# CS 480: Compiler Design
# Problem Set 4

Due: 07/15/2017

**Instructions:**
I leave plenty of space on each page for your computation. If you need more sheet, please attach your work right behind the corresponding problem. If your answer is incorrect but you show the computation process, then partial credits will be given. It is prefered that you type up your solution (you can use the DocHub extension on Google Drive to edit pdf files and insert images without using LaTeX) and upload your solution onto blackboard for grading.

**First Name**:

**Last Name**:

**Score**: **/100**

## Problem 1 Three Address Code: 20pts

We know that the three in three-address code refers to the number of operands in any instruction. Please translate the following into TAC:
(a) int x; int y; bool b1; bool b2; bool b3;
$b1 = x * x < y$ b2 = x + x ¡= y
b3 = x / x ¿ y $b4 = -x$

(b) void main() {
int x, y;
int m2 = x * x + y * y * y;
while $(m2 > 5)$
{ m2 = m2  x;} }

## Problem 2 IR Optimization: 40pts

Consider the following basic block, in which all variables are integers.

1. a := f * f
2. b := 1 * f + 0
3. c := 7 * 7
4. d := b + c
5. e := f * f
6. g := a + d
7. x := e + d
8. k := b * f
9. h := g * x
10. y := h + k

Assume that the only variables that are live at the exit of this block are x and y, while f is given as an input. In order, apply the following optimizations to this basic block. Show the result of each transformation. For each optimization, you must continue to apply it until no further applications of that transformation are possible, before writing out the result and moving on to the next.

(A)

1. Arithmetic simplification
2. Copy propagation
3. Common sub-expression elimination
4. Constant folding
5. Copy propagation
6. Dead code elimination

(B)When you have completed the last of the above transformations, the resulting program will still not be optimal. What optimization(s), in what order, can you apply to optimize the result further?

(A)

(B)

## Problem 3 Semantics: Scope Checking: 20 (14+6) pts

In semantics analysis, it is important to check the variable scope and the type compatibility between variables. For the vraible scope, the most commonly used approach is symbol table. Given the following code:

```
public class Base {
    public int value = 1;
}

public class Derived extends Base {


    public void doSomething() {
        int value = 3;
        System.out.println(value);
        System.out.println(this.value);
        System.out.println(super.value);
    }
}
```

(a) Please generate the corresponding variable tables to trace the scope of the variables

(b) Please predict the output

## Problem 4 IR Generation: 20 (5+5+10) pts

Intermediate Representation (IR) Generation is the last product that compiler front-end generates to pass to the back-end. There are many important reasons that IR Generation is desirable for the compiler back-end.
(a) What do we do IR generation (check week 6-2 slides)

(b) What is a spaghetti stack ?

(c) What is a runtime stack? How is it used to optimize the spaghetti stack ?