CMSC330 Summer 2010—Midterm #2
July 7, 2010

Name _____________________________________________________________

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

Instructions

• You have 80 minutes for this exam.

• The exam is worth 100 points. Allocate your time wisely: some hard questions are worth only a few points, and some easy questions are worth a lot of points.

• If you have a question, please raise your hand and wait for the instructor.

• You may use the back of the exam sheets if you need extra space.

• 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

<table>
<thead>
<tr>
<th>Problem</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Programming languages</td>
<td>/12</td>
</tr>
<tr>
<td>2 OCaml Types &amp; Type Inference</td>
<td>/16</td>
</tr>
<tr>
<td>3 Lambda Calculus</td>
<td>/16</td>
</tr>
<tr>
<td>4 Operational Semantics</td>
<td>/16</td>
</tr>
<tr>
<td>5 OCaml Programming</td>
<td>/40</td>
</tr>
<tr>
<td>Total</td>
<td>/100</td>
</tr>
</tbody>
</table>
1. (Programming Languages, 12 pts)

(a) (4 pts) What is modularity? Briefly describe how this concept can be implemented in two different programming languages (i.e., OCaml, Ruby, Java, C, C++).

(b) (3 pts) What is the difference between dynamic and lexical scoping?

(c) (5 pts) OCaml functions take only one parameter. What are the two ways of passing more than one value to an OCaml function? Briefly describe and give an example of each type of function call.
2. (OCaml Types and Type Inference, 16 pts)
   (a) (5 pts) Give the types of the following OCaml expressions.
      
      i. \( [(2,3);(4,5)] \)  \text{TYPE:}
      
      ii. let f x y = x + y  \text{TYPE:}

   (b) (6 pts) Write an OCaml expression with the following types.

      i. int -> int -> int list  \text{CODE:}

      ii. bool -> (int -> int)  \text{CODE:}

   (c) (5 pts) Give the value of the following OCaml expressions or describe the error that exists.

      i. let x = 6 in let x = x in x+6  \text{VALUE:}

      ii. (fun x -> fun y-> x * y) (5,6)  \text{VALUE:}

3. (Operational Semantics, 16 pts)

   (a) (4 pts) Explain in English the meaning of the following operational semantics statement: \( \bullet, x:5 \; ; \; (+ \; x \; 1) \rightarrow 6 \)
(b) (12 pts) Using the operational semantics rules given in class, prove the following statement:

\[ \text{\textbullet;}(\text{fun } x = + x 4) \ ( + 2 3) \rightarrow v \]

for some value \( v \). For full credit, you must show the complete tree of rule applications.

4. (Lambda Calculus, 16 pts)

(a) (8 pts) Apply \( \beta \)-reduction to the following \( \lambda \)-expressions as much as possible.

i. \( (\lambda z. \lambda x. z \ x \ x) \ (\lambda z. z) \ z \)
ii. \((\lambda x. x) (\lambda y. y)\) \((\lambda y. y)\)

(b) (8 pts) Using the Church numeral encodings given below, prove that 0 * 3 = 0

\[
M \ast N = \lambda x. (M (N x)) \\
0 = \lambda f. \lambda y. y \\
1 = \lambda f. \lambda y. f \ y \\
2 = \lambda f. \lambda y. f \ (f \ y) \\
3 = \lambda f. \lambda y. f \ (f \ (f \ y))
\]
5. (OCaml Programming, 40 pts)

(a) (12 pts) Functional Programming Basics
Write an OCaml function `intersect` that takes two lists `s1` and `s2` as input and returns a list that is the intersection of these two lists.

Example: `intersect [1;2;3;4] [2;4;6;8;10] = [2;4]

You can assume that the elements of the lists can be compared using `=` or `==`, but the functions should be polymorphic. You may only use basic functional OCaml (no modules allowed!). You may also write helper functions if necessary.
(b) (12 pts) Higher Order and Anonymous Functions

Using fold (given below) and an anonymous function, write a function `get_greater` that takes an int `n` and a list `l` and returns a list containing all elements of `l` that are greater than `n`.

Example: \( \text{get_greater} \ 2 \ [1;2;3;4;5] = [3;4;5] \)

Partial credit will be given for solutions that do not use fold.

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

(c) (16 pts) OCaml Polymorphic Datatypes

A map or associative array is a data structure comprised of a collection of key-value pairs (in Ruby this was called a hash). Consider the following OCaml polymorphic type `(a,b) map` that implements a map with keys of type `a` and values of type `b`:

```ocaml
type (a,b) map =
  Nil |
  Map of (a * b) * (a,b) map;;
```

For the following questions, you may only use basic functional OCaml (no modules!). If helpful, you can reuse solutions to one part in your solution to a later part.

(i) (3 pts) Write an OCaml expression that creates map with the following key=>value pairs:

\( 1=>"foo" \)
\( 2=>"bar" \)
(ii) (5 pts) Write an OCaml function `keys` that takes a map and returns a list of the map’s keys.

(ii) (8 pts) Write an OCaml function `put` that takes a key-value pair \((k, v)\) and a map \(m\) and returns a new map with \((k, v)\) added to \(m\). If the key \(k\) already exists, then `put` will just update the value. You can use helper functions if necessary.