CS 553 Graduate Compilers	Final Examination
Fall 2009	December 14, 2009

Name:

The total number of possible points is 100. **Please think before you write,** so that your answers can be brief and fit in the space provided.

Total points: 100, 25% of the course grade. Good luck!
75 minutes (maximum)
Closed Book

- You may use two sides of one sheet (8.5x11) of paper with any notes you like.
- This exam has 12 pages, including this cover page. Do all your work on these exam sheets.
- Show all your work if you wish to be considered for partial credit.

Question	Points	Score
1	15	
2	15	
3	20	
4	15	
5	15	
6	20	
7	5	

Name: _____

Email: _____

DO NOT TURN TO NEXT PAGE TILL YOU GET PERMISSION

1. [15 points] Control flow and loops. For the given 3-address code (DO NOT simplify the code),

Dominators

```
        z = read()
        x = read()
        y = read()
        a = read()
        b = read()
        j = 0


loop:   if j>0 goto end


        a = x + y
        z = a + 1


        if (j mod 2) == 0 goto else

        y = 25
        goto endif


else:   b = y + x
        z = b + 1


endif:  j = j + 1
        goto loop


end:    print( a, b, j, x, y, z)
```

a) Draw the control-flow graph (preferably in place). Label each node in the graph.

b) Calculate the set of nodes that dominate each node and write them in the dominators column next to the program.

c) Label each of the following (there can be more than one of each): pre-header, header, back edge and exit edge.

d) Indicate the loop(s) as set(s) of nodes.

(blank intentionally)

2. [15 points] SSA and Program Optimization.

   (a) [5 points] Convert the program in problem (1) to SSA. DO NOT use Briggs or Pruned for placing phi functions, just use the basic "minimal" rule.

(b) [10 points] Perform pessimistic global value numbering on the SSA representation followed by copy propagation and dead code elimination. Indicate value numbers for each subscripted variable, indicate SSA variables with the same value, and then modify uses and cross out definitions to show the effect of the global value number optimization. Assume that $hash(+, \#p, \#q) = hash(+, \#q, \#p)$ and that $\phi(\#p, \#p) = \#p$.

3. [20 points] Data dependence analysis and loop transformations

```
for (i=1; i<=4; i++) {
  for (j=1; j<=4; j++) {
       B[i,j] = B[i-1, j+1] + y * i;
  }
}
```

(a) [5 points] For the above loop nest, set up the data dependence problem for calculating the dependence vector (show the problem setup) and solve for the dependence vector. Let the write `B[i,j]` be in iteration space (i, j) and read `B[i-1,j+1]` be in iteration space (i', j'). What is the dependence type?

(b) [5 points] Draw the iteration space with data dependences for the above program.

(c) [5 points] Write the specification for the transformation that makes the loop nest fully permutable using either the unimodular framework or the Kelly/Pugh framework. Show the effect of the transformation on the polyhedral specification of the loop. What is the name of the needed transformation?

(d) [5 points] Write a tiling specification using the Kelly/Pugh framework with tile sizes of 2 by 2. Apply the transformation to iteration space resulting from the transformation in (c) and show the final loop polyhedron (note that a polyhedron needs to have affine constraints). You do NOT have to generate code.

4. [15 points] Synchronization-free affine partitioning

(a) [8 points] For the *original* program in question 3, calculate an affine partitioning that results in synchronization-free parallelism on a one-dimensional array of processors. The affine mapping for the statement is shown below. Solve for the values C_{11}, C_{12}, and c_1.

$$p = \begin{bmatrix} C_{11} & C_{12} \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix} + \begin{bmatrix} c_1 \end{bmatrix}$$

(b) [7 points] Determine the range of p and write a loop that iterates over all possible processors and has each processor execute only the points that are mapped to it. (Hint: Use FM.)

5. [15 points] Alias/Pointer Analysis (flow and context sensitivity)

FSCS **FICI**

```
    main() {
      int **d, *r, *s, *c, *x;
      int e, f, a, b;

S1    c = &f;

S2    c = &e;

S3    r = goo(c, &a);

S4    s = foo(c, &b);

S5    d = &c;

S6    *d = &a;

    }

int * goo(int *t, int *v) {
    if ( read() > 3) { return t; }
    else { return v; }
}

int * foo(int *x, int *y) {
    return y;
}
```

For the above program, perform flow-sensitive, context-sensitive alias analysis (FSCS) and flow-insensitive, context-insensitive alias analysis (FICI). Show you work in the space provided next to the program.

(a) For FSCS, what is the points-to set for the variables `c`, `r`, and `s` after statement S6?

(b) For FICI, what is the points-to set for the variables `c`, `r`, and `s` for the whole program?

6. [20 points] Data-flow analysis with Datalog

```
S1:  b = 2;

S2:  c = 4;

S3:  if (t==0) {

S4:    a = 3; }

S5:  else {

S6:    a = 5; }

S7:  print(a, b, c);
```

The final goal is to perform reaching constants analysis on the above C snippet using datalog.

(a) [5 points] First write the datalog program that specifies reaching *definitions* analysis. Make sure to indicate the meaning of each atom.

(b) [5 points] What will the output of the example C program be? (Perform reaching *constants* by hand).

(c) [10 points] Write down all of the input and output datalog relations for the given C program using your formulation of reaching *definitions* and describe how you could use the reaching definitions results to perform reaching constants analysis.

7. [5 points extra credit] Suggest a project for next year's CS 553 that is different from the projects you were assigned or asked about in the homeworks.

   (a) Describe the project in one paragraph.

   (b) What concept will the students learn in more depth by doing the project?

   (c) What experimental results should be presented in the project report?