CS 243, Winter 2017
February 14, 2017

CS 243 Midterm Exam Solution

This is a open-book, open-laptop, *closed*-network exam. The maximum possible score is 75 points. You have 80 minutes to complete the exam. Make sure you print your name legibly and sign the honor code below. All of the intended answers may be written within the space provided. You may use the back of the preceding page for scratch work.

The following is a statement of the Stanford University Honor Code:

A. *The Honor Code is an undertaking of the students, individually and collectively:*

   (1) *that they will not give or receive aid in examinations; that they will not give or receive unpermitted aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading;*

   (2) *that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the Honor Code.*

B. *The faculty on its part manifests its confidence in the honor of its students by refraining from proctoring examinations and from taking unusual and unreasonable precautions to prevent the forms of dishonesty mentioned above. The faculty will also avoid, as far as practicable, academic procedures that create temptations to violate the Honor Code.*

C. *While the faculty alone has the right and obligation to set academic requirements, the students and faculty will work together to establish optimal conditions for honorable academic work.*

I acknowledge and accept the Honor Code.

_____

(Signature)

_____

(Print your name)

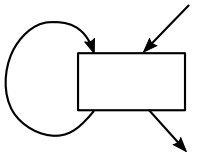
Prob	# 1	# 2	# 3	# 4	# 5	Total
Score						
Max	16	15	5	15	24	75

1. Multiple Choice (16 points)

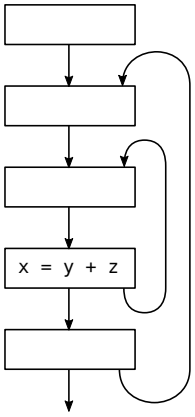
Answer the following questions and justify your answer in 1-2 sentences. Please keep the justification brief.

(a) **True** or False: A node's dominance frontier can include itself. (The dominance frontier of a node B is the set of all nodes which have a predecessor dominated by B but are not strictly dominated by B.)

Any node with a self loop is in its own dominance frontier:



(b) **True** or False: A single application of the Lazy Code Motion algorithm to the following doubly-nested loop can eliminate ALL its redundant operations.



A single application of the Lazy Code Motion algorithm will hoist the computation of `y + z` completely out of the doubly nested loop and assign its value to a temporary `t` to be used inside the loop for the assignment to `x` (i.e. `x = t`).

(c) True or **False**: It is possible to eliminate more redundancy in some graphs by applying the Lazy Code Motion algorithm a second time.

In the following loop, both `x` and `y` are loop invariant but `y` cannot be lifted until `x` is:

```
while (1) {
    x = a + b;
    y = x + c;
}
```

The critical part is that you need to have more than one expression for multiple passes to be necessary. This is a reasonable assumption for most programs.

(d) True or **False**: If the graph coloring algorithm finds a coloring without spilling, it always uses the minimum number of registers.

Depending on the

(e) **True** or False: A semi-lattice defines a partial order.

A semi-lattice is defined by a set of values and a meet operator, and a meet operator defines a partial order via $x \leq y \equiv x \wedge y = x$.

(f) True or **False**: A partial order defines a semi-lattice.

Consider three elements, a, b, c, with the order $a < c$ and $b < c$, but a and b uncomparable. This defines a partial order but not a semi-lattice because the meet operation is undefined for a and b (the resulting lattice would have "two bottoms", which is illogical). Note the answer to this question is **not** that a partial order does not define a set of elements; it does define a set as well as a relation $\leq$ on that set.

(g) Which passes of PRE are responsible for removing redundancy? Choose and justify all that apply.

