CS243 Midterm

11AM – 12:15PM
February 15, 2006

The exam is closed book, but you may use a single-sided 8.5 by 11 inch piece of paper with whatever you wish written upon it. Answer all 5 questions on the exam paper itself.

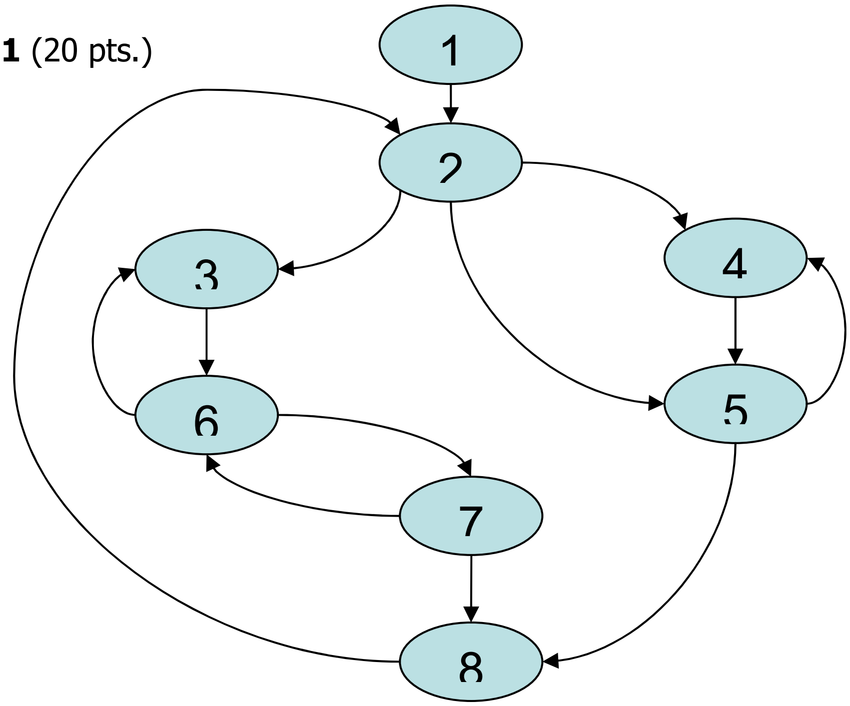
Write your name here: ____SOLUTIONS___________

I acknowledge and accept the honor code.

(signed) ____

Question	Max	Score
1	20	
2	20	
3	30	
4	15	
5	15	
TOTAL	100	

Question 1 (20 pts.)



a) (6 pts.) In the following table, fill out the dominator relationships for the above flow graph (whose entry is node 1).

	Dominates
1	**1,2,3,4,5,6,7,8**
2	**2,3,4,5,6,7,8**
3	**3,6,7**
4	**4**
5	**5**
6	**6,7**
7	**7**
8	**8**

b) (3 pts.) Identify the back edges in the flow graph above.

backedges: 8→2 , 7→6, 6→3, retreating edges that are not back edges: 5→4 or 4→5

c) (4 pts.) What are the natural loops of the back edges you identified in (b)?

8→2: 2,3,4,5,6,7

7→6: 7,6

6→3: 3,6,7

d) (5 pts.) How many DF orders (reverse postorder) are there for this flow graph? Write down four of them.

The number of DF orders: ______5__________

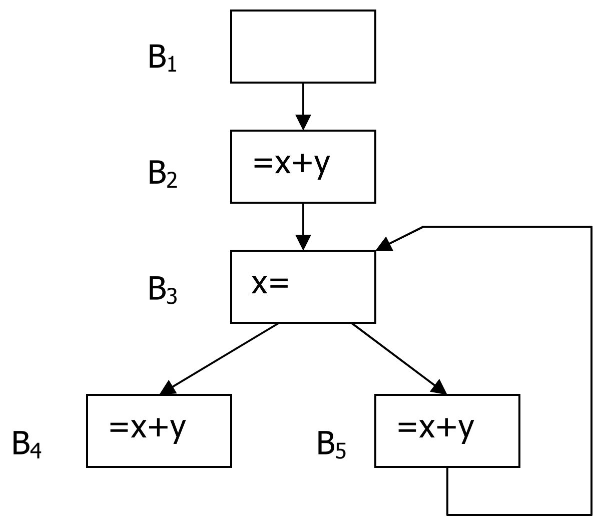
Give 4 of them here:

1,2,4,5,3,6,7,8
1,2,5,4,3,6,7,8
1,2,3,6,7,4,5,8
1,2,3,6,7,5,4,8
1,2,3,6,7,5,8,4

e) (2 pts.) Is the graph reducible? Why or why not?

