6.035
Spring 2022
Quiz 2 Solutions

Name: _____
MIT Kerb: _____
Time Limit: 50 min

- **DO NOT open the exam booklet until you are told to begin. You should write your name and kerb id at the top. Read the questions carefully.**

Problem	Points	Score
1	11	
2	14	
3	8	
4	17	
5	11	
Total:	61	

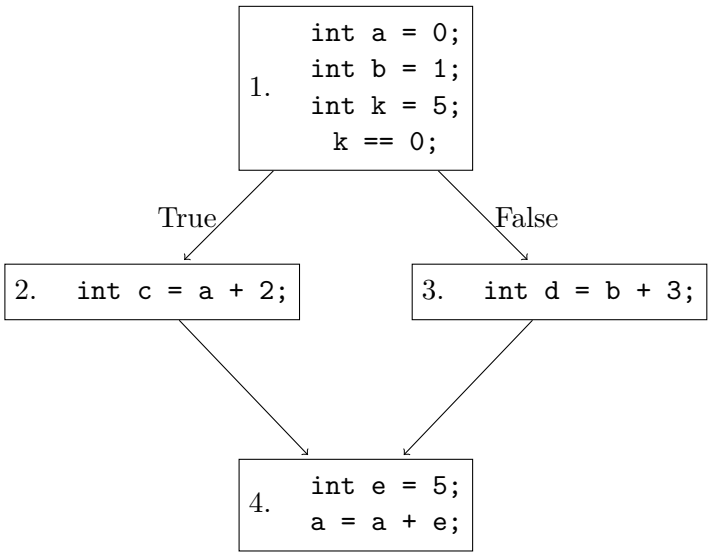
1. **Program Analysis**

Liveness analysis determines which variables are "live" at every program point. It has applications in register allocation as well as dead code elimination.

Remember that a variable V is said to be live at point P if:

- there is a use U of V along some path starting at P
- there is no redefinition of V along that path until U

In this problem we will guide you through performing a backwards dataflow analysis for the variables in the following CFG:



Note that the analysis does not attempt to resolve control flow branches. So it would not be able to detect that the true branch of the if statement is never taken. Instead, it assumes that either branch may be taken.

(a) (4 points) In lecture we defined the DEF and USE sets for each basic block as follows:

- DEF[b] is the set of all definitions generated by basic block b
- USE[b] is the set of all uses in basic block b with definitions outside of basic block b.

Complete the table below by writing the DEF and USE sets, represented as sets of variable names (example $\{a, e\}$), for each basic block in the CFG.

block no.	DEF[b]	USE[b]
1	a, b, k	
2	c	a
3	d	b
4	a, e	a

(b) (4 points) We also define the IN and OUT sets:

- IN[b] is the set of all variables that are live at the beginning of basic block b
- OUT[b] is the set of all variables that are live at the end of basic block b

Using the previously computed USE and DEF sets, compute the IN and OUT (still as sets of variable names) sets for each basic block after the analysis is complete. You can assume that variable a is available outside of the method and is therefore live upon exit. This means that the starting OUT for last CFG block is $\{a\}$.

block no.	IN[b]	OUT[b]
1		a, b
2	a	a
3	a, b	a
4	a	a

(c) (3 points) Given the solution to the dataflow analysis you found above, which variables are dead (not live on any path) after their definition? Which statements constitute dead code and can therefore be eliminated?

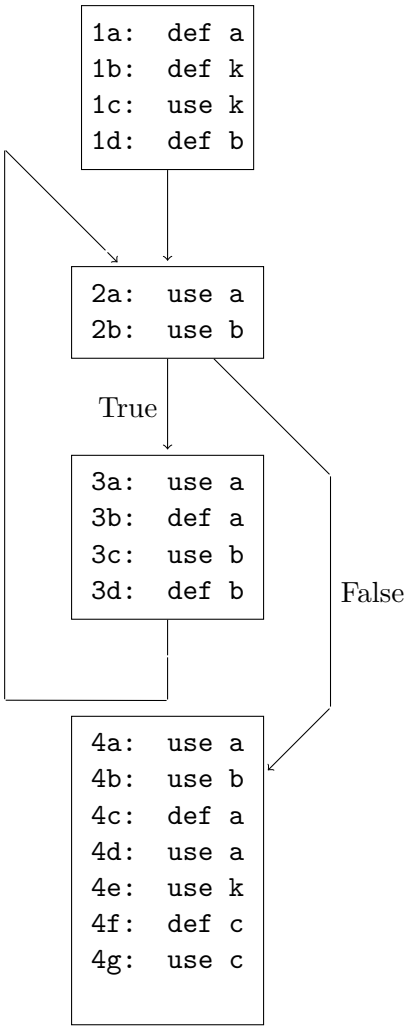
Variables c and d are dead all throughout the program, including at the point after they are defined, since they don't appear in any of the sets. Therefore, assignments to c and d are dead code and can be removed.

