CS 380C: Sample Mid-semester Exam
Spring 2025

Name:

UT EID:

```
Problem 1 (20 points max): _______

Problem 2 (15 points max): _______

Problem 3 (10 points max): _______

Problem 4 (25 points max): _______

Problem 5 (30 points max): _______

Total (100 points max):    _______
```

1. (Grammars, 20 points)) Consider the following grammar

   $S \rightarrow Z\ \$$
   $Z \rightarrow d \mid XYZ$
   $Y \rightarrow \epsilon \mid c$
   $X \rightarrow Y \mid a$

   In these productions, ϵ is the empty string, and the terminal symbols come from the set $\{a,c,d,\$\}$. S is the start symbol.

   (a) (5 points) Compute the NULLABLE relation for the grammar.

   (b) (5 points) Compute the FIRST relation for the grammar.

   (c) (7 points) Compute the FOLLOW relation for the grammar.

   (d) (3 points) Use these relations to determine whether the grammar can be parsed by a recursive-descent parser.

   Explain each answer briefly.

   (a) NULLABLE = $\{X,Y\}$

   (b) i. FIRST(Y) = $\{\epsilon,c\}$
       ii. FIRST(X) = $\{a,\epsilon,c\}$
       iii. FIRST(Z) =$\{d,a,c\}$
       iv. FIRST(S) = $\{d,a,c\}\}$

   (c) i. FOLLOW(Y) = $\{d,a,c\}$
       ii. FOLLOW(X) = $\{a,d,c\}$
       iii. FOLLOW(Z) =$\{\$\}$

   (d) No. For example, FOLLOW(Y) includes c, so you don't know which production of Y to pick if the look-ahead symbol is c.

2. (Optimization, 15 points)

```
   dp = 0.0
   i = 0
L: t1 = i*8
   t2 = A[t1]
   t3 = i*8
   t4 = B[t3]
   t5 = t2*t4
   dp = dp + t5
   i = i+1
   if i<n goto L
```

The intermediate code shown above computes the dot product of two vectors A and B. The notation `A[t1]` stands for what you would expect: *load the contents of the memory location whose address is the sum of the starting address of array A and the contents of t1*.

(a) (5 points) Explain the following optimizations in a few sentences each, using the intermediate code to illustrate your answers: common subexpression elimination, strength reduction, induction variable elimination.

i*8 is computed twice so it is an opportunity for common subexpression elimination.

strength reduction: i is an induction variable incremented by 1 each time through the loop so the expression i*8 can be strength reduced to t1 = t1 + 8.

induction variable elimination: t1, i, t3 are induction variables so we can eliminate two of them.

(b) (10 points) Use these and any other relevant optimizations to optimize the code shown above. Hint: it might help to think first of how you would write a highly optimized dot product in this notation.

```
   dp = 0.0
   t1 = 0
   bound = n*8
L: t2 = A[t1]
   t4 = B[t1]
   t5 = t2*t4
   dp = dp + t5
   t1 = t1+8
   if t1<bound goto L
```

3. (Dataflow analysis, 10 points) In this problem, you need to formulate dataflow equations to solve certain problems. You need to specify (i) the domain, (ii) whether the problem is forward or backward, (iii) the dataflow equation for an assignment statement, (iv) the confluence operator, (v) whether we need to compute the least or greatest solution, and (vi) how to initialize the unknowns in the set of dataflow equations.

   (a) (5 points) A variable is said to be possibly uninitialized if there is a path in the control-flow graph from START to a use of that variable that does not pass through a definition of that variable. Formulate the problem of computing the set of possibly uninitialized variables of a procedure as a dataflow analysis problem.

   (b) (5 points) A variable is said to be definitely uninitialized if all paths in the control-flow graph from START to a use of that variable do not pass through a definition of that variable. Formulate the problem of computing the set of definitely uninitialized variables of a procedure as a dataflow analysis problem.

(a)
- Domain = power-set of variables in program
- Forward-flow problem
- OUT[START] = set of all variables in program
- assignment: x = e KILL = $\{x\}$ GEN = $\{\}$
- Confluence operation = union
- Find least solution

(b)
- Domain = power-set of variables in program
- Forward-flow problem
- OUT[START] = set of all variables in program
- assignment: x = e KILL = $\{x\}$ GEN = $\{\}$
- Confluence operation = intersection
- Find greatest solution

4. (Control dependence, 25 points)

   (a) (8 points) Compute the dominance relation for the control-flow graph shown below. Your answer should be in the form of a table as shown in lecture.

   (b) (8 points) Compute the post-dominance relation for the control-flow graph shown below.

   (c) (9 points) Compute the (Pingali/Bilardi version) control-dependence relation for this control-flow graph. Your answer should be in the form of a table in which the rows are CFG edges and the columns are vertices as discussed in class.

