CMSC330 Spring 2013 Midterm #2

Name:

_____

Discussion Time	**9am**	**10am**	**11am**	**Noon**	**1pm**
TA Name (circle):	**Ilse**	**Daniel**	**Casey**	**Yoav**	**Ilse**
	Richard	**Richard**	**Richard**		

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.

	Problem	Score
1	OCaml types & type Inference	/16
2	OCaml programming	/8
3	OCaml higher order & anonymous functions	/14
4	OCaml polymorphic datatypes	/14
5	Context free grammars	/10
6	Using context free grammars	/14
7	Parsing	/14
8	Multithreading	/10
	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. (3 pts) (fun c a t -> 5) 6 7 **Type =**

   b. (3 pts) fun d o g -> [g o] **Type =**

   Write an OCaml expression with the following type:

   c. (3 pts) (int -> int -> ‘a) -> ‘a

      Code =

   d. (3 pts) ‘a -> (int -> ‘b) -> ‘b list

      Code =

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

   e. (2 pts) (fun x -> fun y -> x y) (fun z -> z + 2) 3

      Value / Error =

   f. (2 pts) (fun x y -> x + y) (2 3)

      Value / Error =

2. (8 pts) OCaml programming

Write a function *nth* which given a number *n* and a list *lst*, returns the *n*th element of *lst*. You may assume the list has at least n members.

You may not use any library functions, with the exception of the List.rev function, which reverses a list in linear time. Your function must run in linear time (i.e., not use append/reverse for every element of the list). You may use helper functions.

Examples:

nth 1 [4;5;6;7] = 4
nth 2 [4;5;6;7] = 5
nth 3 [4;5;6;7] = 6
nth 2 [[1;2] ; [3;4] ; [5]] = [3;4]

3. (14 pts) OCaml higher-order & anonymous functions

Using either map or fold and an anonymous function, write a curried function *partition* which when given a predicate function f and a list x, returns a tuple of two lists y, z, where y is a list of all the members of x for which f is true, and z is a list of all the members of x for which f is false. The members of y and z must be in the same relative ordering as in x.

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).

Example:
partition (fun x -> true) [1;2;3] = ([1;2;3],[ ])
partition (fun x -> x = 0) [1;2;3] = ([ ],[1;2;3])
partition (fun x -> x < 2) [1;2;3] = ([1],[2;3])
partition (fun x -> x > 2) [1;2;3] = ([3],[1;2])

```
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
```

4. (14 pts) OCaml polymorphic types

Consider the OCaml type *boolExp* implementing boolean expressions:

```
type boolExp =
      True
    | False
    | And of boolExp * boolExp
    | Not of boolExp
```

a. (2 pts) Write an OCaml expression with type boolExp that is equivalent to the expression "true && (not false)"

b. (12 pts) Write a function *evalExpr* of type (boolExp -> bool) that takes an Boolean expression tree and calculates its value. 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 &&, ||, and *not*. You may use helper functions.

Examples:

```
let x = True;;                (* (evalExpr x) returns true  *)
let y = False;;               (* (evalExpr y) returns false *)
let z = Not x;;               (* (evalExpr z) returns false *)
let p = And (x,x);;           (* (evalExpr p) returns true  *)
let q = And (p,z);;           (* (evalExpr q) returns false *)
let r = Not q;;               (* (evalExpr r) returns true  *)
```

5. (10 pts) Context free grammars.

Consider the following grammar (S = start symbol and terminals = **0 1 ^ # !**):

$S \rightarrow S\text{^}S \mid S\#S \mid S! \mid E$

$E \rightarrow 0 \mid 1$

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

i.	**^1**	Yes	No	(circle one)
ii.	**0#1!^1**	Yes	No	(circle one)
iii.	**0!^1!**	Yes	No	(circle one)

b. (3 pts) Draw a parse tree for the string "1#0!"

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 grammar for generating OCaml *bools*, create a grammar that generates OCaml expressions of type *bool list.* Your grammar should be able to generate strings such as "[]", "[true]", "[false]", "[true;true]", etc...

      B → ***true*** **|** ***false***

   b. (8 pts) Consider the grammar from Problem 5 again:

      S → S^S | S#S | S! | E
      E → 0 | 1

      Modify the grammar to make the # operator *left associative* and the ^ operator *right associative*. Make ! have the highest precedence, # the lowest precedence, and ^ medium precedence.

7. (14 pts) Parsing

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

S →	AB	|	Ac
A →	aA	|	epsilon
B →	b	|	epsilon

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

FIRST(S) = { }

FIRST(A) = { }

FIRST(B) = { }

b. (6 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. (10 pts) Multithreading

Consider the following multithreaded Java 1.4 code. Assume there are multiple producer and consumer threads being executed in the program.

```
class Buffer {
  Buffer ( ) {
    Object buf = null;
    boolean empty = true;
  }
```

```
  void produce(o) {
    synchronize (buf) {
1.    while (!empty) wait( );
2.    empty = false;
3.    notifyAll( );
4.    buf = o;
    }
  }
```

```
  Object consume( ) {
    synchronize (buf) {
5.    while (empty) wait( );
6.    empty = true;
7.    notifyAll( );
8.    return buf;
    }
  }
}
```

a. (3 pts) Is it possible given two threads x and y for the last statement executed by both threads to be statement 2 in the code above? 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 2, and the last statement executed by thread y to be statement 6? Explain your answer.

c. (4 pts) Is it possible given two threads x and y for the last statement executed by thread x to be statement 1, and the last statement executed by thread y to be statement 2? Explain your answer.