i. **Pass 1**

ii. **Pass 2**

iii. Pass 3

iv. Pass 4/**Pass 4**

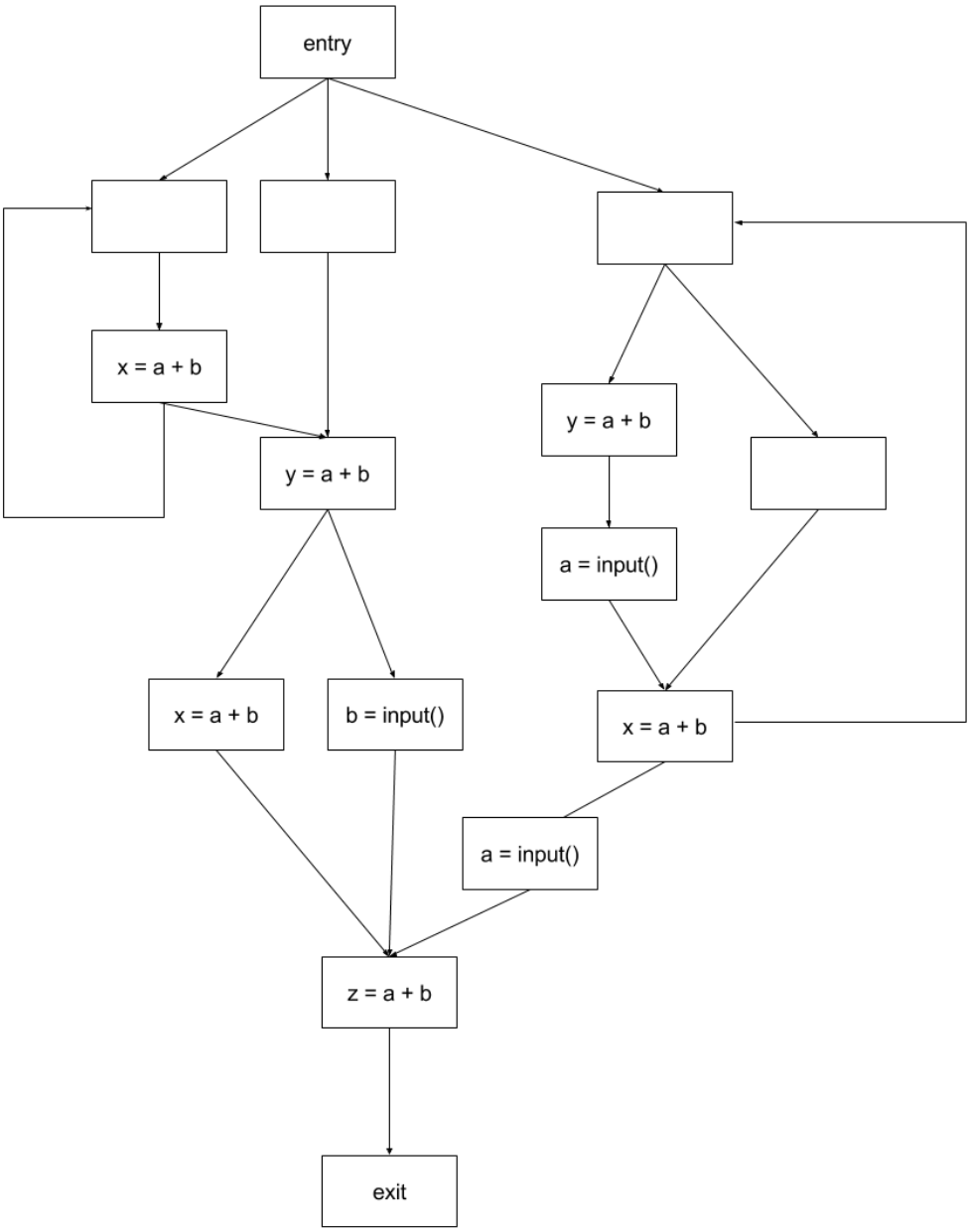
Pass 1 places operations at the frontier of anticipation (boundary between not anticipated and anticipated). Pass 2 is responsible for ensuring if anticipation oscillates then redundant operations are not inserted where they are available. These two passes are responsible for eliminating redundancy.

Pass 3 does not remove redundancy and is only used to postpone expressions to reduce register pressure.

Pass 4 also does not remove redundant *operations* but does remove redundant *assignments*. We accepted either answer based on your justification and interpretation of redundant.

