CS 243 Midterm Examination

Winter 2011-2012

This is an open-book, open-notes exam. You cannot use the computer.

You have 1 hour 15 minutes to work on this exam. The examination has 5 questions worth 70 points. There is a final bonus question, worth 5 points, which will NOT be curved. Please budget your time accordingly. Write your answers in the space provided on the exam. If you use additional scratch paper, please turn that in as well.

Your Name: _____

In accordance with both the letter and spirit of the Honor Code, I have neither given nor received assistance on this examination.

Signature: _____

Problem	Points	Score
1	15	
2	5	
3	10	
4	15	
5	25	
6	5	
Total	75	

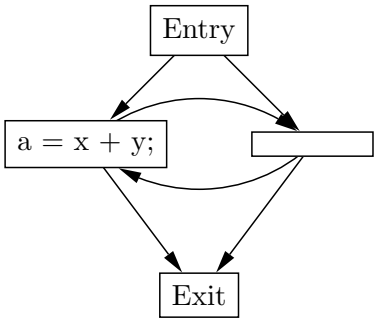
1. (15 points) True of false. Explain your answer. No credit is given unless the explanation is correct.

   (a) Consider the DOM relation, where a DOM b is read "a node, a, dominates a node, b". If a DOM c and b DOM c, then either a DOM b or b DOM a.

   Solution: True. This is in the textbook.

   (b) Lazy code motion can move expressions out of a cycle in a control flow graph, even if it is not part of a natural loop.

   Solution: True. Consider:



   (c) Lazy code motion should be run before constant propagation.

   Solution: We accepted both answers depending on the quality of the analysis.

   (d) Given a monotone dataflow framework, an iterative algorithm will converge to a fixed point solution even if all the interior program points of a program are initialized to bottom.

   Solution: We accepted both answers. If the framework can include infinite descending chains, the answer is false. If not, the answer is true.

   (e) Given a monotone dataflow framework, if an input flow graph has no cycles, an iterative algorithm will produce the maximum fixed point solution even if all the interior program points are initialized to bottom.

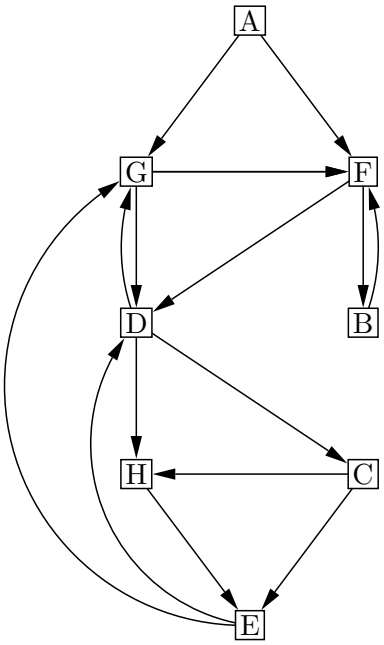
   Solution: True.

2. (5 points) Suppose you have a register interference graph on a machine with 5 registers, where every node in the graph has 6 edges. Is this graph impossible to color (with 5 colors)?

   Either provide an argument that such a graph cannot have a 5-coloring, or give an example of such a graph that does have a 5-coloring.

   Solution: Such a graph can exist. Consider a complete bipartite graph with 6 white nodes and 6 black nodes.

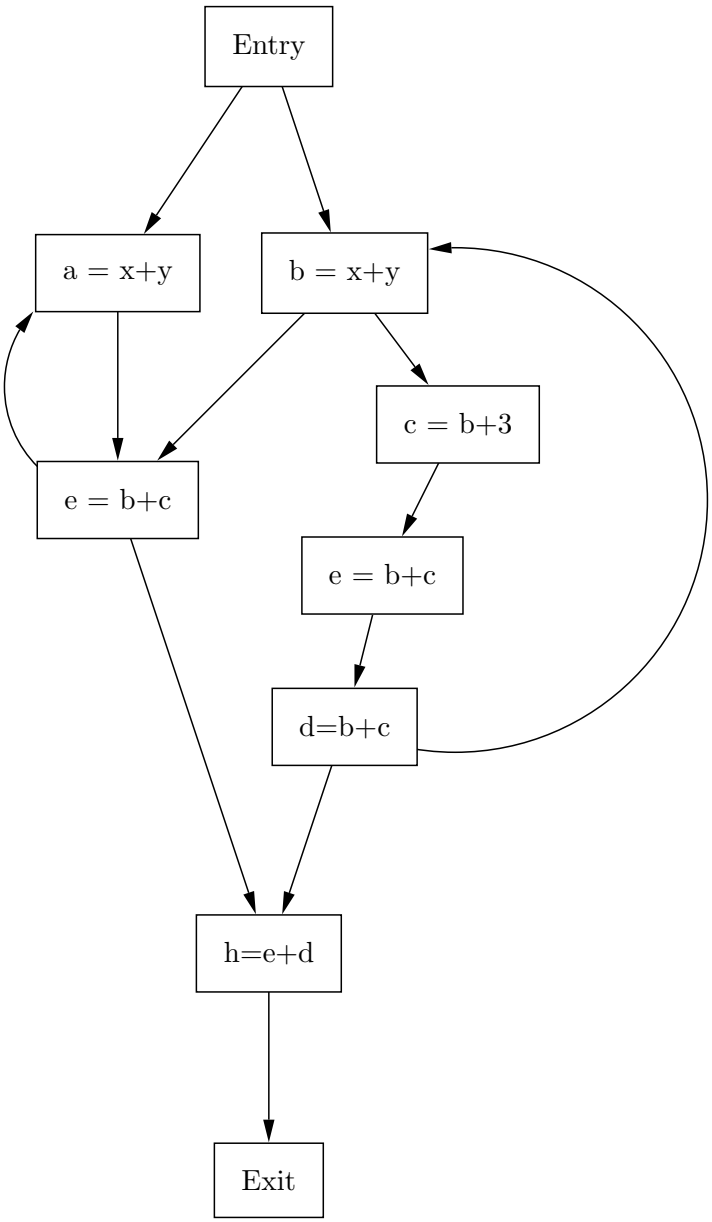
3. (10 points) Consider the following flow graph (A is the start node):



