CS 6340 – Fall 2009 – Problem Set 2 **Name**_____

Assigned: August 25, 2009
Due: September 1, 2009

At the beginning of class on the due date, submit your neatly presented solution with this page stapled to the front (70 pts).

1. Given the following program (in a Pascal-like language) and the statement-based control-flow graph for that program (which you created in Problem Set 1):

```
 procedure sqrt(real x):real
    real x1,x2,x3,eps,errval;

    begin
1.    x3 = 1
2.    errval = 0.0
3.    eps = .001
4.    if (x <= 0.0)
5.       output("illegal operand");
6.       return errval;
7.    else
8.       if (x < 1)
9.          x1 = x;
10.         x2 = 1;
11.      else
12.         x1 = eps;
13.         x2 = x;
14.      endif
15.      while ( (x2-x1) >= 2.0*eps )
16.         x3 = (x1+x2)/2.0
17.         if ( (x3*x3-x)*(x1*x1-x) < 0)
18.            x2 = x3;
19.         else
20.            x1 = x3;
21.         endif;
22.      endwhile;
23.      return x3;
24.   endif;
25. end.
```

a. Compute the reaching definitions for each node in the control-flow graph: create a table that shows the initial values of the IN, OUT, GEN, and KILL sets, along with the results of the first and second iterations of the reaching-definition data-flow analysis algorithm (10).
b. Compute the reachable uses for each node in the control-flow graph: create a table that shows the initial values of the IN, OUT, GEN, and KILL sets, along with the results of the first and second iterations of the reachable-use data-flow analysis algorithm (10).
c. Compute the definition-use pairs (du-pairs) for the program: create a table that shows, for each node that contains a use, the du-pairs that involve that use (5).
d. Compute DU-Chains and UD-Chains for the program (5).

2. Given the following C program and the statement-based control-flow graph for the program (which you created in Problem Set 1):

```
    main()
    {
       int sum, i, j;

1.     sum = 0;
2.     i = 1;
3.     while (i <= 5) {
4.         scanf("%d",&j);
5.         if (j < 0)
6.               continue;
7.         sum = sum + j;
8.         if (sum > 10)
9.               break;
10.        i = i + 1;
11.    }
12.    printf("sum is %d", sum);
    }
```

a. Construct the control-dependence graph (CDG) without regions for the program; show all steps in the construction using the Ferrante, Ottenstein, Warren algorithm (15).
b. Add regions to the CDG from (a); show all steps in this addition (5).

3. It is often desirable to keep data-flow information from one version of a program to the next, and consider the changes to incrementally update the information for the new version of the program. For example, when a program is optimized, the data-flow changes. We can compute the data-flow for the unoptimized program, and then, given the locations and types of changes, we can incrementally update the old data-flow information to get the data-flow for the optimized program. As another example, we can use a similar process to get the data-flow for a program after it has been changed because of user edits.

   You are to develop an algorithm computes reaching definitions incrementally. In developing your algorithm, answer the following questions (in the remainder of the discussion, *old program* refers to old, original program and *new program* refers to the new, changed, modified program):

   a. Your algorithm should not require the entire reaching-definition solution for the old program for use in computing the reaching definitions for the new program. What information would you need to store about the old program for use in the analysis (5)?
   b. What method would you use to update the information about the old program to get the reaching definitions for the new program (5)?
   c. Use Version 1 and Version 2 of the following program to illustrate your techniques (10).

```
Program Version1
1.   read (x)
2.   x1 = .0001
3.   x2 = x
4.   while (x2-x1) >= .0002
5.       x3 = (x1+x2)/2
6.       if (x3*x3-x) * (x1*x1-x) < 0
7.           then x1 = x3  ← Version2, change to x2 = x3
8.       else
9.            x1 = x2      ← Version2, change to x1 = x3
10.      endif
11.  endwhile
12.  print (x)
13.  print (x2)
14.  print (x3)
end Version1
```