ECE 468
Problem Set 7: Dataflow analysis

1. Show the results of running a *liveness* analysis on the following piece of code: For each line of code, show which definitions reach that line of code by indicating the line number the definition occurred in.

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

2. Show the results of running an *available expression* analysis on the code, by indicating which expressions are available at each instruction.

3. In this problem, your goal is to develop a dataflow analysis to find *leaking* values. A common error in programs is forgetting to free a variable that has been `malloc`ed. Write an analysis that can tell, at the *end* of a function, whether any allocations during the function *may not have been freed* by the end of the function. Hint: think about this analysis' relationship to reaching definitions.

   (a) This is a forward bitvector analysis, where each object of interest in the analysis takes on the value 0 or 1. a) What are the objects of interest in this analysis? b) What does it mean for an object to have value 0? c) What does it mean for the object to have value 1?

   (b) Which statements GEN information for this analysis? Which statements KILL information? Give the GEN and KILL sets for the relevant statements.

(c) Define IN and OUT for this analysis (don't forget which direction your analysis is running: if you're running forward, IN should be defined in terms of a statement's predecessors, and OUT should be defined in terms of IN. If you're running backwards, OUT is defined in terms of a statement's successors, and IN is defined in terms of OUT). Don't forget to think about how your analysis should behave at merge statements.

(d) (Tricky question): this problem can be reformulated as a backwards analysis, instead. At each `malloc`, we want to know whether the variable being allocated here will definitely be freed. Answer the previous three questions for this new analysis. Which of the two analyses will give you better results (or will they be the same)?