1. For the basic block:

\[
\begin{align*}
q &= 3 \\
r &= 10 \\
s &= q + r \\
t &= 2r + s \\
t &= q \\
u &= q + r \\
v &= q + t \\
w &= 3 + x
\end{align*}
\]

State for each of the basic blocks on the following page which optimization was performed on the above:

- Constant Propagation/Folding
- Copy Propagation
- Common Subexpression Elimination
- Dead Code Elimination.
(a) \[ \begin{align*}
q &= 3 \\
r &= 10 \\
s &= q + r \\
t1 &= s \\
t &= 2*r+s \\
t &= q \\
u &= t1 \\
v &= q + t \\
w &= 3 + x
\end{align*} \]

(b) \[ \begin{align*}
q &= 3 \\
r &= 10 \\
s &= q + r \\
t &= 2*r+s \\
t &= q \\
u &= q + r \\
v &= q + q \\
w &= 3 + x
\end{align*} \]

(c) \[ \begin{align*}
q &= 3 \\
r &= 10 \\
s &= 13 \\
t &= 33 \\
t &= 3 \\
u &= 13 \\
v &= 36 \\
w &= 3 + x
\end{align*} \]

(d) \[ \begin{align*}
q &= 3 \\
r &= 10 \\
s &= q + r \\
t &= q \\
u &= q + r \\
v &= q + t \\
w &= 3 + x
\end{align*} \]
2. In class we discussed *available expression* dataflow analysis. Recall that an expression $e$ is available at point $p$ if:

- Every path from the initial node to $p$ evaluates $e$ before reaching $p$, and
- There are no assignments to any operand of $e$ after evaluation but before $p$.

In the table below, fill in the final values of IN obtained after performing available expression analysis on the CFG of Figure 1 (next page). A '1' should indicate the expression is available on entry to the block.

<table>
<thead>
<tr>
<th></th>
<th>a + b</th>
<th>c * d</th>
<th>e / f</th>
</tr>
</thead>
<tbody>
<tr>
<td>B1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>B2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B7</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Figure 1: CFG for problem 2.
3. Recall from lecture that a variable $v$ is live at point $p$ if:

- $v$ is used along some path starting at $p$, and
- There is no definition of $v$ along $p$ before its use.

In the table below, fill in the final values of OUT obtained after performing liveness analysis on the CFG of Figure 2 (next page). A '1' should indicate the variable is live on exit from the block. Assume all variables are visible outside the procedure.

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>B1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B7</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>


Figure 2: CFG for problem 3.
4. A compiler hacker writes an analysis to compute values of integer variables in a program. The hacker’s analysis maintains a set for each variable at each program point, the set contains the possible values for that variable. The hacker uses set union to combine values at the control-flow join points.

The hacker tests the analysis on several acyclic control flow graphs and it is shipped in the compiler. One of the customers tries to compile a program that contains a loop, and the analysis fails to terminate. What is the problem?

Describe the changes that the compiler hacker must make to fix the analysis.