No it is not because all the recreating edges are not back edges thus after we remove the back edges the graph is still cyclic.

Question 2 (20 pts.) Below is a flow graph with the evaluations of expression x+y shown, along with the one place where that expression is killed (by assignment to x).



a) (7 pts.) For which blocks B is x+y anticipated at the beginning? At the end? List those sets IN[B] and OUT[B] that contain x+y when we compute anticipated expressions. Recall that "anticipated expressions" is a backwards/intersection problem with a transfer function that says INB] is the set of expressions that are either evaluated in B or are in OUT[B] and not killed.

Answer: IN[B_1], IN[B_2], IN[B_4], IN[B_5]; OUT[B_1], OUT[B_3]. Common mistake: not propagating anticipation backwards through B_1.

b) (7 pts.) For which blocks B is x+y "available" (in the sense used in the PRE algorithm) at the beginning? At the end? Recall this data-flow analysis is of the forwards/intersection type, with a transfer function that says OUT[B] contains those expressions that are not killed in B and are either in IN[B] or are anticipated at the beginning of B. Give the sets IN[B] or OUT[B] that contain x+y for this data-flow analysis.

Answer: IN[B_2], IN[B_3], OUT[B_1], OUT[B_2]; OUT[B_4], OUT[B_5].

c) (6 pts.) For which blocks B is x+y in Earliest of B?

Answer: B_1, B_4, and B_5.

Question 3 (30 pts.)
In many cases knowing the *range* of variables is beneficial. For instance, knowing that variables *a* and *b* are between 0 and 127 may allow us to represent *both* variables within one byte instead of two words, thereby providing a more compact representation for certain data structures.

Suppose you are analyzing a program consisting of the following types of statements:

- a = <const>
- a = b
- a = b + <const>
- a = b + c

where all variables and constants are integers.

Your task is to formulate a dataflow problem called VarRange that would allow one to approximate the range of any given variable at any point in the program.

The range is to be represented by an interval [*x*, *y*] where both *x* and *y* are constants. Assume that MI is the biggest represent able integer and we are dealing with *positive* numbers only.

a. **(2 pt)** What are the top and bottom elements of the lattice for the dataflow framework formulation of VarRange?

Top: UNDEF, bottom: [0, MI]

b. **(3 pt)** What is the meet operator for VarRange?

For every variable in a dataflow value, $[x_1, y_1] \wedge [x_2, y_2] = [min(x_1, x_2), max(y_1, y_2)]$.

c. **(2 pt)** What is the ≤ relation induced by the ∧ operator?

$[x_1, y_1] \leq [x_2, y_2]$ iff $x_1 \leq x_2$ and $y_2 \leq y_1$. In other words, $[x_1, y_1]$ covers $[x_2, y_2]$.

d. **(5 pt)** Assume for simplicity that each basic block consists of at most one statement. Define the transfer function for VarRange.

- a = <const> $TF(B)_a$ = [const,const]
- a = b $TF(B)_a = [b_{low}, b_{high}]$
- a = b + <const> $TF(B)_a = [b_{low} + \text{<const>}, b_{high} + \text{<const>}]$
- a = b + c $TF(B)_a = [b_{low} + c_{low}, b_{high} + c_{high}]$

e. **(5 pt)** Is the transfer function you defined above monotonic?

Yes / No (circle one)

If so, please give a proof. Otherwise, please provide a counterexample.

