Programming Language Qualifying Exam
Fall 2011

Answer the following five problems.

1. Languages and Compilation

   (a) In the near future, the Java specification committee wants to add *closures* to the Java programming language. What are closures, and why might they be useful?

   (b) Many compilers compile to an intermediate representation, such as LLVM, instead of compiling directly to machine code. What advantage is there to building compilers this way?

   (c) One hallmark of functional programming languages is that variables, once assigned, can never be re-assigned. Now that parallelism has become more prevalent, there is much more interest in this feature. Why?

   (d) What is the von Neumann model of computation? How does it differ from lambda calculus?

   (e) Consider the following C code:

```
int i;
int a[100];
int b[100];
for(i=0; i<=100; i++) {
   a[i] = 5;
   b[i] = 20;
}
printf("%d\n",b[0]);
```

   Much to the surprise of the programmer, this code output the number 5, rather than the expected number 20. Why did that happen?

2. Abstraction

   (a) What is an abstract data-type?

   (b) A queue is often implemented by encapsulating a list, remapping enqueue and dequeue to list operations, and not providing the remaining list operations. In short, a queue provides strictly *less* functionality than a list. What would be wrong with simply providing a list with the enqueue and dequeue methods added?

   (c) In C, you can add pointers and subtract them, almost as if they were integers. In Java, you can only check if they are equal to each other. (Of course, both languages allow you to dereference pointers.) Compare these two approaches, and explain for both of these the circumstances that would make them useful.

3. Grammars

Consider the following grammar:

$$\begin{array}{lcl} S & \rightarrow & y\ E \\ & \rightarrow & y\ z \\ E & \rightarrow & E\ y\ E \\ & | & a \end{array}$$

(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 $WP(S, T) = F$ 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.

```
x := x * y;
y := y * x;
if x > y then x := x - 1
         else y := y - 1
fi
```

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 $\forall i.0 \leq i < |A|.a[i] < b[i] \wedge (a[i] = A[i] \vee a[i] = B[i]) \wedge (b[i] = A[i] \vee b[i] = B[i])$. Here, A and B represent the initial values of the arrays a and b.

   You will need to determine the loop invariant.

```
i := 0;
do i < |A| ->
   if a[i]>b[i] then a[i],b[i] := b[i],a[i]
                else skip
   fi
od
```