MIT 6.035 Spring 2011 Quiz 2

Full Name: ______________________________

MIT ID: ______________________________

Athena ID: ______________________________

Question:	1	2	3	4	5	6	Total
Points:	10	20	20	15	15	20	100
Score:							

1. We want to optimize the following program snippet written in the Decaf language by eliminating common subexpressions:

```
i = callout("get_int_035");
j = i + 1;
k = i;
l = k + 1;
```

where the `get_int_035` function reads an integer form standard input and returns it.

(a) (5 points) What does the optimized code look like when we use *value numbering* to find common subexpressions?

i = callout ("...");
j = i + 1
k = i
l = j

i = callout
j = i + 1
t_1 = j
k = i

Var to Val: $i \to v_1$, $j \to v_2$, $k \to v_1$

Exp to Val: $v_1 + 1 \to v_2$

Exp to Tmp: $v_1 + 1 \to t_1$

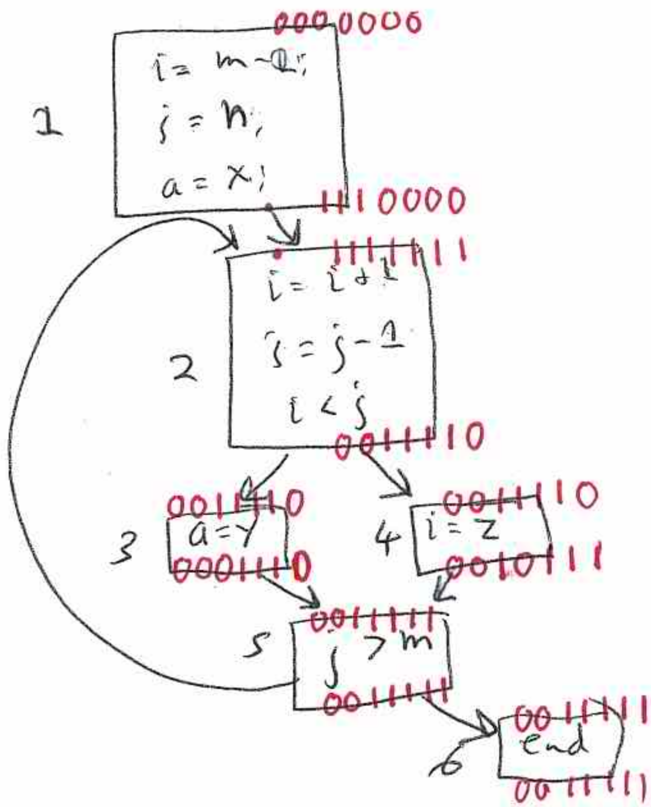
(b) (5 points) What does the optimized code look like when we use *available expression* analysis to find common subexpressions?

no optimization
code is same

2. We want to compute reaching definitions for the following program:

```
L1:     i = m - 1;
L2:     j = n;
L3:     a = x;
        do {
L4:             i = i + 1;
L5:             j = j - 1;
                if (i < j) {
L6:                     a = y;
                } else {
L7:                     i = z;
                }
        } while(j > m);
```

(a) (5 points) Draw the control flow graph of this program.



(b) (5 points) Compute GEN[n] and KILL[n] for each basic block n.

GEN[1] = 1110000
2 = 0001100
3 = 0000010
4 = 0000001

KILL[1] = 0001111
2 = 1100001
3 = 0010000
4 = 1001000

(c) (5 points) Set up data-flow equations (IN[n] $= \cdots$ and OUT[n] $= \cdots$) for each basic block n.

IN[1] = 0000 0000
2 = OUT[1] ∪ OUT[5]
3 = OUT[2]
4 = OUT[2]
5 = OUT[3] ∪ OUT[4]
6 = OUT[5]

OUT[n] = (IN[n] - KILL[n]) ∪ GEN[n]

IN or OUT not both

(d) (5 points) Find out the solution of the data-flow equations.

IN[1] = 0000 0000
IN[2] = 111
3 = 0011111
4 = 0011111
5 = 0011111

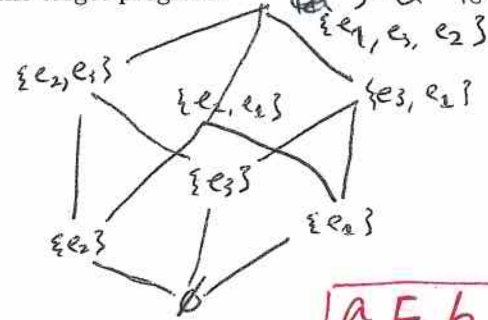
3. Design a data-flow analysis that determines which expressions are very busy at each program point. An expression is very busy at a program point p if along every path from p the expression is always used before a redefinition of any of the variables occurring in it.

   (a) (5 points) Is the data-flow analysis a forward analysis? Or a backward analysis?

   ~~Forward~~ Backward

   (b) (5 points) Draw the Hasse diagram of the lattice used in the analysis, assuming that there are just three expression (e_1, e_2 and e_3) in the target program.

   3-d Hypercube



   $a \sqsubseteq b \rightarrow a \sqcup b = b$

   (c) (5 points) What is the confluence operator?

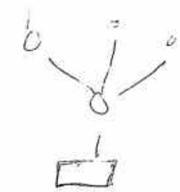
   $\cap$

   $a \sqcup b$

   $a \cap b = b$

   (d) (5 points) What is the initial value of IN[entry] or OUT[exit], depending on your answer to (a).

   $\emptyset$



4. (a) (10 points) Prove that the "~~grater~~ greater than or equal" relation ($\geq$) is a partial order on the set of integers with both positive and negative infinity ($\mathbb{Z} \cup \{-\infty, \infty\}$).

