CMSC330 Spring 2010 Final Exam Solutions

1. (8 pts) Programming languages
   a. (4 pts) Briefly define a weak type system. Provide a code fragment example.

      Allows one type to be used as another type (or to be easily cast as another type). For instance, in C int can be used as bool in if statements "if (1) { … }".

   b. (4 pts) Recall that functional languages allow functions to be passed as arguments; e.g., map takes a function as an argument and applies it to elements of a list. Java does not directly support passing functions/methods as arguments. Briefly describe how you can encode function-passing in Java to implement something like map. (1 sentence should be sufficient).

      Pass to the function an object A implementing an interface with a known method B, then in the function invoke A.B( ).

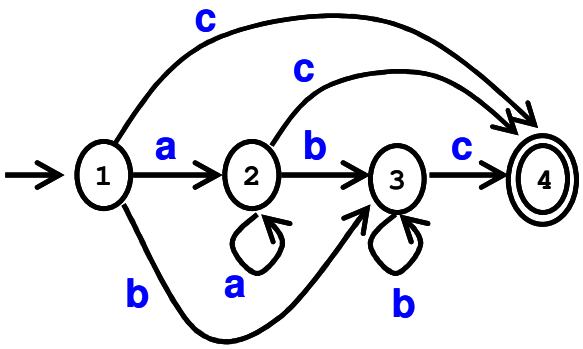
2. (6 pts) Ruby
   a. (4 pts) Name an important difference between Ruby's *nil* and Java's *null*.

      nil is an object, while null is not.
      nil can be treated as false, while null cannot.

   b. (2 pts) What is the output (if any) of the following Ruby program? Write FAIL if code does not execute.

```
a = { }                    # Output = nil
a[1] = "foo"
puts a[0]
```

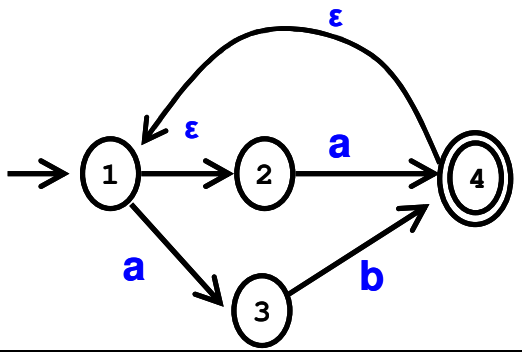
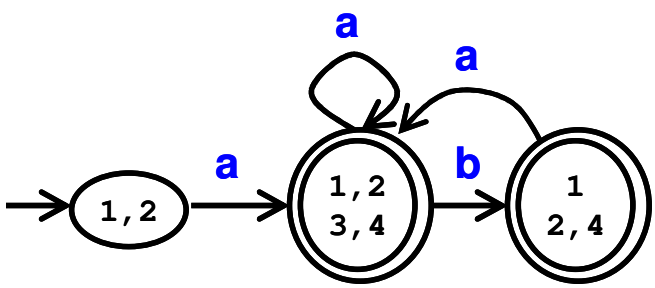
3. (18 pts) Regular expressions & finite automata
   a. (5 pts) Give a DFA that is equivalent to the regular expression a*b*c.



   b. (3 pts) Give a regular expression (formal REs or in Ruby) that accepts all 3-digit binary numbers. If you choose to use Ruby, you may not use the {3} feature.

      (0|1)(0|1)(0|1) or /[01][01][01]/ **/[01]{3}/ not allowed**

c. (10 pts) Convert the following NFA to a DFA using the subset construction algorithm. Be sure to label each state in the DFA with the corresponding state(s) in the NFA.

NFA	DFA resulting from subset construction
	

4. (6 pts) OCaml Types and Type Inference
   a. (2 pts) Write an OCaml expression with the following type
      int -> int -> int

      fun x y -> x+y

   b. (4 pts) Give the type of the following OCaml expression
      let f x y = y (x+1)

      int -> (int -> 'a) -> 'a

5. (6 pts) OCaml programming

   Write a function *convert* that takes as its argument a function of type ('a -> ('b -> 'c)) and returns a function of type ('a * 'b) -> 'c.

   Example:
   ```
   let add x y = x + y ;;
   let addPair = (convert add) ;;
   addPair (1,2) = 3
   ```

   let convert f (a,b) = f a b ;;
   let convert f = function (a,b) -> f a b ;;
   let convert = fun f (a,b) -> f a b ;;
   let convert f x = match x with (a,b) -> f a b ;;

6. (15 pts) OCaml polymorphic types & higher-order functions

Consider the OCaml type *shape* implementing a square and rectangular shapes:

```
type shape =
  Square of int              (* height *)
 | Rectangle of int * int    (* height & width *)
```

a. (5 pts) Write a function *area* of type (shape -> int) that takes a shape and returns its area. Recall the area of a square = height^2, rectangle = width*height

Examples:
area (Square 2) = 4
area (Rectangle (3,5)) = 15

```
let area x = match x with
      Square h -> h*h
      | Rectangle (h,w) -> h*w
```

b. (10 pts) Write a function *totalArea* that, given a list of shapes, produces the total area of all of the shapes. Your function should not use any recursion, but instead should use fold and/or map in combination with *area* and anonymous functions. Solutions using recursion and/or helper functions may receive partial credit.

