CSE 401 - Dataflow Extravaganza

1. Consider the following small program that we used as a dataflow example for live variable analysis. This time all of the statements are labeled and we want to compute reaching definitions.

```
L0:   a = 0
L1:   b = a + 1
L2:   c = c + b
L3:   a = b * 2
L4:   if a < N goto L1
L5:   return c
```

The *reaching definitions* dataflow problem is to determine for each variable definition which other blocks in the control flow graph could potentially see the value of the variable that was assigned in that definition. To simplify things, we will treat each individual statement above as a separate block, and use the statement labels as the names of both the blocks and the definitions in them. So, for example, reaching definition analysis would allow us to determine that definition L0, which assigns to a, can reach block L1.

A definition *d* in block *p reaches* block *q* if there is *at least* one path from *p* to *q* along which definition *d* is not killed.

This can be set up as a dataflow problem as follows: For each block *b* in the control flow graph, define GEN(*b*) to be the set of definitions generated in that block and not subsequently killed in the block, and KILL(*b*) to be the definitions killed by that block. These sets can be computed once, statically, for each block. If block *b* contains *d*: x = a op b,
Then GEN(*b*) contains *d*, provided that *d* is not killed later in block *b*. KILL(*b*) contains all other definitions *d*' elsewhere in the program that define the same variable x.

Given the GEN and KILL sets for the blocks, we can compute the IN and OUT sets of definitions that reach each block as follows:

$$IN(b) = \bigcup_{p \in pred(b)} OUT(p)$$

$$OUT(b) = GEN(b) \cup (IN(b) - KILL(b))$$

For this problem, compute the reaching definitions for the blocks in the given program (treating each statement as a separate block). You should give a table with a row for each block showing the GEN and KILL sets for that block, and then compute IN and OUT sets using successive iterations until you there are no further changes to any IN or OUT set.

Note that this is a forward dataflow problem so the answer will converge faster if you start computing with L0, L1, …

Block	GEN	KILL	IN (1)	OUT (1)	IN (2)	OUT (2)
L0						
L1						
L2						
L3						
L4						
L5						

2. Very Busy

Rearrange the expressions in the blocks (i.e. moving expressions from one block to another) to apply “Very Busy” expressions. In other words, where is the earliest I can compute each expression without affecting the outcome?