1: $a \leq a$

2: $a \leq b \wedge b \leq c \Rightarrow a \leq c$

3: $a \leq b \wedge b \leq a \Rightarrow a = a$

True for all $\mathbb{Z}$

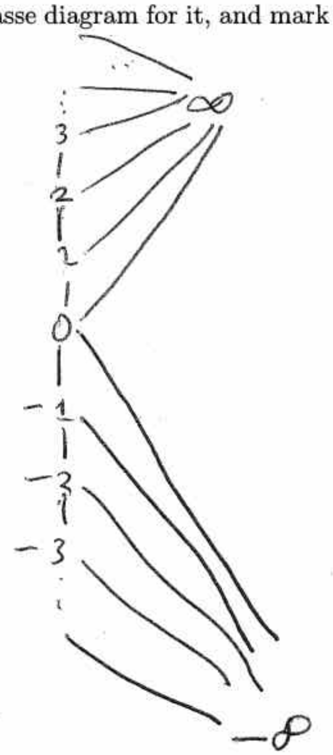
consider $a = \infty$ 1, 2, 3 all hold, $b = c = \infty$

$b = \infty$ 1, 2, 3 all hold, $c = \infty$

$c = \infty$ 1, 2, 3 all hold

similar case analysis for $-\infty$

(b) (5 points) Draw the Hasse diagram for it, and mark its greatest element ($\top$) and least element ($\perp$).



5. If your proof is not correct, you will not get a score even when your yes/no answer is correct.

(a) (8 points) If a lattice has a greatest or least element, is it always unique? Prove it or disprove it.

yes.

Proof: contradiction.

assume 2 greatest elements g_1, g_2

consider $g = g_1 \vee g_2$ $g_1 \le g$, $g_2 \le g$ so

~~g...~~ either $g = g_1$ $g_2 \le g$ so g_2 not greatest

(similar for $g = g_2$)

or $g \neq g_1$, $g \neq g_2$, so

g_1, g_2 not greatest

(b) (7 points) Does a complete lattice always have both greatest and least elements? Prove it or disprove it.

yes. ~~Consider~~ consider $\bigvee L$ = greatest element (here L is lattice)

$\bigwedge L$ = least element

6. Alice designed a mysterious data-flow analysis on programs written in the following language:

$$\begin{array}{lcl} \mathrm{S} & \rightarrow & id := \mathrm{E} \mid \texttt{if}\ (\mathrm{E} < \mathrm{E})\ \texttt{then}\ \mathrm{S}\ \texttt{else}\ \mathrm{S} \\ \mathrm{E} & \rightarrow & id + id \mid c \end{array}$$

where id and c denote a variable and a non-negative integral constant. It is known that she modeled a program state at each program point as a function that maps each variable to its value. For example, the $\{x \mapsto 1, y \mapsto 2\}$ program state means that x and y have 1 and 2 at the program point, respectively. Also, her abstraction function is as follows:

$$\mathrm{AF}(\{id_1 \mapsto v_1, id_2 \mapsto v_2, \cdots, id_n \mapsto v_n\}) = \{id_1 \mapsto (v_1 \,\%\, 3), id_2 \mapsto (v_2 \,\%\, 3), \cdots, id_n \mapsto (v_n \,\%\, 3)\}$$

where % is the remainder operator. Let's restore her data-flow analysis from the abstract function.

(a) (5 points) Define the base lattice for her data-flow analysis. Note that the actual lattice is defined using element of the base lattice as follows:

$$\{id_1 \mapsto \widehat{v}_1, id_2 \mapsto \widehat{v}_2, \cdots, id_n \mapsto \widehat{v}_n\}$$

where $\widehat{v}_1$, $\widehat{v}_2$ and $\widehat{v}_n$ are element of the base lattice.

Base lattice =

(b) (10 points) Define the transfer functions for the following basic blocks.

$f \downarrow$ `x := c` $\downarrow$

$$f'(id) = \begin{cases} c \,\%\, 3 & \text{if } id = x \\ f(id) & \text{otherwise} \end{cases}$$

$\downarrow f$ `x := y + z` $\downarrow$

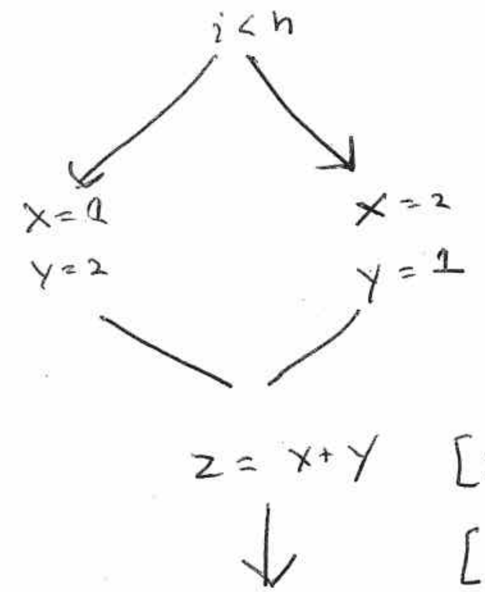
$$f'(id) = \begin{cases} (f(y) + f(z)) \,\%\, 3 & \text{if } id = x \\ f(id) & \text{otherwise} \end{cases}$$

(c) (5 points) ~~Can~~ will the analysis always produce the meet-over-path solution? Justify your answer by proof sketch or example. If your justification is not correct, you will not get a score even when your yes/no answer is correct.

~~No. Have control-flow imprecision~~

~~x = 1~~
~~y = 2~~

No, have control-flow imprecision



$[z = \top, \dots]$ in analysis

$[z = 0, \dots]$ is MOP solution