   (a) Give all the natural loops of this graph.
   Solution: {F,B} and {D,H,C,E}.

   (b) Is this graph reducible? If yes, explain. If no, give a minimal set of edges that should be added or removed to make it reducible.
   This graph is not reducible. Cutting either AF or FD will make it reducible.

4. (15 points) Show the result of applying partial redundancy elimination to the following program. Just show the result of the optimization — it is not necessary to show intermediate steps.



Solution: Introduce expressions "u = b + c" and "t = x + y". Insert both before "a = x + y". Insert t before "b = x + y". Insert u between "b = x + y" and "e = b + c". Rewrite all uses of the expressions "x + y" and "b + c" with t and u, respectively.

5. (25 points) Define a static program analysis that eliminates dead code from a program. Assume a programming language with simple integer variables and the following statements:

   - input functions: input(x),
   - output functions: output(x),
   - x1 = x2;
   - x1 = x2 + x3;
   - if (x) goto L;

   You cannot eliminate any input, output, or conditional branch instructions. Any assignments and additions that do not contribute to the computation of the variables used in output and conditional statements should be eliminated.

   Given the following program:

```
        input(a);
        b = a;
        c = a;
    L1: c = b + c;
        input (a);
        if (a) goto L1;
        d = a;
        e = a + d;
        if (e) goto L2;
        f = 1;
    L2: output (e);
```

   The following code should be generated:

```
        input(a);
    L1: input(a);
        if (a) goto L1;
        d = a;
        e = a + d:
        if (e) goto L2;
    L2: output (e);
```

   Specify your data flow algorithm fully by giving concise answers in the third column of the following table: (Note that you will not receive full credit if the algorithm does not necessarily converge). For the sake of simplicity, treat each statement as a separate basic block. **Please state clearly how you eliminate the dead code after the data flow analysis**.

   Solution:

a	Direction	Backward
b	Set of values in the semi-lattice Sets of variables	Explain what the values mean Variables that will be used in the computation of a branch or output statement
c	Semi-lattice diagram The set of subsets of variables in the program, arranged in a semi-lattice by set-inclusion	Label the top and the bottom elements Top: the empty set. Bottom: the universal set.
d	Transfer function of a basic block	 Given an input set of variables, keep everything the same but what is indicated below: "input(x)": Change nothing "output(x)": Add x to the set. "x1 = x2": Add x2 to the set if x1 is in the set. Remove x1 from the set if it is present. "x1 = x2 + x3": Add x2 and x3 to the set if x1 is in the set. "if(x) goto L": Add x to the set.
e	Boundary condition	Assignment to: out[entry] or in[exit] in[exit] = top
f	Initialization for the iterative algorithm	 Initialize all interior nodes to top
g	Is the dataflow framework monotone?	Yes or no. No explanation is necessary. Yes
h	Is the framework distributive?	Yes or no. No explanation is necessary. Yes
i	Does the algorithm necessarily converge?	Yes or no. Explain why. Yes. The transfer function is monotone and the semilattice has finite descending chains.

Remove computations and assignments where the target variable is not used in the output set. Note that we cannot remove input statements under any circumstance.

6. (Bonus 5 points) Describe briefly how you can improve the dead code elimination algorithm from Problem 5 if you are allowed to remove unnecessary conditional statements.

   Solution: We gave credit based on the quality of the analysis proposed. What we had in mind was to see if, after dead code elimination, an if statement contained no code remaining. If so, we can remove the if statement and iterate, since an additional expression is now dead.