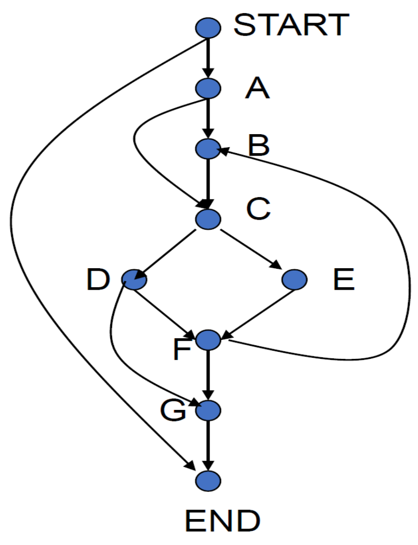

Figure 1: A control-flow graph

(a)
- dom(START) = {START}
- dom(A) = {START,A}
- dom(B) = {START,A,B}
- dom(C) = {START,A,C}
- dom(D) = {START,A,C,D}
- dom(E) = {START,A,C,E}
- dom(F) = {START,A,C,F}
- dom(G) = {START,A,C,G}
- dom(END) = {START,END}

(b) Compute the post-dominance relation for the control-flow graph shown below.

- pdom(START) = {START,END}
- pdom(A) = {END,C,G,A}
- pdom(B) = {C,B,G,END}
- pdom(C) = {END,G,C}
- pdom(D) = {END,G,D}
- pdom(E) = {END,F,G,E}
- pdom(F) = {END,G,F}
- pdom(G) = {END,G}
- pdom(END) = {END}

(c) (9 points) Compute the (Pingali/Bilardi version) control-dependence relation for this control-flow graph. Your answer should be in the form of a table in which the rows are CFG edges and the columns are vertices as discussed in class.

- – cd(START→A) = {A,C,G}
  – cd(START→END) = {}
  – cd(A→B) = {B}
  – cd(A→C) = {}
  – cd(C→D) = {D}
  – cd(C→E) = {E,F}
  – cd(D→F) = {F}
  – cd(D→G) = {}
  – cd(F→G) = {}
  – cd(F→B) = {C,B}

5. (Cultural knowledge about compilers, 30 points)) Answer each of the following questions briefly.

   (a) (4 points) What was the first high-level programming language? Who is credited with leading the team that built the compiler for this language?

   FORTRAN, Backus

   (b) (4 points) If a programming language does not support recursion, we do not need a stack to implement it. True or false? Explain briefly.

   True. You can statically allocate a frame for each procedure. If you have recursion, you need a frame per procedure invocation, so you need a stack.

   (c) (4 points) Name two optimizations that are usually done using the abstract syntax tree representation of programs. Name two optimizations that usually done using a lower-level representation like 3-address code representation.

   Loop interchange, in-line expansion of functions. Common subexpression elimination, strength reduction.

   (d) (2 point) Roughly how many compiler passes are executed by modern optimizing compilers when compiling programs? Pick one of the following: (i) between 1 and 10, (ii) between 10 and 100, and (iii) between 100 and 1000.

   Between 10 and 100.

   (e) (2 points) In the context of compilers and programming languages, what is the difference between a language and a grammar?

   Language is a set of strings. Grammar for language is a finite set of rules for generating strings in language.

   (f) (2 points) What is an ambiguous grammar? Is the following grammar ambiguous (E is the start symbol and int is a terminal)? Explain your answer briefly.

   $E \rightarrow int \mid (E + E) \mid E + E$

   A grammar is ambiguous if a string in the language can have more than parse tree. Grammar is ambiguous. Consider 2+3+4.

   (g) (2 points) In recursive-descent parsing, FOLLOW sets are needed only if the grammar has ϵ-productions. True or false? Explain your answer briefly.

True. If there are no ϵ-productions, we can use FIRST sets to determine which production to apply for a given non-terminal, look-ahead symbol combination.

(h) (2 points) In the context of programming languages, what is aliasing? Why does it complicate program analysis? Explain in three or four sentences.

Two names for same memory location. You cannot determine defs and uses by looking only at names.

(i) (4 points) In the context of dataflow analysis, what is meant by a confluence operator? Explain how you would use the confluence operator to determine whether to compute the greatest fixpoint or least fixpoint in a dataflow problem.

Confluence operator is used to merge values from domain at control-flow points where paths come together. If confluence operator is join, you compute least solution, otherwise greatest solution.

(j) (4 points) Consider live variable analysis. Show a simple program whose dataflow equations have different least and greatest solutions. Explain briefly which of these solutions you would compute for live variable analysis.

```
z = 0
y = 0
x = y
while p(z) {z = x + z}
end
```

At the output of z=x+z, live = {x,z}. This is given by least solution. However there is another solution {x,y,z} in which y is spuriously assumed to be live in the loop, which is the greatest solution to the dataflow equations. We would compute the least solution.