Assignment 3
More dataflow analysis
Due: February 4, 11:00 am

This is a written assignment, every student must hand in his or her homework. Bring your homework to class on February 4th. SCPD students may submit their homework by e-mail via scpd-distribution@lists.stanford.edu or give your homework to the courier. There is also an online-component on Gradiance that must be finished before the deadline.

1. Consider a language which has only non-negative integer variables in the range \([0,255]\). This language has the following types of statements.

**Assignments**
\[x = k\]
(k is a constant)

**Expressions**
\[x = a \text{ op } b\]
(a is a variable and \(b\) is a variable or constant and \(\text{op}\) is either +,-,\(*,/\))

**Gotos and conditional branches**
(No assignments are done to any variables in conditional branch statements.)

We would like to issue two types of warnings:
WARNING I: Possible overflow
WARNING II: Possible underflow

Example:
\[y = 16\]
\[x = y \times 16\] (Warning I)
\[y = 5\]
\[x = y - 10\] (Warning II)

a. Explain what each value in the dataflow analysis represents?
b. What is the domain?
c. What is the direction of data flow?
d. What is the transfer function?
e. What is the meet operator?
f. What are the initialization conditions?
g. How many iterations can it take to converge?

h. If the range of our numbers were \([0,4294967295]\) (unsigned long). Is it feasible to wait for convergence? If not, what do you propose to do?

2. Run PRE on the following block of code and draw the output after the final stage. You may add new empty basic blocks wherever necessary.

```
entry

a = x + y
x = input()

b = x + y
y = input()

exit
```
3. You are given the task of optimizing the code given below but you’re only allowed to use the following three optimization techniques:

- PRE (as discussed in class)
- Constant propagation (as discussed in class)
- Copy Propagation (Discussed in Section 9.1.5, )

in any order and multiple times if necessary.

a. What is the order in which you executed them to produce the best optimized code by running a minimum number of analysis?

b. What is the final optimized program?