CS 6340 – Spring 2013 – Assignment 2 **Name** Sam Britt
Assigned: January 9, 2013
Due: January 16, 2013 **Name**_____

At the beginning of class on the due date, submit your neatly presented solution with this page stapled to the front (50 points).

NOTE: All work on this problem set is to be done with your partner and without solutions from other past or current students. Any violations will be dealt with according to the Georgia Tech Academic Honor Code and according to the College of Computing process for resolving academic honor code violations. All work must be done using some document creation tool. In addition, graphs must be drawn with a graph-drawing tool—no hand-drawn graphs will be accepted. We'll discuss this requirement more in class.

1. Given the following C program, construct control flow graphs (CFGs) for the program (make sure you understand the semantics of the *break* and *continue* statements in C). In the first CFG, use maximal basic blocks; in the second CFG, place each statement in its own basic block. In both cases, label the nodes in the graphs with the numbers of the program statements (10).

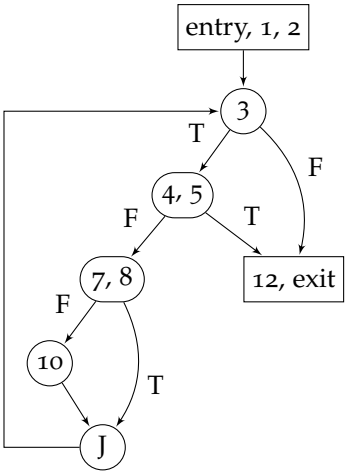
```
    main()
    {
       int sum, i, j;

1.     sum = 0;
2.     i = 1;
3.     while (i <= 5) {
4.         scanf("%d",&j);
5.         if (j < 0)
6.               break;
7.         sum = sum + j;
8.         if (sum > 10)
9.               continue;
10.        i = i + 1;
11.    }
12.    printf("sum is %d", sum);
    }
```

2. For the CFG you created in (1)
   a. Compute and show the dominator and postdominator trees for the graph (10).
   b. Use T1-T2 analysis to determine whether the graph is reducible; show all your work (10).

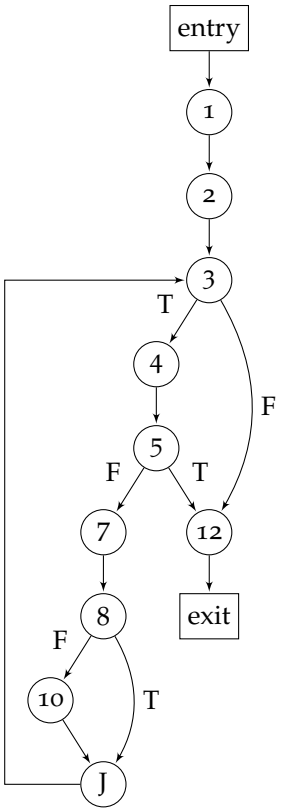
Sam Britt
Jan. 19, 2013

CS 6340
Assignment 2

1. The CFGs are below. A join node, J, has been added for clarity.

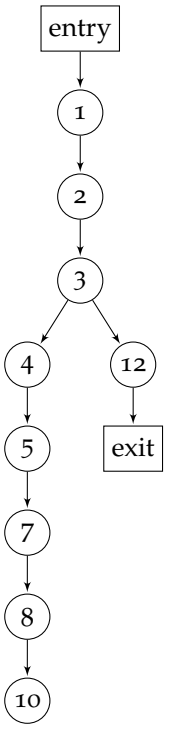


CFG using maximal basic blocks.

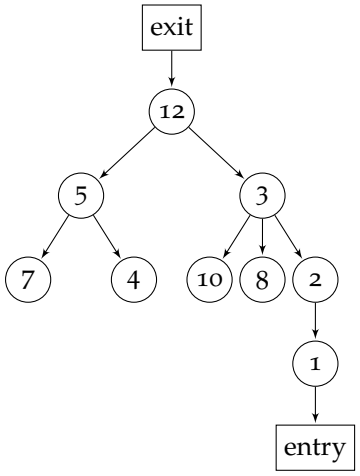


CFG using one statement per block.

2. (a) The dominator and postdominator trees are below.

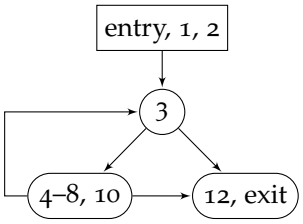


Dominator tree.