Note that even if e is not live before or after the last basic block, it is live within the block, at the program point between the last two instructions. It therefore can not be removed.

Credit was awarded for performing multiple rounds of dead code elimination which would lead to removing b as well.

2. **Register Allocation**

In this problem, you will perform register allocation for the following CFG with def and use statements:



The following are the def-use chains for each variable in the program. A def-use chain is a pair (d, u) where d is the label of a definition and u is the label of a corresponding use of that definition.

a: (1a, 2a), (1a, 3a), (1a, 4a), (3b, 2a), (3b, 3a), (3b, 4a), (4c, 4d)

b: (1d, 2b), (1d, 3c), (1d, 4b), (3d, 2b), (3d, 3c), (3d, 4b)

k: (1b, 1c), (1b, 4e)

c: (4f, 4g)

(a) (4 points) Write the set of webs in the program. Write each web as the set of def-use chains, the pairs, that belong to the web, in terms of the labels from the CFG. We have given you names `w1-w7` for the webs, use only as many names as you need.

Solution:

w1: { (1a, 2a), (1a, 3a), (1a, 4a), (3b, 2a), (3b, 3a), (3b, 4a)}

w2: { (4c, 4d)}

w3: { (1d, 2b), (1d, 3c), (1d, 4b), (3d, 2b), (3d, 3c), (3d, 4b)}

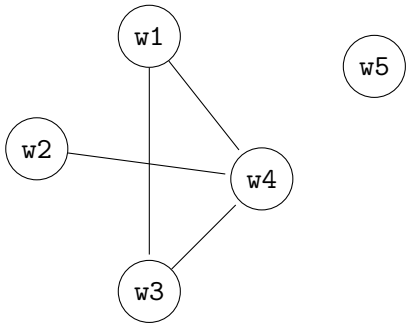
w4: { (1b, 1c), (1b, 4e)}

w5: { (4f, 4g)}

w6:

w7:

(b) (4 points) Draw the interference graph for the webs. Each node in the interference graph should represent one web. There should be an edge between two nodes if the two webs interfere. Label each node with the name (`w1-w7`) of the corresponding web.



(c) (2 points) What is the minimum number of registers you need in order to perform the computation on variables a-c?

3, since the interference graph can be colored in 3 colors.

(d) (4 points) In order to reduce the number of registers needed we can split the web of a variable into two webs by spilling (storing) the variable to memory over some portion of the program. Which variable would you spill, what web is the one that gets split, and why?

Spill variable k

Split web w4

k has the least spill cost since it is used fewer times and does not appear inside the loop.

3. **Loop Optimizations**

In the following program, j is a derived induction variable in the family of base induction variable i.

```
i = 0;
j = 0;
while (i < 1000) {
    i = i + 5;
    j = i*3 + 32;
}
return j;
```

(a) (4 points) Rewrite the program after induction variable recognition and induction variable strength reduction (and no other optimizations): i = 0; j = 32; while (i < 1000) { i = i + 5; j = j + 15; }

(b) (4 points) Rewrite the program after induction variable recognition, induction variable strength reduction, and induction variable elimination (and no other optimization): j = 32; while (j < 3032) { j = j + 15; }

4. **Range Analysis**

Consider a set P, which contains $\perp$ and values of the form $\langle c_1, c_2 \rangle$, where c_1 and c_2 are **non-negative integers** (0, 1, 2, 3, ...) and c_1 is not greater than c_2, i.e., $0 \leq c_1 \leq c_2$.

The following partial order $\leq_p$ is defined on all elements of P:

$$\langle c_1, c_2 \rangle \leq_p \langle c_3, c_4 \rangle \text{ if and only if } c_3 \leq c_1 \text{ and } c_2 \leq c_4, \text{ i.e., } c_3 \leq c_1 \leq c_2 \leq c_4$$

(a) (2 points) Define the join operator $\vee$ (least upper bound) for elements in the lattice:

$$\langle c_1, c_2 \rangle \vee \langle c_3, c_4 \rangle =$$

$\langle \min(c_1, c_3), \max(c_2, c_4) \rangle$

(b) (2 points) Is the lattice complete? Explain. The lattice is not complete, as defined does not contain infinity, and therefore there is no upper bound for the set of non-negative integers

You will use this lattice to perform a range analysis on programs that compute with non-negative integers. For each variable in the program, the analysis uses a single lattice element to represent its information about the maximum and minimum value of each variable.