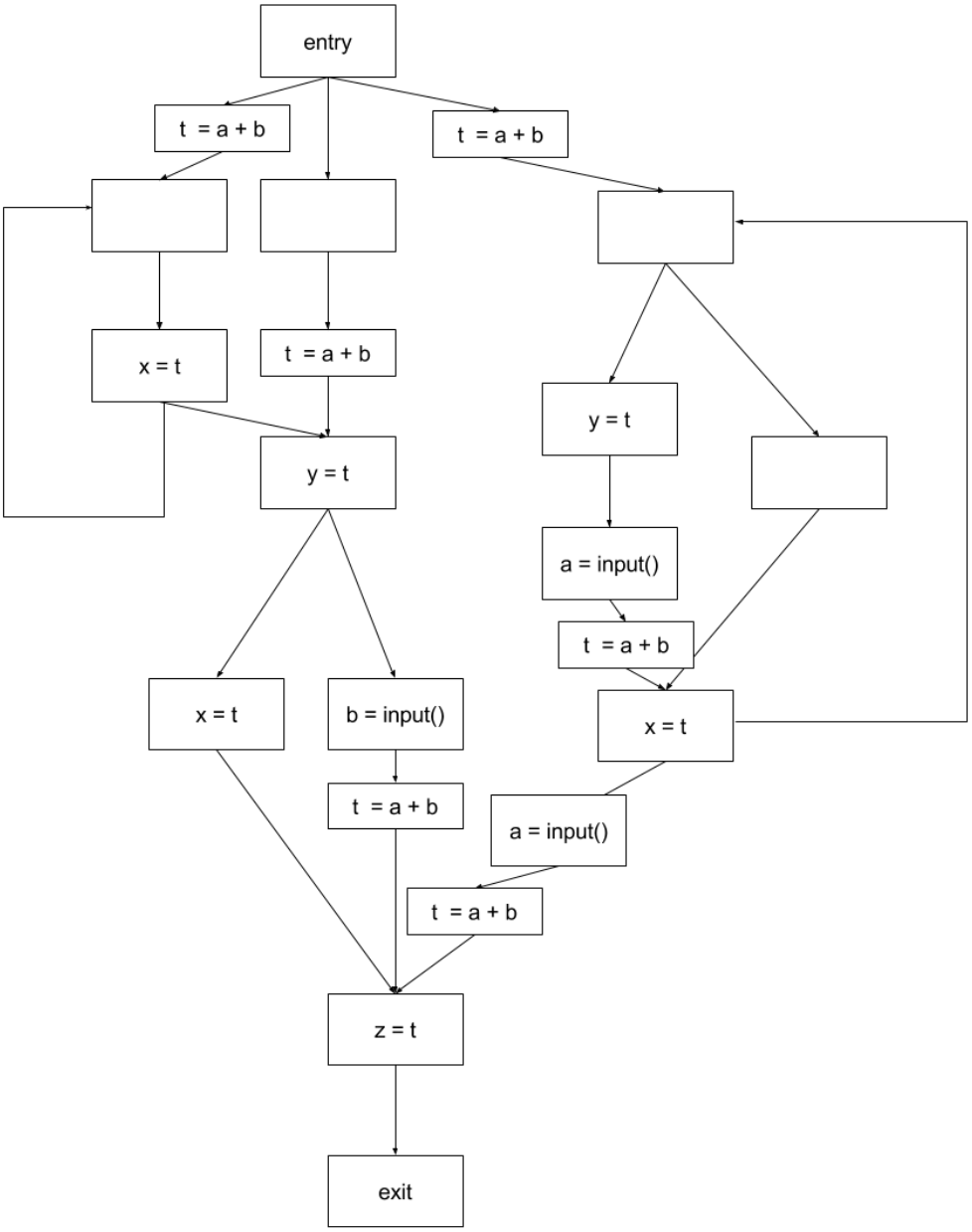
2. Partial Redundancy Elimination (15 points)

Apply lazy code motion to the following program. You do not need to show the intermediate steps, just show the optimized code. You may add basic blocks to the flow graph, but only show those that are not empty in your solution. You may use the following page for scratch work.



