CMSC330 Spring 2008 Final Exam

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

Instructions

- You have 120 minutes to take this midterm.
- This exam has a total of 240 points, so allocate 30 seconds for each point.
- There is also a 16 point extra credit problem if you finish early.
- 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 1 sentence, or *at most* 2 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 will not be given for illegible answers.
- Good luck!

1. (9 pts) Programming languages
    a. (3 pts) Give an example of a language feature (not including library functions) in Ruby & OCaml whose syntax is different but semantics are the same.
    b. (3 pts) Give an example of a language feature (not including library functions) in Ruby & Java whose syntax is the same but semantics are different.
    c. (3 pts) Describe a feature of an old programming language which proved vexing for programmers.

2. (12 pts) Ruby features
    What is the output (if any) of the following Ruby programs? Write FAIL if code does not execute.
    a.
```
a = [2,3]
a[3] = “c”
a.push(“b”)
puts a
```
    b.
```
if  “CMSC 330” =~ /([0-9]+)/ then
    puts $1
else
    puts “Error”
end
```
    c.
```
s = "CMSC 330 is too HARD!"
s.scan(/[A-Z]+/) { |x| puts x }
```
    d.
```
h = {2 => 3, 1 => 2, 0=>1}
puts "#{h[1]} #{h[2]}"
```

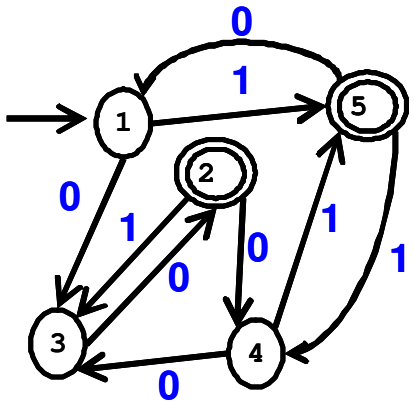
3. (20 pts) Ruby programming

    Consider the following programming problem. We need to read from a text file a list of cars and the years they were produced. The file contains a number of lines of the form <car> <year>, where <car> is composed of letters, <year> is composed of digits, and the two are separated by a single space. Write a Ruby program to read in the list of cars and their model years, and output the list of cars and years in sorted order, with each car on a separate line. Any lines not in the correct format should be ignored.

Skeleton Code	Example Input	Example Output

file = File.new(ARGV[0], "r")
until file.eof? do
  line = file.readline
  …
end
// a = Array.new ; a.sort!
// print 1 ; puts 2
	yaris 2008
	prius 2004
	prius 2007
	rav4 1998
	yaris 2006
	prius 2006
	camry 1983
		camry 1983
		prius 2004 2006 2007
		yaris 2006 2008

4. (30 pts) Regular expressions finite automata
   a. (6 pts) Reduce the regular expression (01)* | 1 to an NFA, using the algorithm shown in class.
   b. (12 pts) Reduce the NFA to a DFA using the subset algorithm. Show the NFA states represented by each DFA state.
   c. (12 pts) Minimize the following DFA using Moore's algorithm. Show the entire sequence of partitions created (list the DFA states in each partition), and explain why each partition was split.



5. (10 pts) Grammars and parsing
   a. (4 pts) Write a grammar for $a^x b^y a^z$, where x = y – z, for x,y,z ≥ 0
   b. (6 pts) Rewrite the following grammar

      S → S and S | not S | true | false

      so that "and" is right associative & has higher precedence than "not".

6. (24 pts) OCaml Types & Type Inference

   Give the type of the following OCaml expressions:

   a. (3 pts) let f x = match x with

      [] -> []

      | (h::_) -> h

   b. (3 pts) let f x y = y (x+1)

   Write an OCaml expression with the following types:

   c. (3 pts) int -> (int -> 'a) -> 'a
   d. (3 pts) (int -> 'a) -> int -> 'a

   What are the values of the following OCaml expressions? If an error exists, describe the error.

   e. (3 pts) let x = x in 1
   f. (3 pts) let x = 1 in if x > 0 then x else print_string "Error"
   g. (3 pts) (fun x -> if x > 0 then x else (–x)) 2

   h. (3 pts) Are type inference and strong typing orthogonal language features? Explain.

7. (14 pts) OCaml Polymorphic Types

Consider a OCaml module Bst that implements a binary search tree:

```
module Bst = struct
  type bst =
    Empty
    | Node of int * bst * bst

  let empty = Empty              (* empty binary search tree          *)

  let is_empty = function        (* return true for empty bst         *)
    Empty -> true
    | Node (_, _, _) -> false

   (* Implement the following functions
          val height : bst -> int
          val post_fold : ('a -> int -> 'a) -> 'a -> bst -> 'a
    *)
    let rec height =             (* height of bst      *)
    let rec post_fold f a t =    (* apply f to nodes of t in postorder    *)
  end
```

a. (6 pts) Implement height. You may use Pervasives.max (a -> 'a -> 'a), which returns the greater of its two arguments.
b. (8 pts) Implement post_fold as a version of fold which performs a *postorder* traversal of the tree (visits in order 1: left subtree, 2: right subtree, 3: node).

Example:

```
let s = Node(9,Node(5,Node(3,empty,empty),empty),Node(12,empty,empty)) ;;
height s ;;                          (* returns 3 *)
post_fold (fun a m -> m::a) [] s     (* returns [3;5;12;9] *)
```

8. (12 pts) Recursive Descent Parser in OCaml

The example OCaml recursive descent parser 15-parseArith_fact.ml employs a number of shortcuts. For instance, the function parseT handles the grammar rules for

T → F * T | F

directly instead of rewriting the grammar, creating the following productions:

T → ? C → ?

