CMSC330 Fall 2013 Quiz #3 Solution

1. (6 pts) OCaml Types and Type Inference
   
   a. (2 pts) Give the type of the following OCaml expression
      
      ```ocaml
      fun d o g -> [ g ; o ]
      ```
      
      Type = ‘a -> ’b -> ’b -> ’b list
   
   b. (2 pts) Write an OCaml expression with the following type
      
      ```ocaml
      ‘a -> ‘a list -> ‘a list
      ```
      
      Code = `fun x y -> x::y`
   
   c. (2 pts) Give the value of the following OCaml expression. If an error exists, describe the error.
      
      ```ocaml
      let x = 1 in let x = x+2 in let x = x+3 in x+1
      ```
      
      Value = 7

2. (14 pts) OCaml Programming
   
   a. (6 pts) Write a curried function `apply_n` which when given an int n, a function f, and a value v, applies f to v n times (i.e., evaluates f(f(…f(v))). You may assume initially n ≥ 0. For n=0 simply return v. You do not need to use map/fold.
      
      Example:
      
      ```ocaml
      let f x = x + 1;;
      apply_n 0 f 1   // returns 1
      apply_n 1 f 1   // returns 2
      apply_n 5 f 1   // returns 6
      ```
      
      Example solution:
      
      ```ocaml
      let rec apply_n n f arg = if (n > 0) then apply_n (n-1) f (f arg) else arg ;;
      ```
b. (8 pts) Using either map or fold and an anonymous function, write a function `bubble` which when given a list, returns a list created by going through the list from left to right and swapping two adjacent ints `x` and `y` when `x > y`. Applying `bubble` `n` times to a list of length `n` would result in a list sorted in ascending order (i.e., bubble sort). You may assume list elements may be compared with `>`. 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)`). You are not allowed to use imperative OCaml features such as int ref.

Example:

```ocaml
let bubble lst = List.rev (fold (fun a x -> match a with
    [ ] -> [x]
  | h::t -> if h > x then (h::x::t) else (x::h::t)) [ ] lst)
```

Example solution:

```ocaml
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

let bubble lst = List.rev (fold (fun a x -> match a with
    [ ] -> [x]
  | h::t -> if h > x then (h::x::t) else (x::h::t)) [ ] lst);
```