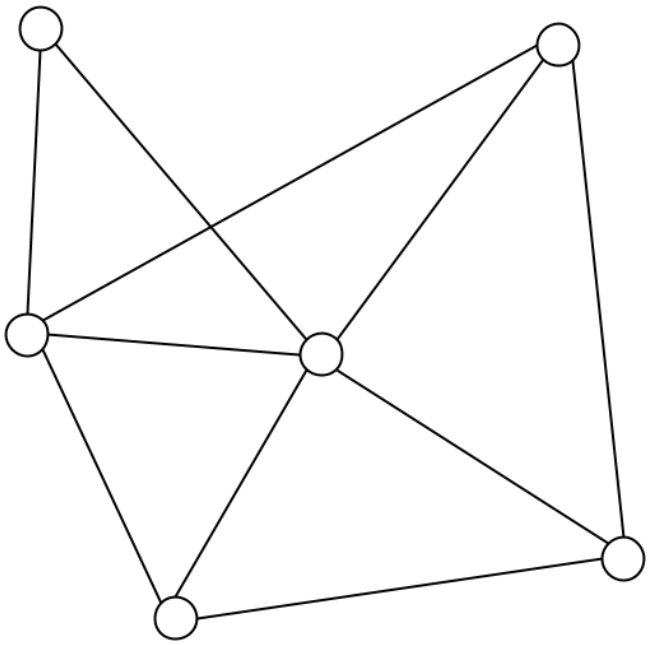
This page is intentionally left blank.

Solution: The following control flow graph shows the changes made as a result of applying lazy code motion.



3. **Register Allocation (5 points)**

On a machine with 3 registers, will the coloring algorithm always, sometimes, or never find a coloring for the interference graph below?

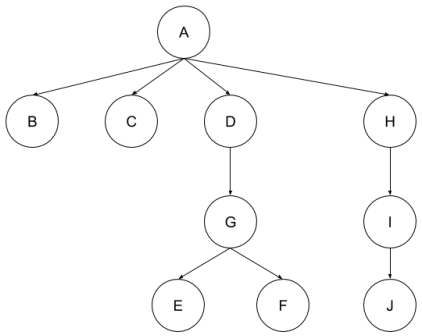


Solution: The algorithm will **sometimes** find a coloring for the interference graph, depending on the heuristic it uses to choose variables. The maximum degree of a node is 5. Since we are coloring with 3 colors, it is not guaranteed that the algorithm will always find a coloring. However, there does exist a coloring with 3 colors, and depending on how the heuristic chooses the next variable and color, it can find a coloring.

4. Dominators (15 points)

Consider the following graph.

(a) Draw the dominator tree for the graph, assuming A is the entry node.
Solution: The dominator tree is given below.



(b) Find the back edges and their natural loops in the graph.
Solution: The back edges and their natural loops are given below:

E $\rightarrow$ D with natural loop {D, E, G}
F $\rightarrow$ D with natural loop {D, E, F}
I $\rightarrow$ H with natural loop {H, I, J}
J $\rightarrow$ I with natural loop {I, J}
F $\rightarrow$ F with natural loop {F}

(c) Is the graph reducible?
Solution: Yes, the graph is reducible. If you construct a depth-first search spanning tree, you will find that the set of retreating edges is the same as the set of back edges.

5. Sign Analysis (24 points)

Define a data flow analysis to determine for every integer variable, whether it can be negative, zero, or positive, at every program point as precisely as possible. Assume that the program has no overflows. The programming language has only statements of the form:

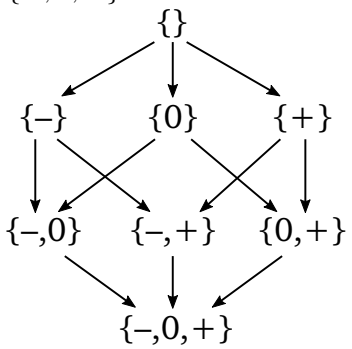
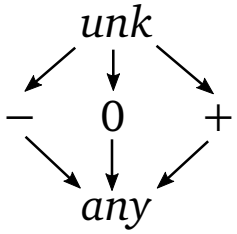
```
z = input()
z = sq(x)     // square
z = incr(x)   // increment
```

```
z = −y
z = x + y
z = x * y
```

Note: there are no constants in this language. All variables have arbitrary values on entry to the program. Your analysis should be precise enough to find that `z` is positive at the end of the following code:

```
x = input()
y = sq(x)
z = incr(y)
```

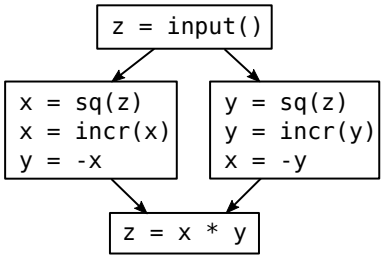
Fill in the table to specify a data flow analysis and answer the following questions.