Yes. Suppose we have two values V_1 and V_2, values in the cross-product lattice such that $V_1 \leq V_2$. In particular, $V_1 \leq V_2$ implies that $V_1(b) \leq V_2(b)$ and $V_1(c) \leq V_2(c)$. Consider the very last case (b + c); others are dealt with similarly. The solution below is more detailed than yours was expected to be.

When V_1 is the input to which TF is applied, we get

$TF(B)_a = [V_1(b)_{low} + V_1(c)_{low}, V_1(b)_{high} + V_1(c)_{high}]$

Similarly, for V_2 we get

$TF(B)_a = [V_2(b)_{low} + V_2(c)_{low}, V_2(b)_{high} + V_2(c)_{high}]$

$V_1(b) \leq V_2(b) \Rightarrow V_1(b)_{low} \leq V_2(b)_{low}$ and $V_1(b)_{high} \geq V_2(b)_{high}$.
$V_1(c) \leq V_2(c) \Rightarrow V_1(c)_{low} \leq V_2(c)_{low}$ and $V_1(c)_{high} \geq V_2(c)_{high}$.

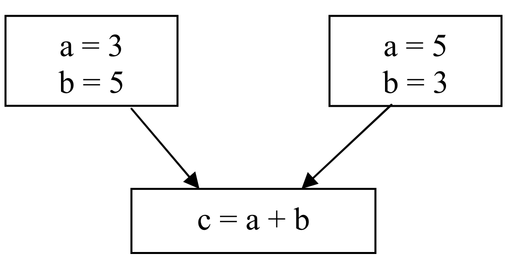
The inequalities above imply that
$V_1(b)_{low} + V_1(c)_{low} \leq V_2(b)_{low} + V_2(c)_{low}$ and $V_1(b)_{high} + V_1(c)_{high} \geq V_2(b)_{high} + V_2(c)_{high}$
which implies that $TF(B)_a(V_1) \leq TF(B)_a(V_2)$.

f. **(6 pt)** Is the transfer function you defined above distributive?

Yes / No (circle one)

If so, please give a proof. Otherwise, please provide a counterexample.

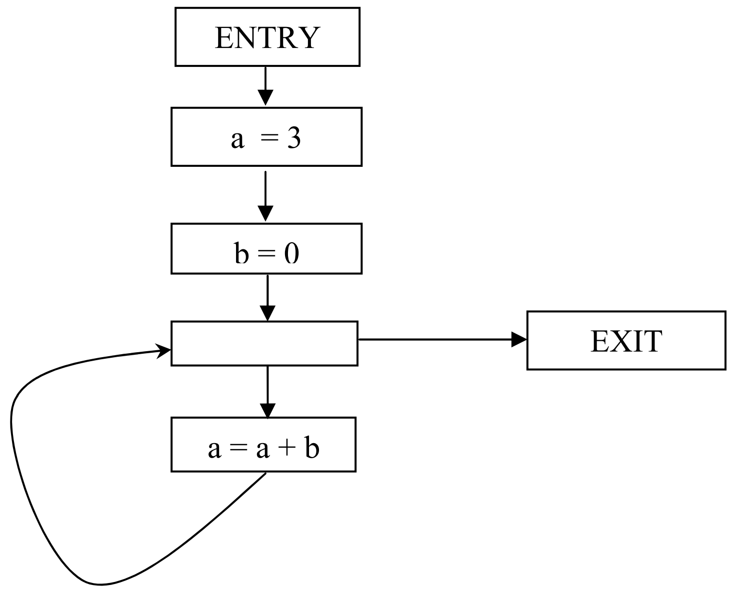
No. Consider the CFG below. At the entrance to the bottom node, the range for a is [3,5]. The range for b is similarly [3,5]. The range for c is [6,10]. However, the MOP solution will produce [8,8] as the answer. Since the MOP and MFP solutions are unequal, we have an non-distributive problem.



g. **(3 pt)** Is the iterative dataflow solver algorithm guaranteed to terminate on VarRange?

Yes, the framework is monotonic only has finite descending chains. Thus, the algorithm with terminate.

h. **(4 pt)** What is the range for variable *a* as computed by your algorithm for the CFG below?



