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.