Direction	Forwards	Forwards
Domain	**We gave 6 points for correctly answering this lattice and meet operation:** Each variable is a subset of $\{-, 0, +\}$, giving 8 elements (with interpretation unk, ≤ 0, ≥ 0, ≤ 0, ≥ 0, $\neq 0$, any)	**We gave 2 points for this simpler, less powerful lattice that isn't able to handle the example in the problem:** Each variable is one of $\{unk, any, -, 0, +\}$
Meet Operation OR Semi-lattice Diagram	$\cup$ Top Element is $\{\}$, bottom is $\{-, 0, +\}$ 	Top Element is unk, bottom is any 
Transfer Functions: `z = input()`	$\{-, 0, +\}$	any
`z = sq(x)`	Set z to the union over: $\{+\}$ if $-$ or $+ \in$ x $\{0\}$ if $0 \in$ x Note that this maps $\{\}$ to $\{\}$, preserving monotonicity.	Set z to x but apply the function: $- \to +, 0 \to 0, + \to +, any \to any, unk \to unk$. You needed to specify what happens to unk. If it becomes any then you have a non-monotone function.
`z = incr(x)`	Set z to the union over: $\{+\}$ if 0 or $+ \in$ x $\{-, 0\}$ if $- \in$ x	Set z to x but apply the function: $- \to any, 0 \to +, + \to +, any \to any, unk \to unk$.

<table>
<tr><td><code>z = -y</code></td><td>Set z to the union over:
$\{+\}$ if $-\in$ x
$\{0\}$ if $0\in$ x
$\{-\}$ if $+\in$ x</td><td>Set z to x but apply the function: $-\rightarrow +, 0\rightarrow 0, +\rightarrow -, any\rightarrow any, unk\rightarrow unk$.</td></tr>
<tr><td><code>z = x + y</code></td><td>Set z to the union over:
$\{+\}$ if $+$ in either
$\{-\}$ if $-$ in either
$\{0\}$ if 0 in both
$\{0\}$ if $+$ in one and $-$ in other</td><td>If either is unk, set z to unk. If either is any, set z to any. Otherwise:<table><tr><td></td><td>-</td><td>0</td><td>+</td></tr><tr><td>-</td><td>-</td><td>-</td><td>any</td></tr><tr><td>0</td><td>-</td><td>0</td><td>+</td></tr><tr><td>+</td><td>any</td><td>+</td><td>+</td></tr></table></td></tr>
<tr><td><code>z = x * y</code></td><td>Set z to the union over:
$\{+\}$ if $+$ in both
$\{+\}$ if $-$ in both
$\{0\}$ if 0 in either
$\{-\}$ if $+$ in one and $-$ in other</td><td>If either is unk, set z to unk. Otherwise:<table><tr><td></td><td>-</td><td>0</td><td>+</td><td>any</td></tr><tr><td>-</td><td>+</td><td>0</td><td>-</td><td>any</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>+</td><td>-</td><td>0</td><td>+</td><td>any</td></tr><tr><td>any</td><td>any</td><td>0</td><td>any</td><td>any</td></tr></table></td></tr>
<tr><td>Boundary Condition</td><td>IN[entry] = $\bot$ = $\{-, 0, +\}$. The problem says that the values are arbitrary on entry, not undefined. This cannot be $\top$.</td><td>$\bot$ = any</td></tr>
<tr><td>Initialization</td><td>IN[B] = $\top$ = $\{\}$. This cannot be $\bot$ or else the values in a loop will never change.</td><td>$\top$ = unk</td></tr>
</table>

(a) Is your framework monotone? **Yes.** We took off a point if you answered incorrectly given your transfer functions. Non-monotone dataflow equations are useless!

(b) Is it distributive? **No**. Consider:



The MOP solution has z negative in the last block but the dataflow analysis cannot determine this.

(c) Will the iterative algorithm always converge? **Yes.** We took off points if this wasn't true for your analysis.