CMSC330 Spring 2014 Midterm #2

Name: ________________________________

Discussion Time	**10am**	**11am**	**noon**	**1pm**	**2pm**
TA Name (circle):	**Tammy**	**Tammy**	**Tammy**	**Daniel**	**Daniel**
		Ilse	**Casey**	**Ian**	

Instructions

- Do not start this test until you are told to do so!
- You have 75 minutes to take this midterm.
- This exam has a total of 100 points, so allocate 45 seconds for each point.
- This is a closed book exam. No notes or other aids are allowed.
- Answer essay questions concisely in 2-3 sentences. Longer answers are not needed.
- For partial credit, show all of your work and clearly indicate your answers.
- Write neatly. Credit cannot be given for illegible answers.
- You may **not** use imperative OCaml constructs unless otherwise noted.

	Problem	Score
1	OCaml types & type Inference	/16
2	OCaml higher order & anonymous functions	/10
3	OCaml programming	/18
4	OCaml polymorphic datatypes	/8
5	Context free grammars	/10
6	Using context free grammars	/14
7	Parsing	/15
8	Multithreading	/9
	Total	/100

HONOR PLEDGE: I pledge on my honor that I have not given or received any unauthorized assistance on this assignment/ examination. SIGNATURE: ______________________

1. (16 pts) OCaml Types and Type Inference

   Give the type of the following OCaml expressions:

   a. (2 pts) fun b -> b + 8 **Type =**

   b. (3 pts) fun b -> b 8 **Type =**

   Write an OCaml expression with the following type:

   c. (2 pts) ‘a -> 'a -> 'a list list

      Code =

   d. (3 pts) (int -> int) -> (int -> int)

      Code =

   Give the value of the following OCaml expressions. If an error exists, describe it

   e. (3 pts) (fun k t m -> [m; t]) 2 4 6

      Value / Error =

   f. (3 pts) (fun (k::t::m) -> (m, t)) [2;4;6;8]

      Value / Error =

2. (10 pts) OCaml higher-order & anonymous functions

   Using either map or fold and an anonymous function, write a curried function *process* which when given a predicate function **p**, a compute function **f**, and a list **lst**, returns a list of the results of applying **f** to the elements of lst, but only where **p** is true for that element of the list. If **p** is false for an element of the list, it does not contribute to the final result. The relative order of the elements in the new list must the same as in **lst**.

   Your function must run in linear time. You may not use any library functions, with the exception of the List.rev function, which reverses a list in linear time. You may not use imperative OCaml (i.e., no ref variables). For example, the results of calling *process* will be as follows:

```
Example:
let p x = x > 2;;
let f x = x+1;;
process p f  [ ] = [ ]
process p f  [ 1; 2 ] = [ ]
process p f  [ 1; 3; 2; 4 ] = [ 4; 5 ]
```

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

3. (18 pts) OCaml programming

   Write a function ***allPairs*** with type (‘a list -> ‘a list list) which given a list *lst* with n elements, returns a list of all n^2 pairs of elements in *lst* (in any order), where each pair of elements is in a list of length 2.

   You may not use any library functions, with the exception of the @ function (which concatenates two lists). You may use helper functions. You may use map or fold, but it is not required. Your function must execute in polynomial time. You may not use imperative OCaml constructs.

   Examples:

```
allPairs [ ] ;;        (* = [ ] *)
allPairs [1] ;;        (* = [ [1;1] ] *)
allPairs [1;2] ;;      (* = [ [1;1]; [1;2]; [2;1]; [2;2] ] *)
allPairs [1;2;5] ;;  (* = [ [1;1]; [1;2]; [1;5]; [2;1]; [2;2]; [2;5]; [5;1]; [5;2]; [5;5] ] *)
```

