1. Consider the following Ada program fragment:

```ada
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:

```prolog
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:

\[
\text{DO I=A,B,S} \quad \ldots \quad \text{END DO}
\]

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

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

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:

```plaintext
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:

![Diagram of subroutine nesting and stack frames]

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:

\[
\begin{align*}
x &: \text{integer} /* \text{global} */ \\
y &: \text{integer} /* \text{global} */ \\
\text{procedure one}(P : \text{procedure}) & \quad \text{x : integer} \\
& \quad \text{x} := y \\
& \quad \text{y} := 2 \\
& \quad P() \\
\text{procedure two} & \quad \text{x} := x+y \\
\text{procedure three} & \quad \text{write}\_\text{integer}(x) \\
\text{begin} /* \text{main program} */ & \quad \text{x} := 0 \\
& \quad \text{y} := 1 \\
& \quad \text{one(two)} \\
& \quad \text{three()} \\
\text{end} /* \text{main program} */
\end{align*}
\]

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

<table>
<thead>
<tr>
<th></th>
<th>Static Scoping</th>
<th>Dynamic Scoping with Shallow Binding</th>
<th>Dynamic Scoping with Deep Binding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output:</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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)

\[
\begin{align*}
a &: \text{integer} /* \text{global variable} */ \\
b &: \text{integer} /* \text{global variable} */ \\
\text{procedure p}(x : \text{integer}, y : \text{integer}) & \quad a := y; \\
& \quad x := x + y; \\
\text{begin} /* \text{main program} */ & \quad a := 2 \\
& \quad b := 1 \\
& \quad p(a, b) \\
& \quad \text{write}\_\text{integer}(a) \\
\text{end} /* \text{main program} */
\end{align*}
\]

<table>
<thead>
<tr>
<th></th>
<th>By value</th>
<th>By reference</th>
<th>By value/result</th>
<th>By name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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()
    {
        ... // Constructor
    }
    int pop()
    {
        if (condition)
        {
            return ...
        }
        else throw EmptyQueue();
    }
    void push(int elt)
    {
        if (condition)
        {
            ...
        }
        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)

<table>
<thead>
<tr>
<th>Exception caused by</th>
<th>Handlers executed and continuation point</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td></td>
</tr>
<tr>
<td>(2)</td>
<td></td>
</tr>
<tr>
<td>(3)</td>
<td></td>
</tr>
</tbody>
</table>