ECE 573 – Final Exam
May 4, 2011

Name: _____

Purdue Email: _____

Please sign the following:

I affirm that the answers given on this test are mine and mine alone. I did not receive help from any person or material (other than those explicitly allowed).

X _____

Part 1:	/30
Part 2:	/35
Part 3:	/10
Part 4:	/25
Total:	/100

Part 1: Dataflow analysis (30 pts)

In this problem, you will design a dataflow analysis that detects *memory leaks*. We will define memory leaks as allocations (*e.g.*, `int * x = malloc( ... )`) that do not have corresponding deallocations (*e.g.*, `free(x)`). We want to declare a memory leak if, for any allocation, there exists some path through the program where the variable is not freed. Note that if a variable is allocated twice, this should count as a memory leak, as well. We want to detect the leak *at the point of allocation*. This means at the allocation point, we want to know whether this particular allocation will be freed.

Problem 1 (5 pts): This is a backward bitvector analysis. Explain what information we need to track for each variable at each program point.

Problem 2 (1 pts): Does this analysis use meet or join at merge points?

Problem 3: For each statement below, give the **gen** and **kill** sets:

A) `x := y * z` (2 pts)

B) `x := malloc(sizeof(int))` (2 pts)

C) `free(x)` (2 pts)

Problem 4 (5 pts): Explain how a compiler would use this analysis to detect memory leaks.

Problem 5 (6 pts): Draw the (statement-level) CFG for the following piece of code:

```
1: x := malloc( ... );
2: y := malloc( ... );
3: free(x);
4: x = malloc( ... );
5: x = malloc( ... );
6: if ( x == null ) goto 3;
7: free(y);
8: free(x);
```

Problem 6 (7 pts): Show the IN sets your analysis would compute for each of the statements in Problem 5. Place a check mark beside each statement your analysis would determine is a potential memory leak.

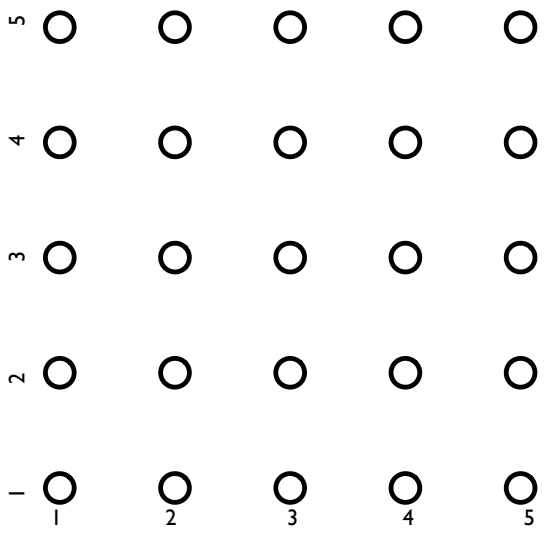
STMT	IN
`1: x := malloc( ... );`	
`2: y := malloc( ... );`	
`3: free(x);`	
`4: x = malloc( ... );`	
`5: x = malloc( ... );`	
`7: free(y);`	
`8: free(x);`	

Part 2: Depedence analysis and loop optimization (35 pts)

For the next four problems, consider the following loop:

```
for (int i = 1; i < 6; i++) {
  for (int j = 1; j < 6; j++) {
     A[i+1][j-1] = A[i][j] + A[i+1][j+2];
  }
}
```

Problem 1 (10 pts): Below is the iteration space graph for the loop nest (i along the x-axis, j along the y-axis). Draw the dependence arrows that arise from analyzing the loop. Use solid arrows for flow dependences, arrows with a slash through them (as in class) for anti-dependences, and arrows with a circle over them for output dependences.



Problem 2 (2 pts): List the distance vectors that arise from this loop nest, and mark whether they are flow, anti or output.

Problem 3 (2 pts): List the direction vectors that arise from this loop nest.

Problem 4 (2 pts): Can the loops in this nest be interchanged? Why or why not?

Problem 5 (10 pts): Give an example of a loop where the GCD test will say there *is* a dependence in the loop, but there actually is no dependence in the loop. Explain your answer.

For the next two problems, consider the following loop:

```
for (int i = 0; i < N; i++) {
A[i] = B[i]; //S1
A[i + 1] = C[i]; //S2
}
```

Problem 6 (5 pts): Can loop fusion be performed on this loop? Why or why not? Your answer should be explained in terms of dependences (or lack thereof) between S1 and S2.

Problem 7 (4 pts): Consider the same loop with statements S1 and S2 swapped. Can loop fusion be performed on *this* loop? Why or why not?

Part 3: Loop optimizations (10 pts)

For the following two problems, consider the following piece of code:

```
     x = 0;
L1:  if (x < 10) goto L2;
     z = 2 * x;
     y = 5 * x + 2;
     x = x + 1;
     goto L1;
L2:  halt;
```

Problem 1 (5 pts): Show what this code would look like after strength reduction:

Problem 2 (5 pts): Show what the code from Problem 1 would look like after linear test replacement.

Part 4: Review (25 pts)

Cam Piler things the following grammar is LR(0) but not LL(1). Your goal is to prove it.

$$
\begin{array}{lcl}
1. S & \rightarrow & A\$ \\
2. A & \rightarrow & xAy \\
3. A & \rightarrow & xAz \\
4. A & \rightarrow & z
\end{array}
$$

Problem 1 (10 pts): Show the predict sets for each rule, and use them to argue that the language is not LL(1). Show your work (first and follow sets) for partial credit.

Problem 2 (10 pts): Build the CFSM for the above grammar, and use it to argue that the language is LR(0).

Problem 3 (5 pts): Explain why performing register allocation (going from an infinite number of registers to just a few) can make instruction scheduling worse.