CS 326 – Programming Languages, Concepts and Implementation

Homework 1

Due September 10

1. (10 pts) Why are the front end and the back end of a compiler usually implemented as separate passes?

2. (28 pts) For each of the following languages, write a regular expression that describes the language.

   (a) (7 pts) The set of strings of length three or more, over alphabet {*a*, *b*}.

   (b) (7 pts) The set of natural numbers divisible by 25.

   (c) (7 pts) The set of strings that consist of an odd number of *a*'s, over alphabet {*a*}.

   (d) (7 pts) The set of strings over alphabet {*a*, *b*} that begin with at least two *a*'s, and end with at least two *b*'s.

3. (21 pts) For each of the following languages, write a grammar that describes the language.

   (a) (7 pts) The set of strings over alphabet {*a*, *b*} that begin with at least two *a*'s, and end with at least two *b*'s (same language as above).

   (b) (7 pts) The set of strings that consist of an even number of *a*'s, over alphabet {*a*}.

   (c) (7 pts) The set of strings of parentheses ( ), brackets [ ], and braces { } that are properly nested. For instance, ( ) [{ } ( )] is properly nested, while ([ )] is not.

4. (20 pts) Consider the following grammar for Scheme:

   expr → ATOM | list
   list → (exprs)
   exprs → expr exprs | ε

   Using this grammar, show a parse tree for the expression `(lambda (a) (* a a))`.
   Does the language described by this grammar contain a finite number of strings? If so, why? If not, why not?

5. (21 pts) Consider the following grammar for a declaration list:

   decl_list → decl ; decl_list | ε
   decl → specifier type name_list
   specifier → const | static | ε
   type → double | int
   name_list → name | name , name_list
   name → id args
   args → (decl_list) | ε

   (a) (4 pts) Indicate whether each of the following strings belongs to the language described by the grammar.

   ```
   int a (int b);
   int c (int d (int e;););
   double f, g (static int h;);
   static int i; const double j;
   ```

   (b) (7 pts) Show a leftmost derivation of the string `static int f();` under this grammar.

   (c) (10 pts) Rewrite the grammar so that arguments are separated by commas (similar to the function arguments in C). For instance, each of the following should be a valid string under the new grammar:

   ```
   int f ();
   int f (double x);
   int f (double x, int y);
   ```

6. (Extra Credit - 10 pts) Write a grammar that describes the following language: the set of strings that consist of a sequence of *a*'s followed by a sequence of *b*'s, where the number of *a*'s is odd, and equal to the number of *b*'s, over alphabet {*a*, *b*}. In other words, the language $\{a^n b^n \mid n > 0 \text{ and } n \text{ is odd}\}$.