CMSC330 Spring 2012 Quiz #3 Solution

1. (8 pts) OCaml Types and Type Inference

   a. (2 pts) Give the type of the following OCaml expression
      fun x y -> [x ; y] **Type = 'a -> 'a -> 'a list**

   b. (3 pts) Write an OCaml expression with the following type
      (int -> 'a) -> 'a **Code = fun x -> x 2 OR**
      let f x = x 2

   c. (3 pts) Give the value of the following OCaml expression. If an error exists, describe the error. The function map is given on the next page.
      map ((fun x y -> x+y) 1) [2;3;4] **Value = [3 ;4 ;5]**

2. (14 pts) OCaml Programming

   Solve the following OCaml programming problems. The following rules apply to both parts of this question. You are allowed to use List.rev (reverses a list) and the (curried) map and fold functions provided, but no other OCaml library functions. Your solution must run in O(n) time for input lists of length n (note that using append instead of prepend will usually make your algorithm $O(n^2)$).

```
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
```

   a. (6 pts) Write a function *growNum* which when given a number k, returns a list of the first k integers, starting from k.

      Example:
      growNum 0 = []
      growNum 1 = [1]
      growNum 4 = [1;2;3;4]

Example solution:

```
let growNum x =
    let rec g n = if n = 0 then [] else n::(g (n-1)) in (List.rev (g x));;
```

b. (8 pts) Using either map or fold and an anonymous function, write a curried function *countNum* which when given a number *n* and a list of ints *lst,* returns the number of times *n* occurs in *lst*.

Example:

countNum 5 [1;2;3;4] = 0
countNum 5 [1;2;3;4;5;6] = 1
countNum 5 [5;5;6;5] = 3

Example solution:

let countNum n lst = fold (fun a y -> if (y = n) then a+1 else a) 0 lst ;;

Partial credit:

let rec countNum n lst = match lst with [] -> 0
| (h::t) -> if (h = n) then 1+(countNum n t) else (countNum n t) ;;

3. (8 pts) Context Free Grammars

Consider the following grammar:

S → aSb | aS | epsilon

a. (2 pts) What is the set of strings accepted by this grammar?

Strings of a's followed by b's, with at least as many a's as b's.
OR $a^m b^n$, where m ≥n

b. (2 pts) Provide a derivation of the string "aab" for this grammar.

S ⇒ aSb ⇒ aaSb ⇒ aab OR S ⇒ aS ⇒ aaSb ⇒ aab

c. (4 pts) Provide a context free grammar for all strings of 0's and 1's representing all odd binary numbers (i.e., binary numbers ending in 1).

S → 0S | 1S | 1 OR S → E1 OR ...
E → EE | 1 | 0