Fall, 2008 Name: ______________________

COP 4020 — Programming Languages I

Test on the Declarative Model

Special Directions for this Test

This test has 10 questions and pages numbered 1 through 6.

This test is open book and notes.

If you need more space, use the back of a page. Note when you do that on the front.

Before you begin, please take a moment to look over the entire test so that you can budget your time.

Clarity is important; if your programs are sloppy and hard to read, you may lose some points. Correct syntax also makes a difference for programming questions.

When you write Oz code on this test, you may use anything we have seen in chapters 1–2 of our textbook. But unless specifically directed, you should not use imperative features (such as cells) or the library functions `IsDet` and `IsFree`. Problems relating to the kernel syntax can only use features of the kernel language.

You are encouraged to define functions or procedures not specifically asked for if they are useful to your programming; however, if they are not in the Oz base environment, then you must write them into your test.

For Grading

Problem	Points	Score
1	5	
2	15	
3	5	
4	10	
5	15	
6	5	
7	10	
8	10	
9	15	
10	10	

The first three problems ask for sets of free or bound variable identifiers that occur bound in the statement above. Write the entire requested set in brackets. For example, write $\{V, W\}$, or if the requested set is empty, write $\{\}$.

1. Consider the following Oz statement in the kernel language.

```
local A in
   {F A B}
end
```

   (a) (3 points) [Concepts] Write the entire set of the variable identifiers that occur free in the statement above.

   (b) (2 points) [Concepts] Write the entire set of the variable identifiers that occur bound in the statement above.

2. Consider the following Oz statement in the kernel language.

```
Q = proc {$ BT P X ?R}
       case BT of
          btree(val:N left:C right:D) then
          local Temp in
             {Add P N Temp}
             local CR in
                {Q C Temp Y CR}
                local Z in
                   {Q D CR Y R}
                end
             end
          end
       else R = N
       end
    end
```

   (a) (6 points) [Concepts] Write the entire set of the variable identifiers that occur free in the statement above.

   (b) (9 points) [Concepts] Write the entire set of the variable identifiers that occur bound in the statement above.

3. [Concepts] Consider the following Java method declaration.

```
public void bake(int a) {
    s = sift(a);
    cook(s, 350);
}
```

(a) (3 points) Write below, in set brackets, the entire set of variable identifiers that occur free in the Java code above.

(b) (2 points) Write below, in set brackets, the entire set of variable identifiers that occur bound in the Java code above.

4. Consider the following Oz code.

```
local Sum in
   Sum = proc {$ N Acc ?R}
            if N >= 0
            then {Sum N-1 Acc+N R}
            else R = Acc
            end
         end
   local Res in
      {Sum 5 0 Res}
      {Browse Res}
   end
end
```

(a) (2 points) [Concepts] When the above code runs, what output, if any, appears in the browser?

(b) (5 points) [Concepts] Does the closure formed for the **`proc`** value expression on lines 2–7 include an environment that has a binding for `Sum`? If so, briefly explain why, if not, then say why such a binding is not needed.

(c) (3 points) [Concepts] Is the call statement `{Sum N-1 Acc+N R}` in the declarative kernel language? If not, briefly explain why it is not.

5. (15 points) [Concepts] Desugar the following Oz code into kernel syntax by expanding all syntactic sugars. (Assume that the identifier `Z`, and the function identifiers `Dec` and `Minus` are declared elsewhere.)

```
fun {Dec N} {Minus N 1} end
Z={Dec 3}
```

6. (5 points) [Concepts] Which of the following correctly states the connection between static scoping and closures in Oz. (Circle the letter of the correct statement.)

   (a) Closures contain an environment, which allows free variable identifiers appearing in the body of a procedure to refer to the location (store variable) associated with the most recent declaration of those identifiers that is still active.

   (b) Closures contain an environment, which allows bound variable identifiers appearing in the body of a procedure to refer to the location (store variable) associated with the most recent decaration of those identifiers that is still active.

   (c) Closures contain an environment, which allows bound variable identifiers appearing in the body of a procedure to refer to the location (store variable) associated with the closest textually surrounding decaration of those identifiers.

   (d) Closures contain an environment, which allows free variable identifiers appearing in the body of a procedure to refer to the location (store variable) associated with the closest textually surrounding decaration of those identifiers.

7. (10 points) [Concepts] What happens when the following code executes in Oz? Briefly explain why that happens.

```
local Swap X Y in
   X = 7
   Y = 5
   Swap = proc {$ A B}
             local Temp in
                Temp = A
                A = B
                B = Temp
             end
          end
   {Swap X Y}
   {Browse 'X is '#X}
end
```

8. (10 points) [Concepts] What is the output, if any, of the following code in Oz? Briefly explain why that output appears.

```
local P HM C in
   P = district(state: fl county: orange rep: dem(congr))
   C = seminole
   case P of
      place(state: S county: C rep: dem(D)) then {Browse first#S#C#D}
   [] district(state: S county: C) then {Browse second#S#C}
   [] district(state: S county: C rep: R) then {Browse third#S#C#R}
   [] district(state: S county: C rep: dem(D)) then {Browse fourth#S#C#D}
   else {Browse none(C)}
   end
end
```

9. (15 points) [Concepts] Give an example of a syntactic sugar or linguistic abstraction in Java, C, C++, or C#, stating clearly: (a) the language, and (b) what desugars into what (e.g., what statement desugars into what other statement), and (c) giving a rule or a concrete example that explains the desugaring.

10. (10 points) [Concepts] What happens when the following program executes in Oz? Briefly explain your answer.

```
local X in
   try
      try
         raise first(X) end
         X = 8
      catch first(Y) then
         X = 9
         {Browse caughtFirst(xval: X yval: Y)}
         raise second(Y) end
      finally
         {Browse third(X)}
      end
   catch second(Z) then
      {Browse caughtSecond(xval: X zval: Z)}
   end
end
```