Spring, 2012 Name: _____

(Please *don't* write your id number!)

COP 4020 — Programming Languages I

Test on the Declarative Model

Special Directions for this Test

This test has 14 questions and pages numbered 1 through 8.
This test is open book and notes, but no electronics.
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 (given in Tables 2.1 and 2.2 of the textbook).
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

Question:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Total
Points:	6	4	10	10	10	4	3	10	5	7	7	7	5	12	100
Score:															

1. (6 points) [Concepts] This is a question about the semantics of Oz's dataflow variables. Consider each of the following pieces of code separately as an independent program. Circle the letter of each program that will suspend when run. There may be more than one correct answer, so circle the letter of each that may suspend (and no others).

A.
```
local X in
   local Y in
      {Browse done}
   end
end
```

B.
```
local X in
   local Y in
      {Browse X+Y}
      {Browse done}
   end
end
```

C.
```
local X in
   local Y in
      X = Y
      {Browse done}
   end
end
```

D.
```
local X in
   local Y in
      X = Y
      X = 4020
      {Browse X+Y}
      {Browse done}
   end
end
```

E.
```
local X in
   local Y in
      X = 4020
      {Browse X+Y}
      {Browse done}
   end
end
```

2. (4 points) What happens when the following program is run in Oz? (Circle the correct answer.)

```
local A in
   local B in
      A = B
      B = 4020
      A = 99 + 0
      {Browse result(a: A b: B)}
   end
end
```

   A. The browser shows `result(a: 99 b: 4020)`, because `A` was assigned 99 and `B` was assigned 4020.

   B. The browser shows `result(a: 99 b: 99)`, because 99 is the last value assigned to `A` and both `A` and `B` are in the same equivalence class.

   C. The browser shows `result(a: 4020 b: 4020)`, because dataflow variables can only be assigned once.

   D. The program has a failure (a "tell error"), because dataflow variables can only be assigned one value.

   E. The program doesn't compile, because `Browse` is a free variable identifier.

3. (10 points) [Concepts] Desugar the following Oz statement into the declarative kernel of Oz. (You are supposed to know what that means.)

```
R = {Add 3 4}
```

4. (10 points) [Concepts] Desugar the following Oz statement into the declarative kernel of Oz. (You are supposed to know what that means.) Note: you can use `Number.'+'` in your answer.

```
local L = (3+X)|nil in
   R = L
end
```

5. (10 points) [Concepts] Desugar the following Oz statement into the declarative kernel of Oz.

```
fun {TwoList Fst Snd}
   [Fst Snd]
end
```

6. (4 points) [Concepts] Which of the following is correct? (Circle the correct answer.)

   A. When a program P is properly desugared, the desugaring of P has no free variable identifier occurrences.

   B. When a program P is properly desugared, the desugaring of P has the same set of free variable identifier occurrences as P has.

   C. When a program P is properly desugared, the desugaring of P has no bound variable identifier occurrences.

   D. When a program P is properly desugared, the desugaring of P is usually much shorter and easier to read than the P was.

7. (3 points) [Concepts] Suppose you are looking at a piece of code written in your favorite programming language. You notice that in that piece of code, that there are some free variable identifier occurrences. Circle the correct statement about this situation below.

   A. The code you are looking at could be fine, if the free variable identifiers that occur in it are declared and initialized elsewhere.

   B. The code you are looking at is certainly in error, because code should never have free variable identifier occurrences in it.

   C. The code you are looking at is unusual, because it is very rare for free identifiers to occur in code.

8. (10 points) [Concepts] Consider the following Oz code.

```
declare
proc {CaseExample X}
   case X of
      burrito(meat: M sauce: S weight: N) then {Browse one(M S N)}
   [] taco(meat: M sauce: S) then {Browse two(M S)}
   [] taco(meat: M weight: N) then {Browse three(M N)}
   [] taco(meat: M sauce: S weight: N) then {Browse four(M S N)}
   [] taco(meat: M sauce: S weight: N) then {Browse five(M S N)}
   [] taco(meat: M sauce: S weight: N cheese: C) then {Browse six(M S N C)}
   else {Browse nomatch}
   end
end

local V in
   V = taco(meat: chicken sauce: hot weight: 4)
   {CaseExample V}
end
```

   What value, if any, is shown in the browser when this code executes?

