CMSC330 Spring 2014 Quiz #2 Solution

1. (8 pts) OCaml Types and Type Inference

   a. (2 pts) Give the type of the following OCaml expression

   ```
   fun x y -> (y + 2, x)
   ```

   Type = 'a -> int -> int * 'a

   b. (3 pts) Write an OCaml expression with the following type

   ```
   (float list -> float -> 'a) -> 'a
   ```

   Code = One possible answer:
   fun x -> x [1.0] 3.0;

   c. (3 pts) Give the value of the following OCaml expression. If an error exists, describe the error. The function fold is given below.

   ```
   fold ((fun x y z -> x + (y * z)) 2) 1 [1; 2; 3];;
   ```

   Value = 26

2. (8 pts) OCaml Programming

   Using either map or fold and an anonymous function, write a curried function called **divisible** which when given a number *n* and a list of ints *lst*, returns a list of all elements of *lst* that are divisible by *n* (maintaining their relative ordering). You are allowed to use List.rev (reverses a list) and the (curried) map and fold functions provided, but no other OCaml library functions. **Hint:** x is divisible by y iff (x mod y = 0) is true.

```
let rec map f l = match l with
   [] -> []
   | (h::t) -> (f h)::(map f t)
```

```
let rec fold f a l = match l with
   [] -> a
   | (h::t) -> fold f (f a h) t
```

   Example:

   divisible 4 [3;16;24] // returns [16; 24]
   divisible 3 [4;1;11] // returns [ ]
   divisible 3 [ ] // returns [ ]

   One possible solution:

```
let divisible v lst = List.rev
 (fold (fun a h -> if (h mod v = 0) then (h::a) else a) [ ] lst)
```

3. (4 pts) Context Free Grammars

Consider the following grammar:

S → aSc | b | epsilon

a. (2 pts) Describe the set of strings accepted by this grammar.

Strings of a's, followed by 0 or 1 b's, followed by c's, where the number of a's and c's is the same (and may be 0). The empty string is also part of the language.

OR

String of N a's followed by N c's, where N >= 0. The a's and c's can possibly separated by a single b. Also includes the empty string.

b. (2 pts) Draw a parse tree for the string aabcc.