The analysis should be sound in the sense that if the analysis computes the lattice point $\langle c_1, c_2 \rangle$ to represent the minimum and maximum information about a variable x, then the actual value of x in any program execution will be at least c_1 and at most c_2.

Programs contain statements of the following form (as well as others):

- $x = c$;
- $z = x + y$;

You will next define the transfer function for these statements. Assume that the analysis represents its information about the values of x, y, and z with the lattice values $\langle c_1, c_2 \rangle$, $\langle c_3, c_4 \rangle$, and $\langle c_5, c_6 \rangle$, respectively so that $[x \to \langle c_1, c_2 \rangle,\ y \to \langle c_3, c_4 \rangle,$ and $z \to \langle c_5, c_6 \rangle]$ is the dataflow information at the program point before the statement.

Your analysis should be as precise as possible.

(c) (2 points) Give the transfer function for the statement $x = c$, where c is a non-negative integer:

$$[x \to \langle c, c \rangle, y \to \langle c_3, c_4 \rangle, z \to \langle c_5, c_6 \rangle]$$

(d) (2 points) Give the transfer function for the statement $x = y + z$:

$$[x \to \langle c_3 + c_5, c_4 + c_6 \rangle, y \to \langle c_3, c_4 \rangle, z \to \langle c_5, c_6 \rangle]$$

Consider the following program:

```
if (...)
{
    x = 1;
    y = 2;
}
else
{
    y = 1;
    x = 2;
}
z = x + y;
```

(e) (3 points) Assume you analyze this program using the forward dataflow analysis algorithm presented in lecture, representing the range information for each variable using the above lattice. What is the sequence of lattice elements that this analysis computes for the variables x, y, and z at the program point after the final statement $z = x + y$?

Any of

$$[x \to \langle 1,1\rangle, y \to \langle 2,2\rangle, z \to \langle 3,3\rangle], [x \to \langle 1,2\rangle, y \to \langle 1,2\rangle, z \to \langle 2,4\rangle]$$

$$[x \to \langle 2,2\rangle, y \to \langle 1,1\rangle, z \to \langle 3,3\rangle], [x \to \langle 1,2\rangle, y \to \langle 1,2\rangle, z \to \langle 2,4\rangle]$$

$$[x \to \langle 1,2\rangle, y \to \langle 1,2\rangle, z \to \langle 2,4\rangle]$$

(f) (3 points) What is the meet over paths solution for x, y, and z for the above program and analysis at the program point after the final statement $x = y + x$?

$$[x \to \langle 1,2\rangle, y \to \langle 1,2\rangle, z \to \langle 3,3\rangle]$$

(g) (3 points) Consider the following program:

```
x = 1;
y = 1;
while (...)
{
    x = x + y;
}
```

What is the sequence of lattice elements that the analysis computes for the variable x at the program point before the statement $x = x + y$ in the while loop?

$\langle 1, 1 \rangle, \langle 1, 2 \rangle, \langle 1, 3 \rangle, ...$

5. **Parallelization**

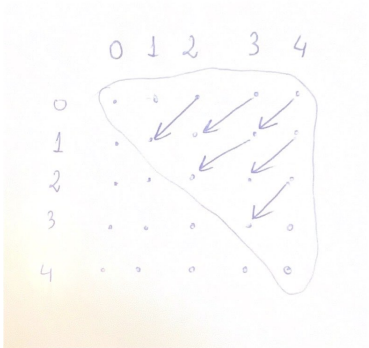
Consider the following program:

```
for (i = 0; i < n; i += 1) {
    for (j = i; j < n; j += 1) {
        A[i, j] = A[i - 1, j + 1] - 2;
    }
}
```

$A[i, j]$ means the element at the i-th row and j-th column in the 2-dimensional array A. You can assume that there are no out of bounds array accesses in this loop.

(a) (3 points) Assume that $n = 5$. In the grid below, draw a line around the dots in the iteration space for this loop and draw the distance vectors. Ignore out of range cases. Each dot represents the value of i and j for an iteration.

```
          j
      0   1   2   3   4

i  0  .   .   .   .   .

   1  .   .   .   .   .

   2  .   .   .   .   .

   3  .   .   .   .   .

   4  .   .   .   .   .
```



(b) (4 points) What is the distance vector for these loops? 1, -1

(c) (4 points) According to this distance vector, which (if any) of these loops can execute in parallel and why? inner loop (j) beacause $i - 1$ information is already processed.