9. (5 points) [Concepts] Consider the following Oz code.

```
local Ls T1 T2 C E1 A E2 B E3 in
   thread Ls = E1|T1 end
   thread T1 = E2|T2 end
   thread T2 = E3|nil end
   thread C = A+B end
   thread E1 = A*10 end
   thread A = 7 end
   thread E2 = B+0 end
   thread B = 5 end
   thread E3 = C*100 end
   {Wait Ls} {Wait T1} {Wait T2} % wait for Ls, T1, and T2 to be determined
   {Browse Ls}
end
```

Circle the letter of the true statement below.

A. This code does not compile, because `Wait` and `Browse` are free variable identifier occurrences in the code.

B. When this code executes, it suspends indefinitely, because when it tries to run `A+B`, `A*10`, and `B+0`, the store variables that `A` and `B` denote are not determined.

C. When this code executes, it shows the list `[70 5 1200]` in the browser, and no other result is possible.

D. When this code executes, it shows either the list `[70 5 1200]` or the list `[10 0 100]` in the browser.

E. When this code executes, it might have many different results, depending on which thread finishes before the other. With concurrent programs like this, it is impossible to predict the outcome.

The next 3 problems ask for sets of free or bound variable identifiers that occur in a program written in the Oz declarative kernel language. For these problems, write the entire requested set in brackets. For example, write $\{V, W\}$, or if the requested set is empty, write $\{\}$.

Also, recall that **`declare`** is *not* in the declarative kernel, so you should *not* imagine an implicit **`declare`** in the examples given for these problems.

10. Consider the following Oz statement in the declarative kernel language.

```
local X in
   local Y in
      local Z in
         X = Y
         X = 3
         {Add X Y Z}
         {Browse Z}
      end
   end
end
```

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

(b) (5 points) [Concepts] Write the entire set of the variable identifiers that occur bound in the statement above.

11. Consider the following statement in the declarative kernel language.

```
local One in
   One = single
   F = proc {$ X Y ?R}
          {Smash X One R}
       end
   local Res in
      local Z in
         {F One One Res}
         {Show Res}
      end
   end
end
```

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

(b) (4 points) [Concepts] Write the entire set of the variable identifiers that occur bound in the statement above.

12. Consider the following statement in the declarative kernel language.

```
case It of
   book(title: T author: A publisher: P) then {Append A T R}
else case It of
        cd(company: C recordTitle: RT) then R = RT
     else case It of
             movie(name: N producer: Prod company: Co) then R = N
          else R = none
          end
     end
end
```

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

(b) (4 points) [Concepts] Write the entire set of the variable identifiers that occur bound in the statement above.

13. (5 points) [Concepts] This is a question about static scoping and closures. Circle the letter of the correct answer.

A. Closures are used with dynamic scoping to make sure that each bound variable identifier refers to a value of the proper type.

B. Closures are used with dynamic scoping to provide the bound variable identifiers in procedure or function values with runtime values.

C. Closures are used with static scoping to ensure that each free variable identifier occurrence in a procedure or function value denotes the closest textually-surrounding declaration of that identifier.

D. Closures are used with static scoping to ensure that each bound variable identifier occurrence in a procedure or function value denotes a determined value.

14. [Concepts] Consider the following Oz code.

```
local R in
   local V in
      V = router(make: netgear model: 22 value: 100)
      case V of
         router(make: C model: M value: V) then {IsRecord V R}
      else R = elsepart
      end
   end
   {Browse R}
end
```

Recall that `IsRecord` tests whether its first argument is a record. For example, `{IsRecord 10 X}` puts **`false`** into `X`, but `{IsRecord hub(make: sony) X}` puts **`true`** into `X`.

(a) (5 points) What, if anything, is shown in the Browser after executing the above code?

(b) (7 points) Briefly explain why the output (if any) of the above code is the way it is.