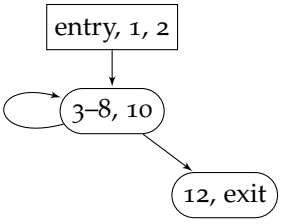


Postdominator tree.

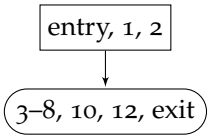
(b) Yes, the graph is reducible. First, many nodes have a single predecessor, so T2 analysis combines many of the nodes:



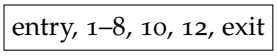
Again applying a T2 transformation:



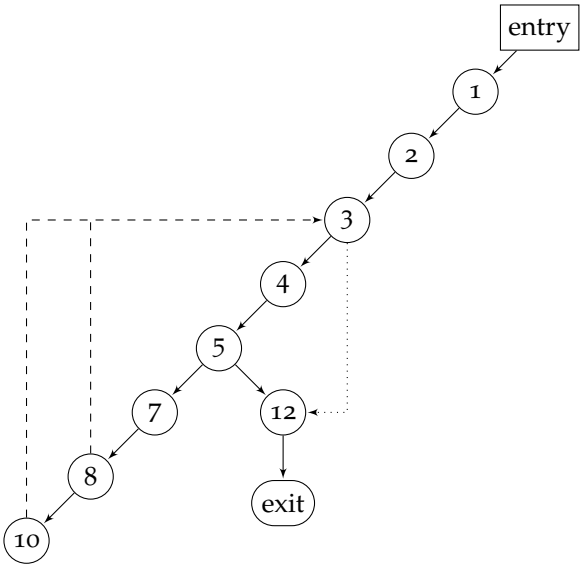
Applying a T1 transformation to get rid of the self loop and a T2 transformation to collapse the node containing the exit statement leaves:



And finally, a T1 transformation collapses this flow graph to a single node, thus showing that the original CFG is indeed reducible.

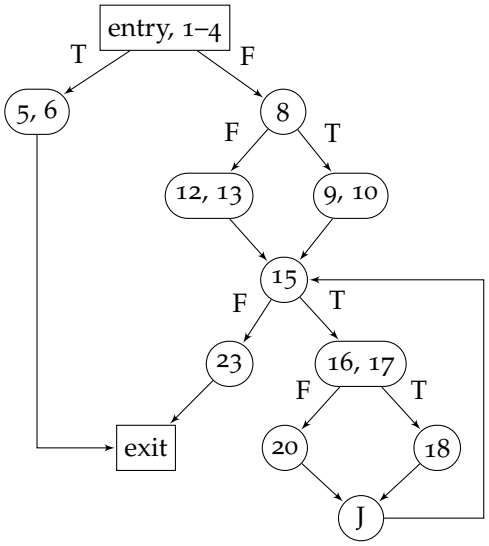


(c) One version of the dept-first presentation is below. Retreating edges are marked by dashes (- - →) and forward edges are marked by dots (·····→). The depth of this graph is 1. The paths 10 → 3 or 8 → 3 are both acyclic paths, each containing one reverse edge. Another path with more reverse edges cannot be found. (In fact, because both back edges have the same head node, they are in the same loop.)

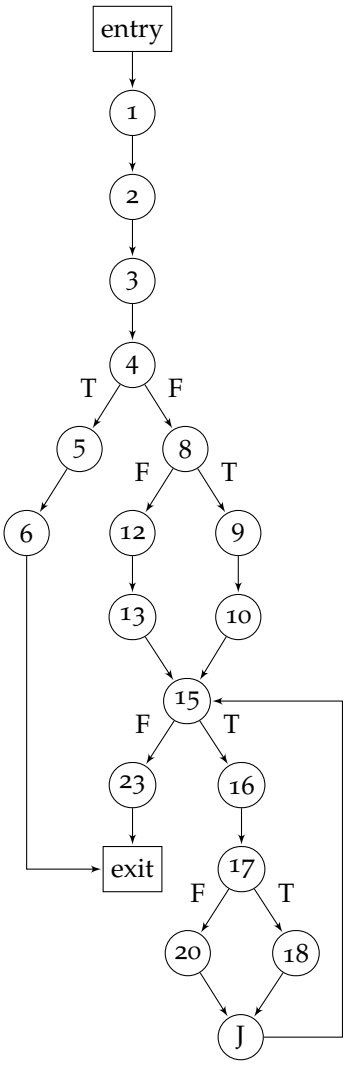


Dept-first presentation.

3. The CFGs are below. A join node, J, has been added for clarity.



CFG using maximal basic blocks.



CFG using one statement per block.