Consider the following code:

1: READ(x);
2: READ(y);
L1 3: if (x > 9) goto L4
4: if (y > 3) goto L2
5: x = 3 + x;
6: b = y + x;
7: goto L3
L2 8: y = 3 + x;
9: b = y + x;
L3 10: y = x + y;
11: goto L1;
L4 12: WRITE(b)
13: halt

1. Draw the CFG for this piece of code.
   We’ll “draw” the CFG by giving the predecessor(s) and successor(s) for each statement in the program.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Predecessor</th>
<th>Successor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>—</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2, 11</td>
<td>4, 12</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>5, 8</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>7, 9</td>
<td>11</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>12</td>
<td>3</td>
<td>13</td>
</tr>
<tr>
<td>13</td>
<td>12</td>
<td>—</td>
</tr>
</tbody>
</table>

2. Show the results of running a *reaching definitions* analysis on this code. For each line of code, show what definitions reach that line. Assume this is the only code in the program.

   We will represent a definition by [v, n], meaning variable v was defined at line n. For each statement, we will show the GEN and KILL sets, with [v, *] meaning that all definitions of x are killed.
Reaching definitions is a forward analysis that uses \( \cup \) to merge information. That means that the two dataflow equations we will use to compute IN and OUT sets for each statement are:

\[
IN(s) = \bigcup_{t \in \text{pred}(s)} OUT(t)
\]

\[
OUT(s) = (IN(s) - KILL(s)) \cup GEN(s)
\]

We will iterate these equations, updating every statements IN and OUT sets, until the values stop changing. When we’re done, we get:

3. Show the results of running a liveness analysis on this code. For each line of code,
show what variables are live \textit{out} for that line (i.e., what variables are live immediately after that line would execute)

As before, we start by constructing the GEN and KILL sets for each statement:

<table>
<thead>
<tr>
<th>Statement</th>
<th>Predecessor</th>
<th>Successor</th>
<th>GEN</th>
<th>KILL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>—</td>
<td>2</td>
<td>—</td>
<td>x</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>—</td>
<td>y</td>
</tr>
<tr>
<td>3</td>
<td>2, 11</td>
<td>4, 12</td>
<td>x</td>
<td>—</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>5, 8</td>
<td>y</td>
<td>—</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>6</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>7</td>
<td>y, x</td>
<td>b</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>10</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>9</td>
<td>x</td>
<td>y</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>10</td>
<td>x, y</td>
<td>b</td>
</tr>
<tr>
<td>10</td>
<td>7, 9</td>
<td>11</td>
<td>x, y</td>
<td>y</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>3</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>12</td>
<td>3</td>
<td>13</td>
<td>b</td>
<td>—</td>
</tr>
<tr>
<td>13</td>
<td>12</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

Liveness is a \textit{backwards} analysis that uses $\cup$ to merge together information, so the equations for IN and OUT are:

\[
\begin{align*}
IN(s) &= (OUT(s) - KILL(s)) \cup GEN(s) \\
OUT(s) &= \bigcup_{t\in\text{succ}(s)} IN(t)
\end{align*}
\]

When we iterate these equations, we get:

<table>
<thead>
<tr>
<th>Statement</th>
<th>IN</th>
<th>OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>b</td>
<td>b, x</td>
</tr>
<tr>
<td>2</td>
<td>b, x</td>
<td>b, x, y</td>
</tr>
<tr>
<td>3</td>
<td>b, x, y</td>
<td>b, x, y</td>
</tr>
<tr>
<td>4</td>
<td>x, y</td>
<td>x, y</td>
</tr>
<tr>
<td>5</td>
<td>x, y</td>
<td>x, y</td>
</tr>
<tr>
<td>6</td>
<td>x, y</td>
<td>b, x, y</td>
</tr>
<tr>
<td>7</td>
<td>b, x, y</td>
<td>b, x, y</td>
</tr>
<tr>
<td>8</td>
<td>x</td>
<td>x, y</td>
</tr>
<tr>
<td>9</td>
<td>x, y</td>
<td>b, x, y</td>
</tr>
<tr>
<td>10</td>
<td>b, x, y</td>
<td>b, x, y</td>
</tr>
<tr>
<td>11</td>
<td>b, x, y</td>
<td>b, x, y</td>
</tr>
<tr>
<td>12</td>
<td>b</td>
<td>—</td>
</tr>
<tr>
<td>13</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>