Compilers (CS362 2002F)

Final Examination

Due: 11 a.m., Friday, 20 December 2002

Preliminaries

There are eleven problems on the exam. Some problems have subproblems. You must select ten of the problems to complete. Each full problem is worth ten points. The point value associated with a problem does not necessarily correspond to the complexity of the problem or the time required to solve the problem.

This examination is open book, open notes, open mind, open computer, open Web. However, it is closed person. That means you should not talk to other people about the exam. Other than that limitation, you should feel free to use all reasonable resources available to you. As always, you are expected to turn in your own work. If you find ideas in a book or on the Web, be sure to cite them appropriately.

Although you may use the Web for this exam, you may not post your answers to this examination on the Web (not now, not ever). And, in case it's not clear, you may not ask others (in person, via email, by posting a "please help" message, or by any other means) to put answers on the Web.

This is a take-home examination. You may use any time or times you deem appropriate to complete the exam, provided you return it to me by the due date. It is likely to take you about ten hours, depending on how well you've learned topics and how fast you work. You will receive five points of extra credit if you indicate the amount of time each problem takes you. I expect that someone who has mastered the material and works at a moderate rate should have little trouble completing the exam in a reasonable amount of time.

You must include both of the following statements on the cover sheet of the examination. Please sign and date each statement. Note that the statements must be true; if you are unable to sign either statement, please talk to me at your earliest convenience. Note also that "inappropriate assistance" is assistance from (or to) anyone other than myself or our teaching assistant.

1. I have neither received nor given inappropriate assistance on this examination.
2. I am aware of no other students who have given or received inappropriate assistance on this examination.

Because different students may be taking the exam at different times, you are not permitted to discuss the exam with anyone until after I have returned it. If you must say something about the exam, you are allowed to say "This is among the hardest exams I have ever taken. If you don't start it early, you will have no chance of finishing the exam." You may also summarize these policies. *You may not tell other students which problems you've finished.*

You must both answer all of your questions electronically and turn them in hardcopy. That is, you must enter all of your answers on the computer, print them out, and hand me the printed copy. You must write your name on the top of *every* page of the printed exam. You must also email me a copy of your exam by copying your exam and pasting it into an email message. Put your answers in the same order as the problems.

Just as you should be careful and precise when you write code, so should you be careful and precise when you write prose. Please check your spelling and grammar. I am likely to penalize you for bad spelling and grammar.

I will give partial credit for partially correct answers. You ensure the best possible grade for yourself by emphasizing your answer and including a *clear* set of work that you used to derive the answer.

Although the problems are numbered sequentially, you need not do them in order. In fact, experience shows that students tend to do better if they are willing to put aside problems that give them difficulty, work on other problems, and then come back to the difficult problems later.

I may not be available at the time you take the exam. If you feel that a question is badly worded or impossible to answer, note the problem you have observed and attempt to reword the question in such a way that it can be answered. If it's a reasonable hour (before 10 p.m. and after 8 a.m.), feel free to try to call me in the office (269-4410) or at home (236-7445). You can also send me electronic mail.

I will also reserve time at the start of classes next week to discuss any general questions you have on the exam.

Problems

Answer ten of the following eleven problems.

Problem 1: A DFA for C Comments

Draw a DFA that accepts C comments. (You do not need to support nested C comments.) In case you've forgotten, C comments start with a slash and a star and end with a star and a slash.

Problem 2: Disambiguating Grammars

Philip and Paula Parser have written the following grammar for noun phrases in a small subset of English:

```
nounphrase ::= adjectives nounphrase
             |  noun
adjectives ::= adjective adjectives
             |  nothing
noun ::= program
       |  teacher
       |  college
       |  language
       |  zebra
       |  variable
       |  stripe
adjective ::= cool
            |  precise
            |  extreme
            |  liberal
            |  wicked
            |  large
            |  silly
```

While they seem happy with it, their computer science professor claims that the grammar is ambiguous.

a. Give a valid noun phrase that has at least two parse trees. Show two such parse trees.

b. Disambiguate the grammar.

