COP5621 Compiler Construction Exam 3 - Spring 2007

Name: ________________________________ (Please print)

Put the answers on these sheets. Use additional sheets when necessary. You can collect 100 points in total for this exam.

1. What is the name of the evaluation method where the evaluation order of semantic rules is determined by a topological sort of the dependence graph obtained from the parse tree? (mark **one**) (4 points)

   (a) Rule-based method

   (b) Oblivious method

   (c) Parse-tree method

   (d) Sorting method

2. Which attributes of S in the following grammar are synthesized or inherited? (mark **one**) (4 points)

   $$\begin{array}{rcll} S_1 & \rightarrow & \mathbf{a}\, S_2 & S_1.\text{x} := S_2.\text{x};\ S_2.\text{y} := S_1.\text{y} + \text{'a'} \\ S_1 & \rightarrow & \mathbf{b}\, S_2 & S_1.\text{x} := S_2.\text{x};\ S_2.\text{y} := S_1.\text{y} + \text{'b'} \\ S & \rightarrow & \varepsilon & S.\text{x} := S.\text{y} \end{array}$$

   (a) x of S and y of S are synthesized

   (b) x of S is synthesized and y of S is inherited

   (c) x of S is inherited and y of S is synthesized

   (d) x of S and y of S are inherited

3. What is the difference between *name equivalence* and *structural equivalence* of types? (8 points)

4. Consider the following C type declaration of a binary tree node:

```
struct IntBinTree
{ int val;
  struct IntBinTree *left;
  struct IntBinTree *right;
} tree;
```

Draw the cyclic type graph for `tree` (as would be constructed by a compiler). (8 points)

5. Consider the following post system rules (8 points):

$$\frac{\rho(v) = \tau}{\rho \vdash v : \tau} \qquad \frac{\rho \vdash e_1 : \text{bool} \quad \rho \vdash e_2 : \text{bool}}{\rho \vdash e_1 \,\textbf{and}\, e_2 : \text{bool}} \qquad \frac{\rho \vdash e_1 : \text{bool} \quad \rho \vdash e_2 : \text{bool}}{\rho \vdash e_1 \,\textbf{or}\, e_2 : \text{bool}}$$

Given $\rho = \{\langle \mathbf{a}, \text{bool}\rangle, \langle \mathbf{b}, \text{bool}\rangle, \langle \mathbf{c}, \text{bool}\rangle\}$, complete the proof of $\rho \vdash \mathbf{a}\,\mathbf{or}\,\mathbf{b}\,\mathbf{and}\,\mathbf{c} : \text{bool}$

$$\frac{\rho \vdash \mathbf{a} : \text{bool} \qquad \rho \vdash \mathbf{b}\,\mathbf{and}\,\mathbf{c} : \text{bool}}{\rho \vdash \mathbf{a}\,\mathbf{or}\,\mathbf{b}\,\mathbf{and}\,\mathbf{c} : \text{bool}}$$

6. Consider the following syntax-directed translation scheme to collect variable names in a list:

$$\begin{array}{lcl} L_1 & \rightarrow & L_2 \textbf{ , id } \{ \; L_1.\text{list} := \textit{merge}(L_2.\text{list}, \textit{mklist}(\textbf{id}.\text{name})) \; \} \\ L & \rightarrow & \textbf{id } \{ \; L.\text{list} := \textit{mklist}(\textbf{id}.\text{name}) \; \} \end{array}$$

(a) Apply left recursion elimination to produce a transformed translation scheme. (10 points)

(b) Draw the annotated parse tree for the input **f, s, u** using the modified translation scheme. (8 points)

7. Given the following incomplete syntax-directed translation scheme

$S \rightarrow$ **repeat** M S **until** C **;** { *???* }
$S \rightarrow$ **id :=** E **;** { *emit*(**id**.place ':=' E.place) }
$E \rightarrow$ E_1 **+** E_2 { E.place := *newtemp*(); *emit*(E.place ':=' E_1.place '+' E_2.place) }
$E \rightarrow$ **id** { E.place := **id**.place }
$E \rightarrow$ **num** { E.place := *newtemp*(); *emit*(E.place ':=' **num**) }
$C \rightarrow$ **id rel id** { C.truelist := *makelist*(*nextquad*()); *emit*(if **id**$_1$.place **rel id**$_2$.place goto _);
C.falselist := *makelist*(*nextquad*()); *emit*(goto _) }
$M \rightarrow$ ε { M.q = *nextquad*() }

(a) Give the actions marked "*???*" to complete the repeat-until loop translation using *backpatch* operations. (10 points)

(b) Given input **repeat k:=k+1; until k**$>$**j;** show the resulting output, assuming that **rel** is a relational operator such as '$>$' and **id**.place is the name of the identifier. (10 points).

8. Describe the use of symbol tables in the code generation process. That is, describe how symbol tables are used to determine scope of variables and how this affects the code that performs the run-time access to load/store values in subroutine frames or in static data space. (10 points)

9. What is the difference between the *triples* and *indirect triples* intermediate code representation? Give an example to compare the two representations and state the advantage(s) of one over the other, if any. (10 points)

10. Complete the following Yacc specification that constructs ASTs for arithmetic expressions denoted by E (using pre-defined `ASTnode *mknode(int op, ASTnode *left, ASTnode *right)` and `ASTnode *mkleaf(Symbol *sym)` functions). (10 points)

```
%{
typedef struct ASTnode
{  int op;        /* node op */
   Symbol *entry; /* leaf */
   struct ASTnode *left, *right;
} ASTnode;
%}

%union
{   ASTnode *ast;
    Symbol *sym;
}

%token <sym> ID
%type <ast> E
%left '+'
%left '*'

%%

E : E '+' E



  | E '*' E



  | ID



  ;
```