Variable *a* belongs to range [3, 3]

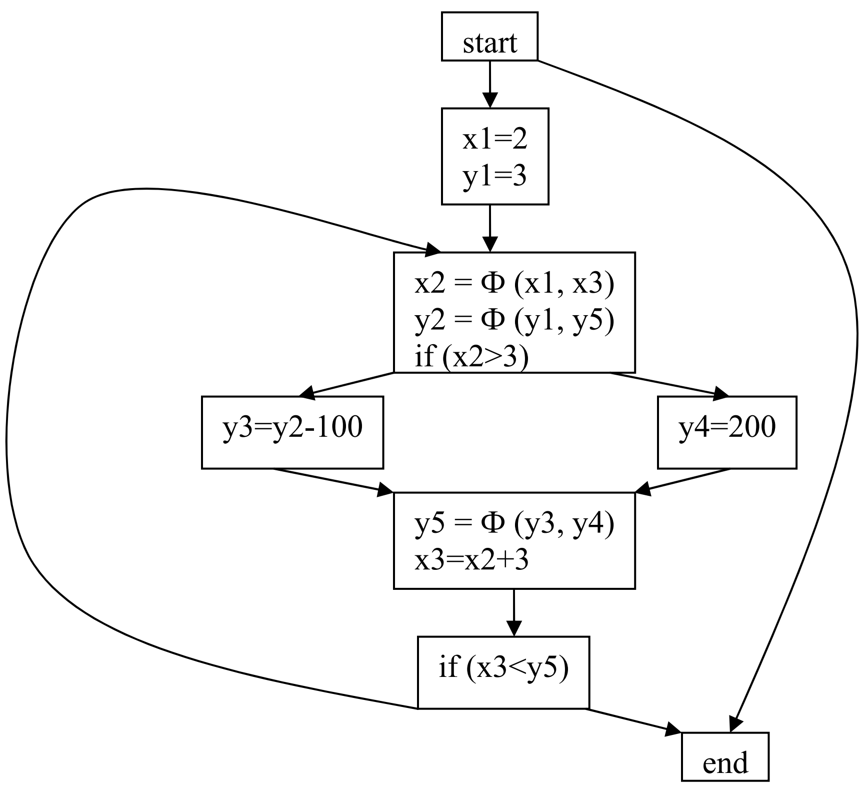
Question 4 (15 points) Convert the program below to SSA form.

a. Construct a CFG
b. Insert Φ functions
c. Rename variables

```
x = 2;
y = 3;
repeat
        if (x>3) then
                y =  y-100
        else
                y = 200;
        x = x + 3;
until (x<y)
```

Draw your CFG with code in the proper form in the space below.

Answer:



Question 5 (15 points) If X appears on every path from Y to END, then X ***pdom*** Y. Prove the following about the *pdom* relation.

a. If *a pdom b* and *b pdom c*, then *a pdom c*.

Answer: Since *b* pdom *c*, *b* appears on every path from *c* to END. Since *a pdom b*, *a* appears on every path from *b* to END. Therefore, *a* appears on every path from *c* to END, then *a* pdom *c*.

b. If *a pdom b* and *b pdom a*, then *a=b*.

Answer: Suppose $a \neq b$. Any path from *a* to END must contain *b*, because *b pdom a*. There has to be *a* between the last *b* and END, because *a pdom b*. Then there is a path from *a* to END without *b*. It can not be true, because *a pdom b*.

c. If *a* and *b* are two postdominators of n, then either *a pdom b* or *b pdom a*.

Answer: Any path from *n* to END must contain both *a* and *b*, because *a* and *b* are both postdominators of *n*.

1. If in every path from *n* to END, *a* follows to *b*, then *a pdom b*, because if not, there is a path from *n* to *b* to END, which can not be true, because *a pdom n*.
2. Similarly, if in every path from *n* to END, *b* follows to *a*, then *b* pdom *a*.
3. If there exist 2 paths P1 and P2 from *n* to END, where *a* follows *b* in P1, and *b* follows *a* in P2. Then there is a new path from *n* to *b* (from P1) and *b* to END (from P2) bypassing *a*, which can't be true because *a* pdom *n*.