CMSC330 Fall 2009 Quiz #2 Solution

1. (6 pts) OCaml Types and Type Inference
   a. (2 pts) Give the type of the following OCaml expression
      fun x -> [x+2] **Type = int -> int list**

   b. (2 pts) Write an OCaml expression with the following type
      int -> bool **Code = many possible solutions:**
      let f x = x > 0 OR fun x -> x == 1 etc…
   c. (2 pts) Give the value of the following OCaml expression. If an error exists, describe the error.
      let x = (fun z -> z – 1) in x 1 **Value = 0**

2. (6 pts) OCaml Programming

   Using fold and an anonymous function, write a function *numAdults* which when applied to a list of ints *lst,* returns the number of elements of *lst* that are 18 or over. Example: numAdults [17;18; 21;16; 25] = 3

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

   let numAdults lst = fold (fun a y -> if (y >= 18) then (a+1) else a) 0 lst OR
   let numAdults = fold (fun a y -> if (y >= 18) then (a+1) else a) 0
   // -2 points for not using anonymous function with fold
   // -4 points for not using fold

3. (5 pts) First Sets
   Compute First sets for S and A in the following grammar:

   S → Ab A → dA
   S → c A → ε (* epsilon *)
   First(S) = {b, c, d} // (3 pts) 1 point per terminal
   First(A) = {d, ε } // (2 pts) 1 point for d, 1 point for ε

4. (3 pts) Parser
   Finish writing a predictive, recursive descent parser for the following grammar.

   S → aSb
   S → ε (* epsilon *)

   You may use the following utilities

lookahead	Variable holding next terminal
match (x)	Function to match next terminal to x
error ( )	Reports parse error for input

```
parse_S( ) {
        if (lookahead == "a") {
                match( "a" ) ; parse_S( ) ; match( "b" );    // (2 pts)
        } else {
                ;                                            // (1 pt) just return
        }
}
```