CMSC330 Fall 2013 Final Exam Solutions

1. (20 pts) Programming languages
   a. (4 pts) Explain why programming languages use types (e.g., int x, char y).
   To specify valid operations for a group of values, and catch errors when the value is used inappropriately.

   b. (4 pts) Explain why static types do not ensure a strong type system.
   Because the language may allow many implicit casts (e.g., C), or operations that are not type safe (e.g., printf(%d,1.0)).

   c. (4 pts) Explain why programming languages use scopes (e.g., {…}, begin … end).
   To limit the lifetime of a variable name (binding), so the name may be reused. Scopes also specify which variable a name refers to when shadowing.

   d. (4 pts) Explain when programming languages need to use closures.
   When functions are used as return values (upwards funargs), and the returned function *F* can access variables in a enclosing scope from a function that will have already returned by the time *F* is called.

   e. (4 pts) Explain why closures are similar to objects.
   Closures contain both code (methods) and an environment (data), and may be used to implement objects.

2. (16 pts) Ruby & OCaml
   What is the output (if any) of the following Ruby programs? Write FAIL if code does not execute.

   a. (2 pts)
   ```
   a = "Captain America"
   if a =~ /([a-z]+)/ then
      puts $1
   end
   ```
   # Output = aptain

   b. (2 pts)
   ```
   a = { }
   a["Iron"] = "Man"
   puts a["Iron"]
   ```
   # Output = Man

   c. (4 pts) Give the type of the following OCaml expression
   ```
   fun x -> fun y -> y::x
   ```
   Type = 'a list -> 'a -> 'a list

   d. (4 pts) Write an OCaml expression with the following type
   ('a -> 'a list) -> int list
   Code = fun x -> (fun y -> y::(x y)) ; [1]

   e. (4 pts) Give the value of the following OCaml expression. If an error exists, describe the error.
   ```
   fold (fun x -> fun y -> y::x) [ ] [2;3;4]
   ```
   Value = [4;3;2]

3. (9 pts) Scoping

Consider the following OCaml code.

```
let app f y = let x = 4 in (f y)+1 ;;
let proc x  = let change z = z-x in app change (x+3) ;;
(proc 2) ;;
```

Value computed by change z is z-x. With static scoping, x = 1st argument for proc. With dynamic scoping, x is redefinition of x=4 in app.

a. (3 pts) What value is returned by (proc 2) with static scoping? Explain.

4, since value of x in change is the formal parameter x in proc x (i.e., 2).
(proc x:2) -> app change (2+3=5) -> (change 5)+1 -> (5-2)+1 = 4

b. (6 pts) What value is returned by (proc 2) with dynamic scoping? Explain.

2, since value of x in change is 4, the redefinition of x=4 in app.
(proc x:2) -> app change (2+3=5) -> (change 5)+1 -> (5-4)+1 = 2

4. (12 pts) Parameter passing

Consider the following C code.

```
int i = 2;
void foo(int f, int g) {
  f  = f + g;
  g = g + 4;
}
int main( ) {
  int a[] = {1, 1, 1, 1};
  foo(i, a[i-2]);
  printf("%d %d %d %d %d\n", i, a[0], a[1], a[2], a[3]);
}
```

a. (2 pts) Give the output if C uses call-by-value

2 1 1 1 1 **// since i & a[0] are unchanged by calling foo**

b. (5 pts) Give the output if C uses call-by-reference

3 5 1 1 1 **// since i & a[0] are changed by calling foo**
// f = i, g = a[0] -> i = i+a[0] ; a[0] = a[0]+4

c. (5 pts) Give the output if C uses call-by-name

3 1 5 1 1 **// since i and a[i-2] are changed by calling foo**
// f = i, g = a[i-2] -> i = i+a[i-2] ; a[i-2] = a[i-2]+4

5. (10 pts) Lazy evaluation

Rewrite the following OCaml code using thunks so that foo uses lazy evaluation.

```
let foo x = x + 1;;
foo (2+3);;
```

let foo x = x() + 1;;
foo (fun () -> 2+3);;

