CMSC330 Spring 2009 Quiz 3 Solutions

1. (12 pts) OCaml Polymorphic Types
Consider the OCaml type `tree` implementing a binary tree:

```ocaml
type tree =
  Empty
| Node of int * tree * tree;;
```

```ocaml
let rec fold f a t = … ;; (* apply f to nodes of t in preorder *)
let sum t = … ;; (* sum of all the nodes in t *)
```

(a. (8 pts) Implement fold (type: ('a -> int -> 'a) -> 'a -> tree -> 'a) as a preorder traversal of the tree so that the code `fold (fun a m -> m::a) [] t)` will produce a list of values in the tree starting at the root. Recall that preorder traversal performs an action on a tree node x before it recursively visits and performs the action on children of x.

```ocaml
let rec fold f a n =
  match n with
  Empty -> a
| Node (m, left, right) -> fold f (fold f (f a m) left) right
```

OR

```ocaml
let rec fold f a n =
  match n with
  Empty -> a
| Node (m, left, right) -> fold f (fold f (f a m) left) right
```

(b. (4 pts) Implement `sum` (type: `tree -> int`) using fold and anonymous functions

```ocaml
let size t = fold (fun a m -> a+m) 0 t
```

2. (14 pts) Context Free Grammars
(a. (2 pts) Write a grammar for `a^x b^y`, where `y <= x <= 2y`, for `x, y >= 0`

```ocaml
S -> aaSb | aSb | ε
```

(b. (8 pts) Given the following grammar

```ocaml
S -> S % S | S @ S | a
```

(i. (2 pts) Prove the grammar is ambiguous

Proof = multiple left-most (or rightmost) derivations for some string (need at least 2 operators)

Example: `S%S%S`

1. `S => S % S => S % S % S => a % S % S => a % a % S => a % a % a`
2. `S => S % S => a % S => a % S % S => a % a % S => a % a % a`
ii. (6 pts) Rewrite the grammar so that @ has higher precedence than % and is left associative.

\[
S \rightarrow S \% S \mid T \quad // \quad @ \text{ higher precedence}
\]
\[
T \rightarrow T @ S \mid a \quad // \quad \text{ @ left associative}
\]

3. (14 pts) Parsing
Consider the following grammar: \( S \rightarrow aA \mid A \mid a \)  \( A \rightarrow bS \mid ca \)

3a. (4 pts) Compute First sets for \( S \) and \( A \)

\[
\text{First}(A) = \text{First}(bS) \cup \text{First}(ca) = \{b, c\}
\]
\[
\text{First}(S) = \text{First}(aA) \cup \text{First}(A) = \{a, b, c\}
\]

3b. (10 pts) Write a predictive, recursive descent parser for the grammar

\[
\text{parse}_S( ) \{
\text{if (lookahead == “a”) } \{ \quad // \text{ } S \rightarrow aA
\text{match(“a”);}
\text{parse}_A( );
\}
\text{else if ((lookahead == “b”) ||
\text{ (lookahead == “c”) }) } \{ \quad // \text{ } S \rightarrow A
\text{parse}_A( );
\}
\text{else error( );}
\}
\text{parse}_A( ) \{
\text{if (lookahead == “b”) } \{ \quad // \text{ } A \rightarrow bS
\text{match(“b”);}
\text{parse}_S( );
\}
\text{else if (lookahead == “c”) } \{ \quad // \text{ } S \rightarrow aA
\text{match(“c”);}
\text{match(“a”);}
\}
\text{else error( );}
\}