Analysis and Verification of Software
Homework 1

due by February 10, 2015

Exercise 1

For the code below apply the following code transformations: constant propagation, constant folding, copy propagation, dead-code elimination and strength reduction.

```
     t1 = t1 + 1
L0:  t2 = 0
     t3 = t1 * 8 + 1
     t4 = t3 + t2
     t5 = t4 * 4
     t6 = *t5
     t7 = FP + t3
     *t7 = t2
     t8 = t1
     if (t8 > 0) goto L1
L1:  goto L0
L2:  t1 = 1
     t10 = 16
     t11 = t1 * 2
     goto L1
```

Exercise 2

Create a small code example in C which reduces to:

```
return 100
```

after the following optimizations are applied (only once) in this order:

1. Common Subexpression Elimination
2. Copy Propagation
3. Dead Code Elimination
4. Constant Folding
5. Constant Propagation
6. Constant Folding
7. Constant Propagation
8. Dead Code Elimination

Show the code before and after each optimization step.

Each optimization should have some real impact (i.e.: your initial program should not just be return 100!!).

Hint: work backwards.

Exercise 3

Identify the basic blocks of the following program, that that reads two numbers and prints their greatest common divisor as long as both numbers are positive. Then draw the control flow graph of the program

```
x = read();
y = read();
while (x != y) {
  if (x > y)
    x = x - y;
  else
    y = y - x;
}
print(x);
```

Exercise 4

Consider the following function written in C:

```
int g(int x) {
    int z = p(x);
    int y = q(x);
    return y;
}
```

- Is there any dead code in this function, and if so where?
- Given that the operation of p and q are unknown, or their analysis is undecidable, comment on the safety of performing a dead code elimination phase.