

# CMSC330 Spring 2009 Midterm #2

Name \_\_\_\_\_

Discussion Time (circle one):      9am              10am

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

## Instructions

- You have 50 minutes for to take this midterm.
- This exam has a total of 100 points, so allocate 30 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.

	Problem	Score
1	Prog Lang & Automata	/12
2	OCaml Types	/23
3	OCaml Programming	/18
4	CFGs	/13
5	Parsing	/16
6	Scoping & Lazy Eval	/10
7	Parameter Passing	/8
	Total	/100



3. (18 pts) OCaml Programming

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

```
type bst =  
  Empty  
  | Node of int * bst * bst ;;  
  
let rec map f t = ... (* type = (int -> int) -> bst -> bst *)  
let add1 t = (* type = bst -> bst *)
```

- a. (12 pts) Implement a function *map f t* that returns a tree with *f* applied to the int value of each node in tree *t*.

- b. (6 pts) Using *map* and an anonymous function (no helper functions) write a function *add1 t* that returns a tree nearly identical to tree *t*, but with the value of each node increased by 1.

4. (13 pts) Context Free Grammars

- a. (3 pts) Write a grammar for  $a^x b^y$ , where  $x \leq y \leq 3x$ , for  $x, y \geq 0$

Given the following grammar

$$S \rightarrow S \wedge T \mid T$$
$$T \rightarrow V \# T \mid V$$
$$V \rightarrow \text{id}$$

- b. (2 pts) Which operator has higher precedence?

- c. (2 pts) Which operator(s) is right associative?

- d. (6 pts) Rewrite the following grammar so it can be parsed by a predictive parser

$$S \rightarrow S \wedge T \mid S \# T \mid T$$
$$T \rightarrow \text{id}$$

5. (16 pts) Parsing

Consider the following grammar:  $S \rightarrow Ab \mid d$        $A \rightarrow aA \mid \text{epsilon}$

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

b. (10 pts) Write a predictive, recursive descent parser for the grammar

6. (10 pts) Scoping & Lazy Evaluation

Consider the following OCaml code.

```
let app f y = let x = 2 in f y ;;
let add x y = let incr x = x+y in app incr (x+5) ;;
(add 3 4) ;;
```

- a. (2 pts) What value is returned by `(add 3 4)` with static scoping? Explain.
  
- b. (4 pts) What value is returned by `(add 3 4)` with dynamic scoping? Explain.
  
- c. (4 pts) Rewrite the following code (using `thunks`) so that the result is the same as if OCaml used call-by-name, even though OCaml uses call-by-value.

```
let f x = x+1 ;;
f y ;;
```

7. (8 pts) Parameter Passing

Consider the following C code.

```
void swap(int f, int g) {
    int tmp = f;  f = g;  g = tmp;
}
int main( ) {
    int i = 2;
    int a[] = {2, 0, 1};
    swap(i, a[i]);
    printf("%d %d %d %d\n", i, a[0], a[1], a[2]);
}
```

- a. (2 pts) Give the output if C uses call-by-value
  
- b. (2 pts) Give the output if C uses call-by-reference
  
- c. (4 pts) Give the output if C uses call-by-name