Examples:
totalArea [Square 2] = 4
totalArea [Square 2 ; Rectangle (3,5)] = 19
totalArea [ ] = 0

```
let rec map f lst =  match lst with
   [] -> []
   | h::t -> (f h)::(map f t)
```

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

```
let totalArea ss = fold (fun a s -> a+s) 0 (map area ss)
```

7. (16 pts) Context free grammars & parsing

Consider the following grammar (where S = start symbol and terminals = **[**, **]**, **;**, **e**):

S → **[**A**]** | *epsilon*
A → S **;** A | **e**

a. (3 pts each) Indicate whether the following strings are generated by this grammar

i. **[e;e]** Yes **No** (circle one)
ii. **[[e];;e]** **Yes** No (circle one)

b. (4 pts) Compute First sets for S and A

First(S) = { [epsilon }
First(A) = { ; [e }

c. (6 pts) Write the function parse_A( ) used in a predictive, recursive descent parser for the grammar. You may assume parse_S( ), match( ) and the variable *lookahead* is provided.

```
parse_A( ) {
  if ((lookahead == '[') || (lookahead == ';')) {
    parse_S( );
    match(';');
    parse_A( );
  }
  else if (lookahead == 'e') {
    match('e');
  }
  else {
    error();
  }
```

8. (8 pts) Operational semantics

In an empty environment, to what value v will the expression (fun z = + z z) 6 evaluate to? In other words, find a v such that you can prove the following:

• ;(fun z = + z z) 6 → v

Use the operational semantics rules given in class, included here for your reference. Show the complete proof that stacks uses of these rules.

Number

$$\frac{}{\bullet;\ n \rightarrow n}$$

Lambda

$$\frac{}{A;\ \text{fun } x = E \rightarrow (A, \lambda x.E)}$$

Addition

$$\frac{A;\ E_1 \rightarrow n \qquad A;\ E_2 \rightarrow m}{A;\ +\ E_1\ E_2 \rightarrow n + m}$$

Function application

$$\frac{A;\ E_1 \rightarrow (A', \lambda x.E) \qquad A;\ E_2 \rightarrow v \qquad A, A', x{:}v;\ E \rightarrow v'}{A;\ (E_1\ \ E_2) \rightarrow v'}$$

Identifier

$$\frac{}{A;\ x \rightarrow A(x)}$$

$$\frac{\dfrac{\bullet\ ;\ (\text{fun } x = + x\ x) \rightarrow (\bullet, \lambda x.+\ x\ x) \qquad \bullet;\ 6 \rightarrow 6}{\bullet, x{:}6\ ;\ x \rightarrow 6} \qquad \dfrac{\bullet, x{:}6\ ;\ x \rightarrow 6 \qquad \bullet, x{:}6\ ;\ (+\ x\ x) \rightarrow 12}{}}{\bullet\ ;\ (\text{fun } x = + x\ x)\ 6 \rightarrow 12}$$

9. (6 pts) Lambda calculus

(3 pts each) Evaluate the following λ-expressions as much as possible

a. (λx.λy.x y) (λz.z) a

(λx.λy.x y) (λz.z) a → (λy. (λz.z) y) a → (λz.z) a → a
(λx.λy.x y) (λz.z) a → (λy. (λz.z) y) a → (λy.y) a → a

b. (λz.λy.z y z) y

(λz.λy.z y z) y → (λz.λx.z x z) y → λx.y x y

10. (9 pts) Lambda calculus encodings

a. (3 pts) Consider the function D = λx.x x. What happens when you apply D to itself ?

D D = (λx.x x) (λx.x x) → (λx.x x) (λx.x x) → D D
You get the same expression back (i.e., an infinite loop).

b. (6 pts) Consider the standard encodings for the booleans *true* and *false*. What does the following function Q compute? Hint: try passing two booleans for a & b, and see what you get from evaluating the function (if you do this, show your work clearly for partial credit).

Q = λa.λb.λx.λy.a x (b x y)

true = λx.λy.x
false = λx.λy.y

Q true true → λx.λy.true x (true x y) → λx.λy.true x x → λx.λy.x → true
Q true false → λx.λy.true x (false x y) → λx.λy.true x y → λx.λy.x → true
Q false true → λx.λy.false x (true x y) → λx.λy.false x x → λx.λy.x → true
Q false false → λx.λy.false x (false x y) → λx.λy.false x y → λx.λy.y → false

Thus Q = OR

11. (8 pts) Garbage collection

Consider the following Java code.

```
class Avatar {
  static Navi Tom, Norm, Jake;
  private void blockBuster( ) {
    Tom = new Navi(1);      // object 1
    Norm = new Navi(2);     // object 2
    Jake = new Navi(3);     // object 3
    Jake = Tom;
  }
```

a. (4 pts) What object(s) are garbage when blockBuster ( ) returns? Explain.

