CMSC330 Fall 2010 Midterm #2

Name _____

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

Do not start this exam until you are told to do so!

Instructions

- 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.
- If you have a question, please raise your hand and wait for the instructor.
- Answer essay questions concisely using 2-3 sentences. Longer answers are not necessary and a penalty may be applied.
- In order to be eligible for partial credit, show all of your work and clearly indicate your answers.
- Write neatly. Credit cannot be given for illegible answers.
- **You are not allowed to use any OCaml library functions unless otherwise noted**.

	Problem	Score
1	OCaml types & type Inference	/22
2	OCaml programming	/12
3	OCaml higher order & anonymous functions	/12
4	OCaml polymorphic datatypes	/14
5	Context free grammars	/22
6	Parsing	/18
	Total	/100

1. (22 pts) OCaml Types and Type Inference

   Give the type of the following OCaml expressions

   a. (2 pts) fun x y -> [x + y] **Type =**

   b. (3 pts) fun x y -> [x ; y] **Type =**

   c. (3 pts) fun x y -> [x y] **Type =**

   Write an OCaml expression with the following type

   d. (2 pts) bool -> int **Code =**

   e. (3 pts) bool -> int -> int **Code =**

   f. (3 pts) (bool -> int) -> int **Code =**

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

   g. (2 pts) let x = 2 in let x = 4 in x+8 **Value / Error =**

   h. (2 pts) let x = 2 in let y = x+4 in x+y **Value / Error =**

   i. (2 pts) let x = 2 in let x = x+4 in x+8 **Value / Error =**

2. (12 pts) OCaml programming

   a. (8 pts) Implement a function *generateFinder* which when passed a list of strings *wanted* returns a function that takes a string *name* as argument, and returns true if *name* is in *wanted*.

      You are not allowed to use any OCaml library functions.

      Example:
```
let findJedi = generateFinder [“Yoda”; “Luke”; “Obi-Wan”] ;;
let findSith = generateFinder [“Palpatine”; “Vader”] ;;
findJedi “Luke”;;          (* returns true *)
findJedi “Vader”;;         (* returns false *)
findSith “Vader”;;         (* returns true *)
```

   b. (4 pts) What feature of closures allows a simple solution to the problem above? Explain.

3. (12 pts) OCaml higher-order & anonymous functions

Recall that association lists are lists of tuples, with the 1st element of each tuple used as a key and the 2nd element of each tuple used as its associated value.

Using fold and an anonymous function, implement a function *countWords* which when passed a **sorted** list of words returns a **sorted** association list with the number of times each words occurs. The type *countWords* is string list -> (string * int) list.

Note you do not need to sort the list of words or associations. Simply build the association list based on the order of the words in the original list.

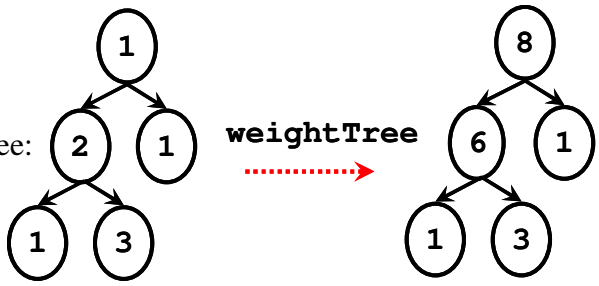
Your code must work in linear time (i.e., avoid using list append). You are not allowed to use any OCaml library functions except for *List.rev*, which reverses a list in linear time.

Partial credit given for solutions which do not use fold and/or an anonymous function.

```
Example:    countWords [ ] ;;                  (* returns [ ] *)
            countWords [“a”;“b”] ;;            (* returns [(“a”,1);(“b”,1)] *)
            countWords [“a”;“b”;“b”;“c”] ;;    (* returns [(“a”,1);(“b”,2);(“c”,1)] *)
```

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

4. (14 pts) OCaml polymorphic types



Consider the OCaml type *tree* implementing a binary tree:

```
type tree =
  Empty
 | Node of int * tree * tree
```

We define the *weight* of a node to the sum of the values of the node and all the nodes of its children. Write a function *weightTree* of type (tree -> tree) that takes a tree and returns a new tree of the same shape but where the value at each node is replaced with its weight.

Your code must work in linear time (i.e., avoid recalculating the weight of the same part of the tree). You are not allowed to use any OCaml library functions. You may use helper functions.

Examples:

```
weightTree (Empty);;                          (* returns Empty *)
weightTree (Node (5, Empty, Empty));;         (* returns Node (5, Empty, Empty)  *)
weightTree (Node (2, Node (1, Empty, Empty), Node (3, Empty, Empty))));;
        (* returns Node (6, Node (1, Empty, Empty), Node (3, Empty, Empty)) *)
```

5. (22 pts) Context free grammars
   a. (8 pts) Let *nested int lists* be int lists with arbitrary number of levels of nesting. For instance, types for nested int lists include int list, int list list, int list list list, etc.

      Give a grammar for the type of OCaml functions that have a single nested int list argument, and returns a single nested int list, given the three terminals **int, list, ->.** I.e., your grammar should be able to generate int list -> int list, int list -> int list list, int list list list -> int list list, etc…

      Make your grammar unambiguous and left recursive.

   Consider the following grammar: S → S x S | S y S | 0 | 1

   b. (2 pts) Provide a derivation for the string "1y0".

   c. (4 pts) Prove that the grammar is ambiguous

   d. (4 pts) What could occur when parsing ambiguous grammars, and why is it a problem?

   e. (4 pts) Explain why the example grammar cannot be parsed using a predictive recursive descent parser.

6. (18 pts) Parsing
   Consider the following grammar

   $S \rightarrow A+S \mid aAc$

   $A \rightarrow bA \mid \varepsilon$ (* epsilon *)

   a. (6 pts) Compute First sets for S and A

   b. (12 pts) Using pseudocode, write a recursive descent parser for the grammar.

   Use the following utilities:

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