Stanford University
Computer Science Department

Fall 2004 Comprehensive Exam in Compilers

1. **Closed Book, No Laptop & Notes. Write only in the Blue Book provided.**
2. **The exam is timed for one hour.**
3. **Write your Magic Number on this sheet & on the Blue Book.**

The following is a statement of the Stanford University Honor Code:

A. The Honor Code is an undertaking of the students, individually and collectively:

1. that they will not give or receive aid in examinations; that they will not give or receive un-permitted aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading;

2. that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the Honor Code.

B. The faculty on its part manifests its confidence in the honor of its students by refraining from proctoring examinations and from taking unusual and unreasonable precautions to prevent the forms of dishonesty mentioned above. The faculty will also avoid, as far as practicable, academic procedures that create temptations to violate the Honor Code.

C. While the faculty alone has the right and obligation to set academic requirements, the students and faculty will work together to establish optimal conditions for honorable academic work.

By writing my Magic Number below, I certify that I acknowledge and accept the Honor Code.

***Magic Number*----------**

Compilers Comprehensive Exam
Fall 2004

This is a 60 minute, closed book exam. Please mark your answers in the blue book.

1. **Regular Expressions** (5 points)

   Consider a language where real constants are defined as follows: A real constant contains a decimal point or E notation, or both. For instance, 0.01, 2.71821828, ¯1.2E12, and 7E¯5 are real constants. The symbol "¯" denotes unary minus and may appear before the number or on the exponent. There is *no* unary "+" operation. There must be at least one digit to left of the decimal point, but there might be no digits to the right of the decimal point. The exponent following the "E" is a (possibly negative) integer.

   Write a regular expression for such real constants. Use the standard regular expression notation described by the following grammar:

   $$R \rightarrow \epsilon \,|\, \text{char} \,|\, R + R \,|\, R* \,|\, RR \,|\, (R)$$

   You may define names for regular expressions you want to use in more than one place (e.g., $foo = R$).

2. **Grammars** (20 points)

Consider the following grammar. The nonterminals are E, T, and L. The terminals are +,**id**,(,), and ;. The start symbol is E.

$$\begin{aligned} \mathrm{E} &\rightarrow \mathrm{E} + \mathrm{T} \mid \mathrm{T} \\ \mathrm{T} &\rightarrow \mathbf{id} \mid \mathbf{id}() \mid \mathbf{id}(\mathrm{L}) \\ \mathrm{L} &\rightarrow \mathrm{E}; \mathrm{L} \mid \mathrm{E} \end{aligned}$$

Give an LL(1) grammar that generates the same language as this grammar. For full credit, you must show (convincingly) that your grammar is LL(1).

3. **Parsing** (15 points)

Consider the following grammar. The nonterminals are S′ and S. The terminals are **op** and **x**. The start symbol is S′.

$$\begin{aligned} \mathrm{S}' &\rightarrow \mathrm{S} \\ \mathrm{S} &\rightarrow \mathrm{S}\ \mathbf{op}\ \mathrm{S} \mid \mathbf{x} \end{aligned}$$

(a) Draw the DFA built from sets of LR(0) items for this grammar. Show the contents of each state. (Note: Don't augment the grammar with a new start symbol.)

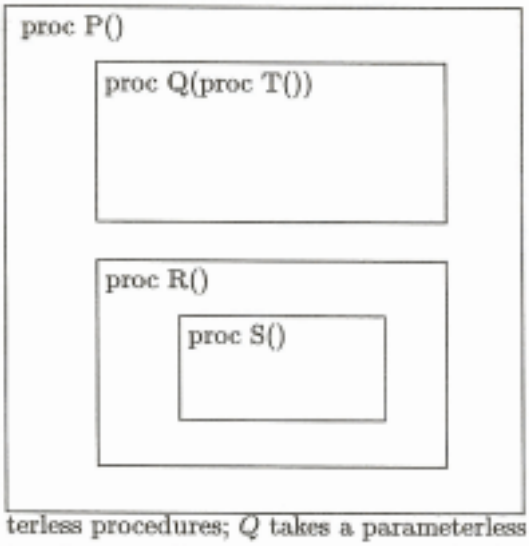
(b) Is this grammar LR(1)? Briefly explain why or why not.

4. **Scope** (10 points)

   Give a simple program that produces different results if executed using lexical scoping than if executed using dynamic scoping. Show what your program produces in both cases. Use any reasonable and clear programming notation.

5. **Activation Records** (10 points)

Consider a program with the following lexical structure. The program is written in a lexically scoped language with nested procedures (like Pascal):



P, R, and S are parameterless procedures; Q takes a parameterless procedure T as a parameter. Suppose that at run-time the following sequence of calls is made:

P is called from some lexically-enclosing main program
P calls R
R calls S
S calls P
P calls R
R calls Q with S as a parameter
Q calls T
T calls S

Draw the stack of activation records present after this sequence of calls. You don't need to show the entire contents of the activation record—for each indicate only the name of the procedure being activated, the control (dynamic) link for that activation record, and the access (static) link for that activation record.