Object 3 is garbage since it is no longer reachable.

b. (4 pts) Briefly describe the difference between reachability and liveness in the context of garbage collection.

An object is *reachable* if there is some way for the program to access it, but is *live* only if it is actually accessed in the future. Garbage collection uses reachability to approximate liveness since the latter is much harder to determine.

12. (8 pts) Polymorphism

Consider the following Java classes:

```
class A { public void a( ) { … } }
class B extends A { public void b( ) { … } }
class C extends B { public void c( ) { … }}
```

(4 pts each) Explain why the following code is or is not legal

a. int count(Set<? extends A> s) { … } … count(new TreeSet<C>());

Legal. TreeSet<C> may be bound to s, since C extends A as indicated by the ? extends A wildcard.

b. int count(Set<? extends A> s) { for (C x : s) x.c( ); … }

Illegal, since S may be type Set<A> or Set<B>, elements of s are not guaranteed to be objects of class C.

13. (16 pts) Java multithreading

For this problem you may use either Java 1.4 or 1.5 synchronization.

Consider the following code for implementing a 1-place buffer:

```
public class Container {
  private Object buf = null;
  public void put(Object x) throws FullException {
    if (buf != null) throw new FullException();
    buf = x;
  }
```

```
  public Object get() throws EmptyException {
    if (buf == null) throw new EmptyException();
      Object x = buf;
      buf = null;
      return x;
  }
}
```

a. (3 pts) This code is only appropriate for single threaded applications. Give an example of what could go wrong if two threads use a Container implemented with this code at the same time.

For multithreaded applications, a data race between checking the value of buf (e.g., buf == null) and modifying the value of buf (e.g., buf = x) can cause the contents of buf to be corrupted.

b. (3 pts) This code is only appropriate for single-threaded applications. Add synchronization so that multiple threads may use this code at the same time. The semantics should be the same: if a thread tries to remove something from the buffer, but the buffer is empty, it will throw an exception. Likewise if the buffer is full, and a thread attempts to put something in, it will throw an exception, just as with the code now.

c. (10 pts) Add two methods to your amended Container class that implement variants of get() and put(), called take() and store(), which act as follows: if a thread calls take() the thread should return the contents of the buffer or wait until something is put there by another thread, and return that. Conversely, if a thread calls store() but something is already there, then it should wait until the buffer is empty. The methods take() and store() do not need to throw any exceptions.

Some helpful functions:

```
Object o;                  // any Object                          [Java 1.4]
o.wait()                   // sleeps until signaled
o.notifyAll()              // wakes up all threads sleeping on object

m = new ReentrantLock()    // returns new lock                    [Java 1.5]
c = m.newCondition( )      // returns conditional variable for lock
m.lock( )                  // acquires lock, block if another thread holds lock
m.unlock( )                // releases lock
c.await( )                 // sleeps until notified
c.signalAll( )             // wakes up all threads sleeping on condition var
```

Java 1.4	Java 1.5

public class Container {
  private Object buf = null;
  public synchronized void put(Object x)
  throws FullException {
  if (buf != null) throw new FullException();
  buf = x;
  }
  public synchronized Object get()
  throws EmptyException {
  if (buf == null) throw new EmptyException();
  Object x = buf;
  buf = null;
  return x;
  }
  public synchronized void store(Object x) {
  while (buf != null) wait();
  buf = x;
  notifyAll();
  }
  public synchronized Object take() {
  while (buf == null) wait();
  Object x = buf;
  buf = null;
  notifyAll();
  return x;
  }
}
	public class Container {
	  private Object buf = null;
	  private ReentrantLock l = new ReentrantLock();
	  private Condition c = l.newCondition();
	  public void put(Object x)
	  throws FullException {
	  l.lock();
	  if (buf != null) throw new FullException();
	  buf = x;
	  l.unlock();
	  }
	  public Object get()
	  throws EmptyException {
	  l.lock();
	  if (buf == null) throw new EmptyException();
	  Object x = buf;
	  buf = null;
	  l.unlock();
	  return x;
	  }
	  public void store(Object x) {
	  l.lock();
	  while (buf != null) c.await();
	  buf = x;
	  c.signalAll();
	  l.unlock();
	  }
	  public Object take() {
	  l.lock();
	  while (buf == null) c.await();
	  Object x = buf;
	  buf = null;
	  c.signalAll();
	  l.unlock();
	  return x;
	  }
	}