CSCE 330 Fall 2007
MIDTERM EXAM
Thursday 2007-09-27—Three Pages, Closed Book

1 Short Questions–1 or 2 points each; 10 points total

1. (1 points) Match: (a) Niklaus Wirth and (b) John McCarthy with (1) LISP and (2) Pascal, and with (i) 1971 and (ii) 1960.
2. (1 point) What is the name of the family of programming languages whose structure is dictated by the von Neumann computer architecture?
3. (2 points) Following up on the previous question, what are the names of the two other main families of programming languages?
4. (1 point) Match: (a) Verification and (b) Validation with (1) "Are we building the program right?" and (2) "Are we building the right product?"
5. (1 point) What does the acronym RUDE stand for?
6. (1 point) The spiral method of software development is characterized by prototyping. True or false?
7. (1 point) Match one of a–b with one of c–d and with one of e–f:
   (a) Syntax
   (b) Semantics
   (c) Meaning
   (d) Form
   (e) How something is expressed
   (f) What something does
8. (1 point) Syntax diagrams are a notational variant of EBNF. True or false?
9. (1 point) The following sentence is syntactically correct: "Time flies like green bananas." True or false?

2 Syntax–15 points

1. (2 points) What does it mean for a (context-free) grammar to be *ambiguous*?

2. (8 points) The grammar of the original definition of Algol 60 contained the following production rules:
   `<statement> ::= <conditional-statement> | begin <statement> end`
   `<conditional-statement> ::= if <condition> then <statement>`
   `| if <condition> then <statement> else <statement>`
   Show that any grammar containing these production rules is ambiguous.

3. (5 points) The grammar of Java avoids the ambiguity described in the exercise above by introducing the nonterminal `<statement-no-short-if>`, which includes all statements except a conditional statement without the else branch. In addition to rules defining `<statement-no-short-if>` as just described, here are the relevant production rules:
   `<if-then-statement> ::= if (<condition>) then <statement>`
   `<if-then-else-statement> ::= if (<condition>) then <statement-no-short-if>`
   `else <statement>`
   `<statement> ::= <if-then-else-statement>`
   Draw the parse tree for the sentential form
   `if (<condition>) then if (<condition>) then <statement-no-short-if>`
   `else <statement>` using the Java grammar, starting with `<if-then-statement>`.

3 Scope and Type Rules–5 points

Consider the (pseudo-Pascal) program below.

```
program MAIN;
 var X: integer;
 procedure A;
  begin
   write(X)
  end; {of procedure A}
 procedure B;
  var X: integer;
  begin
   X := 6;
   call A
  end; {of procedure B}
 begin {of MAIN}
  X := 15;
  call B
 end. {of program MAIN}
```

1. Under static scoping rules, what value of X is printed in procedure A?

2. Under dynamic scoping rules, what value of X is printed in procedure A?

4 Semantics–17 points

1. (5 points) Describe (very briefly) the semantic difference between commands and expressions. (Hint: Use the assignment statement, which is a command, to illustrate the difference.)

2. (2 points) Give the weakest precondition for the assignment statement `foo := x + y;` with postcondition `foo > 0`

3. (5 points) To show that I is a loop invariant for a loop whose condition is B, you need to prove three of the following properties. Which three?

   (a) I implies the precondition of the loop

   (b) I is implied by the precondition of the loop

   (c) $I \wedge \neg B$ implies the postcondition of the loop

   (d) If $I \wedge \neg B$ holds before at the beginning of the loop, I holds after executing the body of the loop.

   (e) If $I \wedge B$ holds before at the beginning of the loop, I holds after executing the body of the loop.

4. (5 points) Give a loop invariant for this loop:

```
while (i <> n) do
  begin
    prod := prod * i;
    i := i+1
  end
```

   with precondition $i = 1 \wedge prod = 1 \wedge n \geq 1$ and postcondition $prod = \Pi_{j=i}^{n-1} j$.