4. (8 pts) OCaml polymorphic types

   Consider the OCaml type *bitwiseExp* implementing bitwise expressions (think of it as a binary tree where all leaves are 0's or 1's). For all answers, your code must work in linear time (i.e., avoid multiple passes over the tree). You are not allowed to use any OCaml library functions except +. You may use helper functions.

```
type bitwiseExp =
   Zero                               (* 0 *)
 | One                                (* 1 *)
 | Pair of bitwiseExp * bitwiseExp    (* b1 • b2 *)
```

bitwiseExp	onesExp
let x = Zero;;	0
let y = One;;	1
let p = Pair (x,y);;	1
let q = Pair (y,y);;	2

   a. (2 pts) Write an OCaml expression with type bitwiseExp that is equivalent to the expression "0 • (1 • 0)"

   b. (6 pts) Write a function o*nesExp* of type (bitwiseExp -> int) that takes a bitwise expression and returns the number of 1's in the expression.

5. (10 pts) Context free grammars.

Consider the following grammar (S = start symbol and terminals = **a b % &**):

$S \rightarrow$ a | S%S | &T

$T \rightarrow$ a | b

a. (1 pt each) Indicate whether the following strings are generated by this grammar

i.	**%a**	Yes	No	(circle one)	
ii.	**a%&b**		Yes	No	(circle one)
iii.	**&a&b**		Yes	No	(circle one)

b. (3 pts) Draw a parse tree for the string "&b%a"

c. (4 pts) Is the grammar is ambiguous? Provide proof if you believe it is ambiguous.

6. (14 pts) Using context free grammars.
   a. (6 pts) Given the following production for B, create a grammar that generates OCaml tuples of strings (for the 3 strings Joe, Laura, and Mike). Your grammar should be able to generate strings such as (“Joe”), (“Joe”, “Mike”), (“Joe”, “Laura”, “Laura”), etc...

      B → ***“Mike” | “Laura” | “Joe”***

   b. (8 pts) Consider the following grammar:

      S → c | d | e | S%S | S@S | &S

   Modify the grammar above to make the % operator left associative, the @ operator right associative, and make @ have the lowest precedence and & have the highest precedence.

7. (15 pts) **Parsing (This question is irrelevant. We did not cover Parsing in this semester.)**

Consider the following grammar, where S, A, B are nonterminals, and a, b, c, d are terminals.

$$S \rightarrow a \mid bB \mid Ac$$
$$A \rightarrow SAd \mid \text{epsilon}$$
$$B \rightarrow c \mid b$$

a. (8 pts) Calculate FIRST sets for S, A, and B

FIRST(S) = { }

FIRST(A) = { }

FIRST(B) = { }

b. (7 pts) Using pseudocode, write *only* the parse_A function found in a recursive descent parser for the grammar. You may assume the functions parse_S , parse_B already exist.

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_A( ) {    // your code starts here
```

8. (9 pts) Multithreading

Consider the following multithreaded Java 1.4 code. Assume there are multiple producer and consumer threads being executed in the program, but only a single Buffer object. Questions about the "last statement executed" by two threads x & y refer to the most recently executed statement by those threads at some arbitrary time during the program execution. It does not mean the last statement executed by a thread before the thread exits. If a situation is possible, you need to give an example of how it is possible (e.g., thread x gets to statement a, then thread y gets to statement b). If a situation is not possible, you need to explain why.

```
class Buffer {
  void produce(o) {
    synchronized (this) {
1.   while (...) wait( );
2.   ...
3.   notifyAll( );
4.   ...
    }
  }
```

```
  Object consume( ) {
    synchronized (this) {
5.   while (...) wait( );
6.   ...
7.   notifyAll( );
8.   ...
    }
  }
}
```

a. (3 pts) Is it possible given two threads x and y for the last statement executed by thread x to be statement 5, and the last statement executed by thread y to be statement 7? Explain your answer.

b. (3 pts) Is it possible given two threads x and y for the last statement executed by thread x to be statement 7, and the last statement executed by thread y to be statement 6? Explain your answer.

c. (3 pts) Is it possible given two threads x and y for the last statement executed by thread x to be statement 6, and the last statement executed by thread y to be statement 1? Explain your answer.