You must identify where code corresponding to parseC was inserted directly in the code for parseT, in the comments below:

```
let rec parseT lr =                          (* parseT *)
  let x = parseF lr in                       (* 1: ? → ? *)
  match !lr with                             (* parseC *)
    | ('*'::t) ->                            (* 2: if lookahead = First( ? ) *)
      lr := t;                               (* 3: ? → ? *)
      Prod (x,parseT lr)
    | _ -> x                                 (* 4: ? → ? *)
```

a. (2 pts) What rule should have been applied to the productions for T?
b. (2 pts) What productions for T & C would be created by applying the rule?
c. (2 pts) What production should appear in place of ? in comment 1?
d. (2 pts) What sentential form should appear in place of ? in comment 2?
e. (2 pts) What production should appear in place of ? in comment 3?
f. (2 pts) What production should appear in place of ? in comment 4?

9. (12 pts) Function arguments, scoping, & FP vs. OOP
   a. (4 pts) Explain why upwards funargs are more difficult to implement than downwards funargs.
   b. (4 pts) Explain why dynamic scoping can be more confusing than static scoping
   c. (4 pts) Describe how OOP may be used to simulate functional programming.

10. (12 pts) Parameter passing
    a. (2 pts) Describe lazy evaluation.
    b. (3 pts) Explain how to implement lazy evaluation in a language using call-by-value.

    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]);
}
```

    c. (2 pts) Give the output if C uses call-by-value
    d. (2 pts) Give the output if C uses call-by-reference
    e. (3 pts) Give the output if C uses call-by-name

11. (18 pts) Polymorphism
    a. (3 pts) Explain the difference between subtype polymorphism & overloading.
    b. (3 pts) Explain why subtype polymorphism causes problems for container classes (e.g., Set, Map, Stack).
    c. (4 pts) Generic programming is used in object-oriented programming languages to solve these problems. Name and describe the technique used to implement generic programming in Java without changing the JVM (Java virtual machine).

    Consider the following Java classes:

```
class A { … }
class B extends A { … }
class C extends A { … }
```

    HashSet<E> is a class supporting generics in the Java 1.5 class library.
    Explain why the following code is or is not legal

    d. (2 pts) int count(HashSet<?> s) { … } … count(new HashSet<C>( ));
    e. (3 pts) int count(HashSet<? extends A> s) { … } … count(new HashSet<C>( ));
    f. (3 pts) int count(HashSet<? super B> s) { … } … count(new HashSet<C>( ));

12. (16 pts) Java multithreading

a. Using Java locks & conditions, you must implement a synchronization construct called Partner. A Partner object is created with a value 0. When a thread calls the method next( ), it waits until a 2nd thread also calls next( ). The 2nd thread returns the current Partner value and continues without blocking, waking up the 1st thread. The next thread to invoke next( ) and acquire the lock for Partner will increment the Partner value and return its previous (non-incremented value). Be careful, since a 3rd thread may invoke next( ) before the 1st thread can wake up and be partnered with the 2nd thread, forcing the 1st thread to wait for a new partner!

```
class Partner {
        public int next( ) { … }
}

Partner p = new Partner( );
thread1.run( ) { … p.next( ); } // blocks & returns 0 after thread 2 calls next( )
thread2.run( ) { … p.next( ); } // next( ) returns 0 immediately
…
thread3.run( ) { … p.next( ); } // blocks & returns 1 after thread 4 calls next( )
thread4.run( ) { … p.next( ); } // next( ) returns 1 immediately
```

13. (6 pts) Memory allocation & garbage collection

Consider the following Java code.

```
public foo(Object a[ ] ) {
        a[0] = new Object( );   // object 1
        a[1] = new Object( );   // object 2
        a[2] = new Object( );   // object 3
        a[1] = a[2];
}
```

a. (3 pts) What object(s) are garbage when foo( ) returns? Explain why.

b. (3 pts) Describe why garbage collection may be preferred over manual memory allocation.

14. (8 pts) Markup & query languages

a. (5 pts) Creating your own XML tags, write an XML document that organizes the following information: Obama and Hillary are Democrats, McCain is a Republican.

b. (3 pts) Explain how query languages are different from programming languages.

15. (20 pts) Lambda calculus & encodings

Evaluate the following λ-expressions as much as possible

a. (3 pts) (λx.λy.x) a b
b. (3 pts) (λz.λy.z y) y z

Using the appropriate λ-calculus encodings

c. (6 pts) Given:

   or = λx.λy.((x true) y)
   true = λx.λy.x
   false = λx.λy.y

   Prove : false or true = true

d. (8 pts) Given:

   0 = λf.λy.y
   1 = λf.λy.f y
   2 = λf.λy.f (f y)
   M + N = λx.λy.(M x)((N x) y)

   Prove : (+ 0 2) = 2

16. (7 pts) Operational Semantics

a. (3 pts) Describe the purpose of analyzing the semantics of a program using techniques such as operational semantics.
b. (4 pts) Write a proof in operational semantics that "(+ a b)" has the value 3, if the value of "a" is 1 and the value of "b" is 2.

17. (16 pts) Extra credit

a. (10 pts) Given:

   Z = λf. (λx. f (λy. x x y)) (λx. f (λy. x x y))
   fact = λf. λn.if n = 0 then 1 else n * (f (n-1))

   What happens when you try to evaluate (Z fact) 1? You do not need to expand if.

b. (6 pts) Given

   Y = λf.(λx.f (x x)) (λx.f (x x))

   What is the relationship between Y and Z?