following program (if there is one), and formally prove the result. The postcondition is $s = max_{i=0}^{n-1} a[i]$. I.e., we determine the maximum element of the array. You will need to determine the loop invariant.

```
s := 0;
i := 0;
do i < n -> s,i := max(s, a[i]), i + 1
od
```

   (c) Writing proofs can be a lot of work. Why not just use testing instead of formal methods to prove programs correct?