CMSC330 Spring 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.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 OCaml types &amp; type Inference</td>
<td>/16</td>
</tr>
<tr>
<td>2 Higher order &amp; anonymous functions</td>
<td>/14</td>
</tr>
<tr>
<td>3 OCaml polymorphic datatypes</td>
<td>/16</td>
</tr>
<tr>
<td>4 Context free grammars</td>
<td>/16</td>
</tr>
<tr>
<td>5 Parsing</td>
<td>/22</td>
</tr>
<tr>
<td>6 Operational semantics</td>
<td>/16</td>
</tr>
<tr>
<td>Total</td>
<td>/100</td>
</tr>
</tbody>
</table>
1. (16 pts) OCaml Types and Type Inference

Give the type of the following OCaml expression

a. (2 pts) \([1 ; 2]\] \(\text{Type} =\)

b. (3 pts) \(\text{fun x -> 2::x}\) \(\text{Type} =\)

Write an OCaml expression with the following type

c. (2 pts) \(\text{int list -> int}\) \(\text{Code} =\)

d. (4 pts) \(\text{int -> bool) -> int}\) \(\text{Code} =\)

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

e. (2 pts) \(\text{if (1 < 2) then 3}\) \(\text{Value / Error} =\)

f. (3 pts) \(\text{let f x = f 2 in 1}\) \(\text{Value / Error} =\)
2. (14 pts) Higher order & anonymous functions

A prefix sum is an operation on lists in which the $n^{th}$ element in the result list is obtained from the sum of the first $n$ elements in the operand list. Using the following code for fold and an anonymous function, write a function prefixSum which given a list of ints, returns the prefix sum for the list.

You are not allowed to use any helper functions or OCaml library functions, with the exception of List.rev (which reverses a list).

Partial credit given for solutions which do not use fold.

Example:  
prefixSum [ ] = []  
prefixSum [1;1;1;1] = [1;2;3;4;5]  
prefixSum [1;2;3] = [1;3;6;10]
3. (16 pts) OCaml polymorphic datatypes

Consider the OCaml type `tree` implementing a binary tree of ints:

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

a. (4 pts) Write an OCaml expression creating the data structure for a binary tree where the root node has value 5 and has one child node with value 7.

b. (5 pts) Implement a function `count5` that takes a tree and returns the number of nodes with the value 5. You may use helper functions (though they are not needed).

c. (7 pts) Implement a function `prune5` that takes a tree and returns a tree where all nodes with the value 5 (and their subtrees) are removed. You may use helper functions (though they are not needed).
4. (16 pts) Context free grammars
   Consider the following grammar: \( S \rightarrow S \times T \mid T \quad T \rightarrow a \mid b \)
   
   a. (3 pts) Describe the set of strings generated by the grammar

   b. (3 pts) Provide a left-most derivation for the string “\( axxbxb \)”.  

   c. (2 pts) Provide a parse tree for the string “\( axxbxb \)”.  

   d. (2 pts) What is the associativity of the \( \times \) operator for the grammar?

   e. (6 pts) Apply the algorithm discussed in class to transform the grammar so that it can be parsed using a recursive descent parser.
5. (22 pts) Parsing semantics (This question is irrelevant. We did not cover parsing in this semester)

Consider the following grammar
\[ S \rightarrow \text{Abc} \mid dS \mid \varepsilon \quad (* \text{epsilon} * ) \]
\[ A \rightarrow aSA \mid f \]
a. (8 pts) Compute First sets for S and A

b. (14 pts) Using pseudocode, write a recursive descent parser for the grammar.

Use the following utilities:

<table>
<thead>
<tr>
<th>lookahead</th>
<th>Variable holding next terminal</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lookahead == “S” when at end of input</td>
</tr>
<tr>
<td>match ( x )</td>
<td>Function to match next terminal to x</td>
</tr>
<tr>
<td>error ( )</td>
<td>Reports parse error for input</td>
</tr>
</tbody>
</table>
6. (16 pts) Operational semantics (This question is irrelevant. We did not cover operational semantics in this semester)

   a. (4 pts) Consider the following operational semantics judgement. State in English what this statement is expressing:

   \[
   \bullet, x:1 ; (+ x 2) \rightarrow 3
   \]

   b. (12 pts) In an empty environment, to what value \( v \) will the expression

   \[(\text{fun } z = z) (+ 1 2)\]

   evaluate to? In other words, find a \( v \) such that you can prove the following:

   \[
   \bullet ; (\text{fun } z = z) (+ 1 2) \rightarrow v
   \]

   Use the operational semantics rules given in class. Show the complete proof that stacks uses of these rules.