6. (8 pts) Garbage collection

Consider the following Java code.

```
class AvengersMovie {
        static Avenger x, y, z;
        private void MoviePlot( ) {
                x = new Avenger ("Iron Man");          // object 1
                y = new Avenger ("Black Widow");       // object 2
                z = new Avenger ("Bruce Banner");      // object 3
                // transformation!
                z = new Avenger("Hulk");               // object 4
        }
}
```

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

Object 3 (Bruce Banner), since it can no longer be accessed.

b. (4 pts) List one advantage and one disadvantage of using garbage collection.

Advantages = less programmer effort, fewer memory errors

7. (12 pts) Polymorphism
   a. (4 pts) Explain why + in Java is an example of ad hoc polymorphism.

      + works only for a small number of argument types (e.g., int, float, String) and performs different actions (e.g., sum, concat) depending on the type.

   b. (4 pts) Briefly explain why Java generics is an example of parametric polymorphism.

      Java generics (e.g., HashMap<T>) allow a function/class to be applied to a range of different argument types.

   c. (4 pts) Briefly explain why "?" is used for Java generics, such as Set<?>.

      Wildcards such as ? are needed to support subtyping for containers.

8. (20 pts) Lambda calculus
   a. (4 pts) (λx.λy.y x y x) y x z

      (λx.λy.y x y x) y x z → (λx.λb.b x b x) y x z → (λb.b y b y) x z → (x y x y) z

   b. (6 pts) (λx.λy.λz.y z x) (λc.c) (λa.λb.b a) d

      (λx.λy.λz.y z x) (λc.c) (λa.λb.b a) d →
      (λy.λz.y z (λc.c)) (λa.λb.b a) d →
      (λz.(λa.λb.b a) z (λc.c)) d →
      THEN ONE OF
      (λa.λb.b a) d (λc.c) → (λb.b d) (λc.c) → (λc.c) d → d
      OR
      (λz.(λb.b z) (λc.c)) d → (λb.b d) (λc.c) → (λc.c) d → d
      OR
      (λz.(λb.b z) (λc.c)) d → (λz.(λc.c) z) d → (λz.z) d → d
      OR
      (λz.(λb.b z) (λc.c)) d → (λz.(λc.c) z) d → (λc.c) d → d

   Lambda calculus encodings
   c. (10 pts) Using encodings, show succ 3 =>* 4. Show each beta-reduction.
      =>* indicates 0 or more steps of beta-reduction

      succ = λz.λf.λy.f (z f y)
      0 = λf.λy.y
      1 = λf.λy.f y
      2 = λf.λy.f (f y)
      3 = λf.λy.f (f (f y))
      4 = λf.λy.f (f (f (f y)))

      (λz.λf.λy.f (z f y)) 3 →
      λf.λy.f (3 f y) →
      λf.λy.f ((λf.λy.f (f (f y))) f y) →
      λf.λy.f ((λy.f (f (f y))) y) →
      λf.λy.f (f (f (f y))) →
      4

9. (15 pts) Multithreading

```
class Buffer {
  Buffer ( ) {
    Object obj = Object.new
    Object buf = null;
    boolean empty = true;
  }
```

```
  void produce(o) {
    synchronize (obj) {
1.    if (!empty) wait( );
2.    empty = false;
3.    notifyAll( );
4.    buf = o;
    }
  }
```

```
  Object consume( ) {
    synchronize (obj) {
5.    if (empty) wait( );
6.    empty = true;
7.    notifyAll( );
8.    return buf;
    }
  }
}
```

Consider the multithreaded Java 1.4 code above. Assume there are multiple producer & consumer threads being executed in the program, but just a *single* Buffer object. If we freeze program execution at some point in time, each thread *T* will have just finished executing some statement *S* (i.e., statement *S* will have been the last statement executed by thread *T*). We wish to examine the state of these threads.

a. (4 pts) Is it possible given two threads x and y for the last statement executed by thread x to be statement 2 and thread y to be statement 7 in the code above? Explain your answer.

