COP4020 Fall 2006 – Final Exam

Name: _____ (Please print)
Put the answers on these sheets. You can collect 100 points in total for this exam.

1. Consider the following Ada program fragment:

```
search: loop
  i := i+1;
  exit search when a[i]<0;
  j := j-1;
  exit search when b[j]<a[i];
end loop search;
```

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

   (a) a nondeterministic statement
   (b) an enumeration-controlled loop
   (c) a logically-controlled mid-test loop
   (d) a selection statement

2. What are *named parameters*? (mark **one**) (4 points)

   (a) they allow formal parameter names to be explicitly bound to actual parameter values in subroutine calls
   (b) parameters that are used in `catch` handlers of exceptions in C++
   (c) parameters that adopt parameter passing by name
   (d) parameters with in-out modes in Ada

3. Consider the following Prolog definitions:

```
wishes(andy, [ball,bike,rocket]).
wishes(buzz, [rocket,laser]).
wishes(mary, [doll,ball,scarf]).
share_toy(T) :- wishes(X,A), wishes(Y,B), \+(X=Y), member(T,A), member(T,B).
```

   Which of the following queries are successful? (mark **one or more**) (4 points)

   (a) `share_toy(X)`
   (b) `share_toy(doll)`
   (c) `share_toy(ball)`
   (d) `share_toy(buzz)`

4. What is the formula that determines the number of iterations of the Fortran loop:

```
DO I=A,B,S
  ...
END DO
```

where `A` is the start value, `B` the end value, and `S` the step size? (mark **one**) (4 points)

(a) $B - A + 1$
(b) $\max(\lfloor \frac{B-A+S}{S} \rfloor, 0)$
(c) $\max(B - A + S, 0)$
(d) $S * (B - A)$

5. *In-mode* parameters in Ada are implemented by (mark **one or more**) (4 points)

(a) call by value
(b) call by reference
(c) call by result
(d) call by name

6. Consider the following pseudo code program fragment:

```
subroutine swap(a,b)
begin
  t := a;
  a := b;
  b := t;
end
```

Suppose that the programming language adopts the *reference model* for variables (similar to Java) and thus the parameters are *passed by sharing*. What is the result after calling `swap(x,y)` on variables `x` and `y`? (mark **one**) (4 points)

(a) the values of `x` and `y` are interchanged
(b) the values of `x` and `y` are unchanged
(c) the value of `x` is set to `y` and `y` is unchanged.
(d) the values of `x` and `y` are set to the value of `t`

7. What happens when a dynamically allocated object's lifetime exceeds its binding lifetime indefinitely (until program termination)? (mark **one**) (4 points)

(a) produces dangling references to the object
(b) gives a runtime error
(c) introduces a memory leak
(d) the object is still in scope

8. After executing the code in a `catch` handler in C++ when an exception occurred, where does the program normally continue? (mark **one**) (4 points)

   (a) in the caller of the current function (current function exits)

   (b) the statements that immediately follow the `throw` statement in the `try` block

   (c) the statements that immediately follow the last `catch` clause

   (d) the statements in another `catch` clause for the same exception

9. Show the typical layout of a stack-allocated subroutine frame, label the frame's slots, and describe the content of each slot. (7 points)

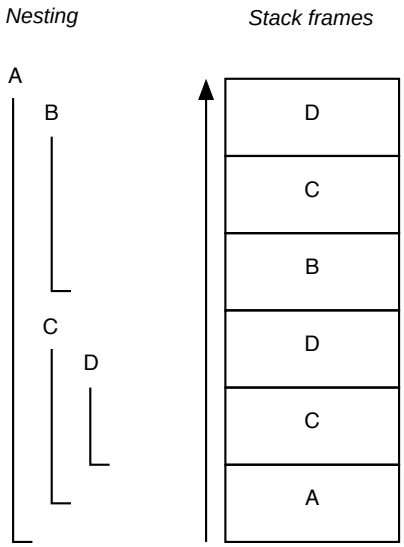
10. Iterators in Java and C++ are called *iterator objects*. Other programming languages, notably Clu, Python, Ruby, and C#, implement "true iterators". Explain the difference. (7 points)

11. *Tail-recursive functions*:

    (a) Give an example of a tail-recursive function. (4 points)

    (b) Optimize this function using *tail-recursion optimization*. (4 points)

12. Consider the following diagram depicting subroutine nesting levels and the stack frames after a sequence of subroutine calls:



Draw the static links. Suppose subroutine D now accesses a variable declared in A. How many static links have to be traversed to access this variable? (6 points)

13. What is *backward chaining* and how is it used as a resolution strategy in Prolog? (8 points)

14. Consider the program:

```
procedure P(n:real, k:integer)
   var x:real;
   procedure Q(m:integer)
      var n:real;
      procedure R(k:real)
         var x:real;
      begin (* begin of R *)
         ...
      end (* end of R *)
   begin (* begin of Q *)
      ... <== (*)
   end (* end of Q *)
begin (* begin of P *)
   ...
end (* end of P *)
```

What is the reference environment at the location marked `(*)` in this program? List the subroutines that can be called at `(*)` and the variables and arguments that can be accessed (for variables and arguments, provide answers in the form "`n` of `Q`")? (8 points)

15. Consider the following pseudo-code program:

```
x : integer /* global */
y : integer /* global */
procedure one(P : procedure)
   x : integer
   x := y
   y := 2
   P()
procedure two
   x := x+y
procedure three
   write_integer(x)
begin /* main program */
   x := 0
   y := 1
   one(two)
   three()
end /* main program */
```

Assuming static scoping or dynamic scoping rules are used (with shallow or deep bindings). What value does the program print? (8 points)

	Static Scoping	Dynamic Scoping with Shallow Binding	Dynamic Scoping with Deep Binding
Output:			

16. What is the value printed by the following pseudo-code program for each of the four parameter passing modes shown in the table? (8 points)

```
a : integer /* global variable */
b : integer /* global variable */
procedure p(x : integer, y : integer)
    a := y;
    x := x + y;
begin /* main program */
    a := 2
    b := 1
    p(a, b)
    write_integer(a)
end /* main program */
```

	By value	By reference	By value/result	By name
Output:				

17. Consider the following outline of a Queue class in C++ that includes exception handling.

```
class EmptyQueue {};
class QueueOverflow {};
class OutOfMemory {};

class Queue
{ public:
    Queue Queue()
    { ...
      if (...) throw OutOfMemory();
    };
    int pop()
    { if (...)
        return ...;
      else throw EmptyQueue();
    };
    void push(int elt)
    { if (...)
        ...
      else throw QueueOverflow();
    }
  private:
    ... // other Queue stuff
};
void sample()
{ Queue q();  // <----- (1)
  try {
    ...
    q.pop();  // <----- (2)
    ...
    q.push(); // <----- (3)
    ...
  } catch(EmptyQueue)
  { ... // <--- Handler (A)
  } catch(OutOfMemory)
  { ... // <--- Handler (B)
  }
}
void main()
{ try{
    sample();
    ... // <----------- (4)
  } catch(QueueOverflow)
  { ... // <--- Handler (C)
  } catch(OutOfMemory)
  { ... // <--- Handler (D)
  }
  ... // <------------- (5)
}
```

There are three types of exceptions associated with the Queue class. The Queue class methods can only raise the exceptions shown in the program outline above. Suppose an exception is raised by one of the three statements `(1)`, `(2)`, and `(3)`. Indicate in the table below which handlers are executed (`A`–`D`) and where the execution continues (`1-5`)? (8 points)

Exception caused by	Handlers executed and continuation point
`(1)`	
`(2)`	
`(3)`	