CSE 6341, Written Assignment 5
Due Wednesday, April 10, 11:59 pm (8 points)

Your submissions should be uploaded via Carmen. Create your answers using a text editor and upload the file (e.g., plain text, Word, PDF). Alternatively, you can write your answers by hand and take a photo (or scan), but please ensure that (1) your handwriting is *clear and legible*, and (2) your photo or scan has *high resolution*, to allow the grader to read and understand your submission.

Q1 (2 points): Consider the abstracted semantics discussed in class, restricted to program that use only unsigned integer values (similar to the "unsigned int" type in C, but with infinite precision). Unsigned integers can have values greater than or equal to zero. Therefore, the set of abstract values is restricted to { *Zero*, *Pos*, *AnyInt* }.

Suppose we add to the language an operator "<<" (left shift). Many languages (e.g., C/C++ and Java) have such operators. Given two unsigned-int values v and w, the value of v << w is equal to $v*(2^w)$. Show the abstracted semantics of the left shift operator working on two abstract values. Your definition should try to capture the most precise information that can be inferred from the operands. For example, saying that the result in all cases is *AnyInt* is correct but not accurate enough.

Q2 (4 points): Consider the following three-address code:

```
x=1
y=1
z=0
L1: if (x>y) goto L2
t1=z+x
z=t1
goto L3
L2: t2=z+y
z=t2
L3: t3=y+1
y=t3
if (y<100) goto L1
t4=x+1
x=t4
if (x<100) goto L1
```

This program contains 15 three-address instructions. Note that labels L1, L2, and L3 are not instructions.

Part 1. Among these 15 instructions, identify the ones that are leaders of basic blocks.

Part 2. Show the control-flow graph for this program. Label the basic blocks B1, … and show which three-address instructions are in each basic block.

Part 3. Show the dominator tree for this control-flow graph. There is no need to show the three-address instructions again.

Q3 (2 points): Consider the following fragment of Java code:

```
class Test {
  public static void main(String[] a) {
    int N = 5000;
    int res = 0;
    for (int i = 0; i < N; i++)
      for (int j = 0; j < N; j++) {
        res += (i+j);
        res -= 17*(res/17);
      }
    System.out.println(res);
  }
}
```

Part 1. Execute "java Test.java" to compile and run this program. Write an equivalent program for the language from Project 3 and execute it on your Project 3 interpreter. Make sure the printed values are the same. As an additional test case, change N to be 10000 in both programs and again make sure that your program (executed by your interpreter) prints the same value as the Java program. Include your program in the homework solution.

Part 2. Perform experiments with N=10000, 15000, 20000, 25000. For each value of N, run the Java program using "time java Test.java" and record the running time (just record "real" time, in seconds). For each value of N, run your interpreter on your program using "time ./plan t" and record the running time, similarly to above.

Run each experiment 3 times. For each configuration, take the median of the 3 values. Record these results in a 4 x 2 table, where each row corresponds to a value of N and shows the two running times – one for the Java program and one for your program. (Note that this comparison is more-or-less fair, since in both scenarios we include parsing+execution time.)