

## CMSC330 Spring 2009 Quiz 3 Solutions

1. (12 pts) OCaml Polymorphic Types

Consider the OCaml type *tree* implementing a binary tree:

```

type tree =
  Empty
  | Node of int * tree * tree ;;
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.

```

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**

```

let rec fold f a = function
  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

```

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  $y \leq x \leq 2y$ , for  $x, y \geq 0$   
 $S \rightarrow aaSb \mid aSb \mid \epsilon$

- b. (8 pts) Given the following grammar

$$S \rightarrow S \% S \mid S @ S \mid a$$

- 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.  $S \Rightarrow S \% S \Rightarrow S \% S \% S \Rightarrow a \% S \% S \Rightarrow a \% a \% S \Rightarrow a \% a \% a$
2.  $S \Rightarrow S \% S \Rightarrow a \% S \Rightarrow a \% S \% S \Rightarrow a \% a \% S \Rightarrow a \% a \% a$

- ii. (6 pts) Rewrite the grammar so that @ has higher precedence than % and is left associative.

$S \rightarrow S \% S \mid T$  // @ higher precedence  
 $T \rightarrow T @ S \mid a$  // @ left associative

- c. (4 pts) Rewrite the following grammar so it can be parsed by a predictive parser

$S \rightarrow T \% S \mid T @ S \mid T$   
 $T \rightarrow id$

$S \rightarrow T L$   
 $L \rightarrow \% S \mid @ S \mid \epsilon$   
 $T \rightarrow id$

3. (14 pts) Parsing

Consider the following grammar:  $S \rightarrow aA \mid A$        $A \rightarrow bS \mid ca$

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

$\text{First}(A) = \text{First}(bS) \cup \text{First}(ca) = \{b, c\}$

$\text{First}(S) = \text{First}(aA) \cup \text{First}(A) = \{a, b, c\}$

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

```

parse_S() {
    if (lookahead == "a") { // S → aA
        match("a");
        parse_A();
    }
    else if ((lookahead == "b") ||
        (lookahead == "c")) { // S → A
        parse_A();
    }
    else error();
}

parse_A() {
    if (lookahead == "b") { // A → bS
        match("b");
        parse_S();
    }
    else if (lookahead == "c") { // S → aA
        match("c");
        match("a");
    }
    else error();
}

```