CMSC330 Fall 2011 Quiz #3

Name ___________________________________

Discussion Time (circle one): **9am** **10am** **11am** **12pm** **1pm** **2pm**

Do not start this quiz until you are told to do so.
Instructions

- You have 20 minutes for this quiz.
- This is a closed book exam. No notes or other aids are allowed.
- Answer essay questions concisely using 2-3 sentences. Longer answers are not necessary and a penalty may be applied.
- For partial credit, show all of your work and clearly indicate your answers.
- Write neatly. Credit cannot be given for illegible answers.

1. (8 pts) OCaml Types and Type Inference

   a. (3 pts) Give the type of the following OCaml expression

      fun x -> [x 1] **Type =**

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

      ‘a list -> ‘a **Code =**

   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 =**

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

2. (16 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)).

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

3. (6 pts) Context Free Grammars

Consider the following grammar:

$$S \rightarrow E+E \mid E*E$$
$$E \rightarrow 0 \mid 1 \mid n \mid (S)$$

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

b. (4 pts) Provide a *leftmost* derivation of the string “(n+1)*n” for this grammar.