CMSC330 Fall 2011 Quiz #3 Solutions

1. (8 pts) OCaml Types and Type Inference

   a. (3 pts) Give the type of the following OCaml expression
      fun x -> [x 1] **Type =** **(int->'a) -> ('a list)**

   b. (3 pts) Write an OCaml expression with the following type
      'a list -> 'a **Code =** **fun (x::y) -> x**
      fun x -> match x with h::t -> h

   c. (2 pts) Give the value of the following OCaml expressions. If an error exists, describe the error.
      (fun x -> fun y -> x+y) 6 4 **Value = 10**

2. (16 pts) OCaml Programming

   a. (8 pts) Write a curried function *findKth* which when given a number k and a list *lst* of int (key, value) pairs, returns the kth value in the list. You may use map or fold if you wish, but it is not required. You may assume *lst* contains at least k pairs. Example:
      findKth 1 [(1,2);(5,9);(9,3)] = 2 // since 2 is 1st value
      findKth 2 [(1,2);(5,9);(9,3)] = 9 // since 9 is 2nd value

```
let rec findKth k lst = match lst with
    (x,y)::t -> if k = 1 then y else (findKth (k-1) t)
```

   b. (8 pts) Using either map or fold and an anonymous function, write a curried function *findGreaterThan* which when given a number *n* and a list of ints *lst,* returns a list of all elements of *lst* greater than n (maintaining their relative ordering). You may assume (x > y) returns true when x is larger than y. Example:
      findGreaterThan 20 [33;18;21;19] = [33;21]
      findGreaterThan 65 [33;18;21;19] = []

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

3. (6 pts) Context Free Grammars
   Consider the following grammar: S → E+E | E*E
   E → 0 | 1 | n | (S)

   a. (2 pts) What is the set of strings accepted by this grammar?
      Arithmetic expressions involving + and *.

   b. (4 pts) Provide a *leftmost* derivation of the string "(n+1)*n" for this grammar.
      S → E*E → (S)*E → (E+E)*E → (n+E)*E → (n+1)*E → (n+1)*n