No, since if the last statement executed by a thread is either statement 2 or 7, it must be holding the lock. No other thread can acquire the lock and reach statement 2 or 7.

b. (5 pts) Is it possible given two threads x and y for the last statement executed by thread x to be statement 5 and thread y to be statement 3 in the code above? Explain your answer.

Yes, since thread x may have reached statement 5 and released the lock after calling wait, so thread y may acquire the lock and reach & execute statement 3.

c. (6 pts) Is it possible in the code above for two threads calling consume( ) to get the same object o passed to produce(o)? Explain your answer.

Yes, since the wait at line 5 is not called in a while loop. It is possible for a thread at line 5 to be woken and execute lines 6-8 even though empty is still true. This will cause the thread to return the buffer value already returned by a different consumer.

10. (30 pts) Ruby multithreading

```
require "monitor"

 def goTrain(me)
      10.times {
       r = getRoom()
       $locks[r].synchronize {
         if $rooms[r].empty?
           $rooms[r].push(me)
           $conds[r].wait_until { $rooms[r].empty? } # woken by partner
         else
           partner = $rooms[r].pop
           sleep 0.01  # train!
           puts "Room #{r} training #{me} & #{partner}"
           $conds[r].broadcast
         end
       }
      }
 end

 def simulate(m,n)
   $nHeroes = m;
   $nRooms = n;
   $rooms = []
   $locks = []
   $conds = []

   # prepare training rooms
   $nRooms.times { |i|
      $rooms[i] = []
      $locks[i] = Monitor.new
      $conds[i] = $locks[i].new_cond
   }

   # start hero training
   threads = []
   $nHeroes.times { |me|
     t = Thread.new { goTrain(me) }
     threads.push(t)
   }
   threads.each { |t| t.join }

 end
```

11. (20 pts) Prolog

Given the following clauses, list all answers returned by the following queries.

```
avenger(ironman).
avenger(thor).
sibling(thor,loki).
asgardian(thor).
asgardian(X) :- sibling(Y,X),asgardian(Y).
train1(X,Y) :- avenger(X),!,avenger(Y).
train2(X,Y) :- avenger(X),X\=Y,avenger(Y).
```

a. (2 pts) ?- avenger(thor).
true.

b. (2 pts) ?- asgardian(A).
A=thor;
A=loki.

c. (2 pts) ?- train1(A,B).
A=ironman,
B=ironman;
A=ironman,
B=thor.

d. (3 pts) ?- train2(A,thor).
A=ironman.

e. (3 pts) ?- train2(thor,A).
false.

```
foo([H,H|T],H).
foo([_|T],R) :- foo(T,R).
```

f. (2 pts) ?- foo([1,2],A).
false.

g. (3 pts) ?- foo([2,2],A).
A=2.

h. (3 pts) ?- foo([1,2,2,3,4,4],A)
A=2;
A=4.

12. (28 pts) Prolog programming

Write a prolog function findDups(A,X,Y) that given a list A, looks for adjacent duplicate values X starting at index Y, where the first element has index 0. If there are multiple duplicates, backtracking should return additional answers, starting from left to right in the list. findDups( ) fails if there are no duplicates in A. You may use the operators =, \=, \+, is, +, -, [H|T], and !. You do not need to worry about efficiency.

Examples:

?- findDups([],X,Y). false. ?- findDups([1],X,Y). false.	?- findDups([1,2],X,Y). false. ?- findDups([1,1],X,Y). X=1, Y=0.	?- findDups([0,1,1,2,2],X,Y). X=1, Y=1; X=2, Y=3.

find([H,H|T],I,H,I).
find([_|T],N,R,I) :- N1 is N+1, find(T,N1,R,I).
findDup(L,X,Y) :- find(L,0,X,Y).