Midterm Exam
CS6013

Maximum marks = 50

28-Sep-2012

1. [10] **Semantic analysis**

   Present a scheme to generate IR code for an extension of Minijava that allows C-style "unions". A union can have multiple fields. But all of them share the same space.

   Your translation scheme should generate code such that, the following code prints 1.

```
union threeInOne{
  int a;
  boolean b;
  int set1(int p) { a = p; return a;}
  boolean set2 (boolean q) {b = q; return b;}
  int get1() { return a;}
  int get2() { return b;}
}
...
int foo(){
  x = new threeInOne();
  x.set1(1);
  if (x.get2()) {System.out.println(1);}
  ...
}
```

   Use your scheme to generate intermediate code for the above example (generate intermediate code for all the functions. Use an imperative intermediate code, something like miniIR). Be liberal with your comments to help us understand your translation.

2. [15 = 4 * 3.5] **Control flow analysis**

   a) We have defined two differnt relations *dom* and *pdom* separately. Why not keep only one, and use one to define the other? Say, for instance, we define: for each pair of nodes in the CFG, *a dom b* iff *b pdom a*. Is there any problem? What type of CFGs can we reasons in such a scenario?

   b) Prove that dominator relation is transitive, anti-symmetric

c) Prove that, If a and b are two dominators of n, then either a *dom* b or b *dom* a.

d) Prove that, each node except the *entry* has an unique immediate dominator.

3. [15] **Iterative Data flow analysis**

   Present an algorithm to identify uses of uninitialized variables. (Extend the conditional constant propagation algorithm)

   In the following code `print` uses an initialized variable (hence not flagged).

```
int a;
int y;
y = 3;
if (y == 3)
  a = 2;
print (a);
```

   The following code should be flagged for using an uninitialized variable.

```
int a;
int y;
y = 3;
if (y == unknownFunc())
  a = 2;
print (a);
```

4. [10 + 5 (bonus)] **Structural analysis**

   Consider a language with the following constructs: sequencing, if-then, and while loop. In the lectures, for each of these constructs we looked at the possible regions (as graphs) and used those graphs to generate control tree from CFG. We will add a new construct to this language "break" (with usual Java semantics).

   a) Redraw the graphs for the language constructs. Draw the CFG and the control tree for the following program:

```
while (cond) {
  S1;
  if (cond2) {
    S2;
    if (cond3) break;
    S3;
  }
  S4;
}
```

   b) [Bonus] Write equations to generate the flow function for a region representing the while loop in this extended language.