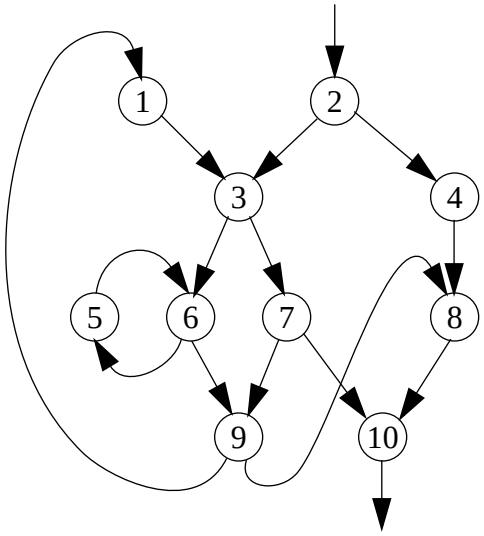
Homework Assignment #1

Due Monday 1/22, at the start of lecture

(For all homeworks, do not look at any previous sample solutions for any previous 501 homework assignments.)

1. For the following control flow graph, for the purposes of a forward data flow analysis (such as available expressions), show the dominator tree, identify any loops, and for each loop, identify the loop head node and the loop entry, exit, and back edges. Is this graph reducible?



2. Repeat problem 1 for purposes of a backwards data flow analysis (such as live variables).

3. Build the DAG representation of the following basic block (where variables whose names start with t are assumed not to be referenced outside of the basic block). You should perform common subexpression elimination as part of this process, exploiting commutativity of operators. You should mark which nodes in the DAG are stored in which user variables at the end of the block.

```
t1 := x * 5
x  := x + 1
y  := x
t2 := x * 5
y  := 5 * y
```

4. Why can we not simply apply this same DAG-construction algorithm over the whole procedure body, across basic blocks?

5. Consider the case where a forward analysis's flow functions for each kind of statement are represented in terms of constant KILL and GEN bit-vectors. Thus, each flow function has the form *out* = *in* - KILL + GEN, with KILL and GEN being constants independent of *in* or *out*. The combined effect of a whole basic block of instructions can be calculated by applying each instruction's flow function in turn, starting with the in bit-vector at the start of the basic block, and computing the out vector at the end of the basic block. I.e., for instruction *i* of the basic

block, $out_i = in_i$ - $KILL_i$ + GEN_i, and for all instructions other than the first, $in_i = out_{i-1}$. However, in this special case where KILL and GEN do not depend on *in*, it is possible to compute a single summary flow function with constant KILL and GEN bit-vectors, of the form $out_n = in_1$ - SUMMARY-KILL + SUMMARY-GEN (where in_1 and out_n are the points before and after the basic block, respectively).

Give a pair of simple inductive equations for computing the $SUMMARY\text{-}KILL_i$ and $SUMMARY\text{-}GEN_i$ bit-vectors, representing the summary KILL and GEN bit-vectors for the first *i* instructions of a basic block, in terms of the $SUMMARY\text{-}KILL_{i-1}$ and $SUMMARY\text{-}GEN_{i-1}$ bit-vectors (if $i>1$) and the $KILL_i$ and GEN_i bit-vectors for the flow function of the *i*th instruction, $1 \le i \le n$. SUMMARY-KILL and SUMMARY-GEN for the whole basic block are then $SUMMARY\text{-}KILL_n$ and $SUMMARY\text{-}GEN_n$, respectively.

What is the chief advantage of this SUMMARY flow function over the original flow functions?

6. Define a data flow analysis that can be used to compute two sets of variables for each procedure: those which are definitely not defined before use, and those which may not be defined before use; these sets can be used to provide extra error checking of programs. Strive for the highest quality information while maintaining safety. Use a lattice-theoretic formalism to define your analysis. Argue that your analysis terminates. Explain how to use the information computed by your analysis to construct the two sets specified above for output to the user.