1. Which of the following optimizations is considered a peephole optimization? (mark one) (4 points)
   (a) Common-subexpression elimination
   (b) Code motion
   (c) Branch chaining
   (d) Register assignment

2. Subroutine frames (activation records) manage a procedure’s local data. For a typical programming language implementation, who deallocates the frame? (mark one) (4 points)
   (a) Caller
   (b) Callee
   (c) Caller and callee
   (d) None of the above

3. List three local optimizations of your choice and describe what they optimize. (9 points)
4. Consider the following program:

```plaintext
program P()
    var p : integer;
    procedure Q(k : integer)
        begin
            R(k, p)
        end
    procedure R(i : integer, j : integer);
        var n : integer
    procedure T(i : integer)
        begin
            ... (* body of T *)
        end;
    procedure S()
        var m : integer
        begin
            T(n)
        end;
        begin
            S()
        end;
    begin
        Q(p)
    end
```

(a) Program P calls Q, Q in turn calls R, R in turn calls S, and S in turn calls T. Draw the resulting stack layout with activation records. Show the arguments and local variables in each record and draw the access links. (10 points)
(b) Which variables are visible (in scope) in the body of \( T \) and how many access links must be traversed to reach the nonlocal data? (6 points)

<table>
<thead>
<tr>
<th>Var</th>
<th>visible (Y/N)</th>
<th>#links</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td></td>
<td></td>
</tr>
<tr>
<td>j</td>
<td></td>
<td></td>
</tr>
<tr>
<td>k</td>
<td></td>
<td></td>
</tr>
<tr>
<td>m</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n</td>
<td></td>
<td></td>
</tr>
<tr>
<td>p</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5. Partition the following fragment of three-address code into basic blocks and construct the CFG. (9 points)

```
if n>0 goto L1
    goto L3
L1:  i := 0
     f := 1
L2:  i := i + 1
     f := f * i
     if i<n goto L2
    goto L4
L3:  f := 0
L4:  halt
```
6. Apply *global constant propagation* to the following CFG. (8 points)
7. Apply register allocation and assignment using graph coloring to the following CFG:

\[
\begin{align*}
&x := \text{read()} \\
&t := \text{read()} \\
&t > 0 ? \\
&x := x + 1 \\
&y := 0 \\
&y := t \\
&t := 0 \\
&p := t + y \\
&p := p + x \\
&x, y, t \text{ are not live}
\end{align*}
\]

Hint: show the live ranges of the variables in the CFG, assuming that \( x \), \( y \), and \( t \) are no longer live at the exit from the CFG. Draw the register-interference graph (conflict graph) for the variables and determine the minimum number of colors to color the graph. (12 points)
8. Consider the following CFG:

(a) Draw the dominator tree of the CFG. (8 points)
(b) Identify the natural loops. (5 points)
(c) Is the CFG reducible? Explain why or why not. (5 points)
9. Give the data-flow equations for reaching definitions (gen, kill, in and out sets) as described in the book and illustrated in class for the four programming constructs (assignment, statement composition, if-then-else, and do-while). (10 points)
10. Consider the following program:

```plaintext
if n>0 then
    d1:  i := 0;
    d2:  f := 1;
    d3:    i := i + 1;
    d4:    f := f * i
        while i < n;
    else
    d5:  f := 0;
end if
```

Annotate the syntax tree of the program with in, gen, kill, and out bit-vectors:

Hint: compute the gen and kill vectors bottom-up first, i.e. start at the leaves. Then compute the in and out vectors in a top-down, left-to-right traversal. (in is inherited, while out is synthesized.) (10 points)