COP4020 Fall 2002 – Final Exam

Name: ______________________________ (Please print)

Put the answers on these sheets. Use additional sheets when necessary. Show how you derived your answer when applicable (this is required for full credit and helpful for partial credit). You can collect 100 points in total for this exam.

1. Consider the following *Algol 60 loop*:

   ```
   for i := 10 step -3 until 1 do ...
   ```

   Which of the following loops are Algol 60 loops that are equivalent to this loop? (mark **one or more** (4 points)

   (a) `for i := 10, 7, 4, 1 do ...`

   (b) `for (i=10; i>0; i-=3) do ...`

   (c) `for i := 10, i-3 while i > 0 do ...`

   (d) `for i := 10, 7, 11-i while i > 0 do ...`

2. Consider the following C construct:

   ```
   while (n > 0)
   { ... }
   ```

   What sort of programming construct is this? (mark **one**) (4 points)

   (a) an enumeration controlled loop

   (b) a logically controlled pre-test loop

   (c) a logically controlled post-test loop

   (d) a logically controlled mid-test loop

3. What purpose serves the `finally` clause in Java? (mark **one**) (4 points)

   (a) To define a final catch-all exception handler

   (b) To execute code for clean-up purposes regardless of whether or not a thrown exception is caught in a method.

   (c) To declare a final class method.

   (d) To declare a method that cannot raise exceptions.

4. Which of the following terms can be successfully unified in Prolog? (mark **one or more**) (4 points)

   (a) `a = b`

   (b) `a = X`

   (c) `foo(bar) = foo(Y)`

   (d) `foo(bar) = bar(foo)`

5. Name the seven major categories of control-flow ordering constructs. (7 points)

6. Give an example of a *tail-recursive function* in Scheme, C or any other language. (4 points)

7. Explain the difference between facts, rules, and queries in Prolog. (4 points)

8. What is an alias? Give an example of an alias. (4 points)

9. Consider the following Pascal program:

```
program scopes(input, output)
    procedure P1(A1 : integer)
        var X : integer
        procedure P2(A2 : integer)
            var Y : integer
        begin (* body of P2 *)
            ... <== (*)
        end;
        procedure P3(A3 : integer)
            var X : integer;
        begin (* body of P3 *)
            P2(X)
        end
    begin (* body of P1 *)
        P3(X)
    end
begin (* body of main program *)
    P1(0)
end.
```

(a) What is the *reference environment* at the location in the program indicated by `<== (*)`? That is, give the variables, arguments, and procedures that are visible (in scope) at this location. (4 points)

(b) The main program calls, `P1`, `P1` calls `P3`, and `P3` calls `P2`. Draw the stack layout of the subroutine stack after these calls. Show the subroutine frames (without their details) with the static links. (4 points)

(c) Draw the specific subroutine frame layout of procedure `P1`. Name each of the slots in this frame and indicate for each of the slots what it is used for. (4 points)

10. Consider the following program:

```
var z : integer; /* global variable */
procedure addto(x, y)
    begin
        z := 1;
        y := y + x
    end
begin /* body of main program */
    z := 2;
    addto(z, z);
    write_integer(z)
end
```

For each of the parameter passing modes shown in the table below show the value printed by the program. (8 points)

	By value	By reference	By value/result
Output:			

The parameter passing mode is applicable to both parameters of `addto`.

11. Consider the following Java program:

```
class TooMuchInput extends Exception
{   public TooMuchInput()
    {    System.out.println("Too much input");
    }
}
class Example
{   private DataInputStream in;
    private int read() throws IOException
    {   return Integer.parseInt(in.readLine());
    }
    public int run() throws __________________________
    {   int sum = 0;
        try
        {   in = new DataInputStream(System.in);
            for (int i = 0; i < 100; i++)
                sum += read();
            throw new TooMuchInput();
        }
        catch (IOException)
        {
        }
        return sum;
    }
}
```

(a) `Example.run()` reads a list of integers from standard input and returns the sum of the values. Mark the point in the program at which an `IOException` can be raised due to an unsuccessful read operation. (4 points)

(b) List the exception(s) that can be raised by method `run` at the blank line in the program. (4 points)