Problem 3: Anonymous Arrays

Violet and Vance Vector note that some languages provide "anonymous arrays", arrays that you can create without initially naming. For example, we might decide that `[1,2,4]` is an array of the three integers 1, 2, and 4 and that `[[1.0,2.0],[1.0,3.0]]` is a two-dimensional array of floats.

a. Extend the expression grammar given below to support anonymous arrays. (Note that it is possible to consider addition, multiplication, division, subtraction, and array reference using arrays as one or both parameters, provided certain restrictions are met.)

```
expression ::= term { addop term }
term ::= factor { mulop factor }
factor ::= ID
        |  number
        |  ( expression )
        |  ID [ expression ]
number ::= INTEGER
        |  REAL
        |  sign INTEGER
        |  sign REAL
sign ::= PLUS
      |  MINUS
```

b. Write an algorithm for a type checker for this revised grammar. Your type checker should permit type promotion (inserting promotion nodes in the array as appropriate), verify that all the elements in an anonymous array have the same type (or can be promoted to the same type), and confirm that both arguments to arithmetic operations have compatible sizes.

Problem 4: Variable Parameters

In our discussion of function and procedure calls, we mostly emphasized *value* parameters, which you can implement by copying the value of the parameter onto the stack. However, Pascal also supports *variable* parameters, in which changes to the formal parameter are supposed to affect the actual parameter. As I mentioned in class, *variable* parameters are typically implemented by putting a pointer to the actual parameter on the stack.

Consider the following function and procedure:

```
function alpha: integer;
var
  i: integer;
begin
  beta(i);
  alpha := i + 1;
end;

procedure beta(var x: integer);
begin
  gamma(x);
end;

procedure gamma(var y: integer);
begin
  y := 0;
end;
```

Assume that local variables are stored at offset -4 from the frame pointer and that parameters are stored at offset +4 from the frame pointer.

a. What PAL code should be generated for the call to `beta(i)`?

b. What PAL code should be generated for the call to `gamma(x)`?

c. What PAL code should be generated for the assignment `y := 0`?

Problem 5: Variable Parameters, Revisited

Upon reading the previous problem, Charles and Charlene Compiler have decided that "variable parameters are just too hard to implement in the way that Sam suggested". In particular, they find it hard to deal with having the address of a variable on the stack. Here's their "solution":

> We'll treat variable parameters much like value parameters. That is, we'll put the value of any actual parameter on the stack whether it's a value parameter or a variable parameter. We'll also update those values on the stack. For variable parameters, it is then the responsibility of the *caller* to copy the new value back into the corresponding variable before popping the stack.

For example, here's their approximate code for the call `foo(X)` where X is a global variable and `foo` is a variable-parameter procedure.

```
PUSH <- X
CALL foo
POP -> X
```

Critique their strategy.

Problem 6: Graph Coloring

Show what happens with the greedy approximate graph coloring algorithm in a graph with ten nodes, five of which are connected to every other node and the remaining five of which are only connected to the first five. Assume that only four colors are available.

Problem 7: Translating Selection Sort

Consider the following fragments of a Pascal program for the famous *selection sort* algorithm.

```
const
  ARRAY_SIZE = 100;

type
  IntArray = array[1..ARRAY_SIZE] of integer;

var
  A: IntArray;

procedure swap(x,y: integer);
  var tmp: integer;
begin
  tmp := A[x];
  A[x] := A[y];
  A[y] := tmp;
end;
```

```
procedure selectionSort;
  var i, j, guess: integer;
begin
  for i := 1 to ARRAY_SIZE do
    begin
      guess := i;
      for j := i+1 to ARRAY_SIZE do
        if (A[j] < A[guess]) then
          guess := j;
      swap(i, guess);
    end;
end;
```

Generate the PAL code for `selectionSort`. You can write the PAL code in text format, rather than writing Java code. You can refer to the base address of `A` as `A`. You need not generate the PAL code for `swap`.

Problem 8: Optimizing Selection Sort

Optimize your PAL code from the previous problem, indicating which optimizations you are using. You need only use the basic optimizations we discussed in class. You should *not* unroll loops.

Problem 9: Copy Propagation

Henry and Hermione Hacker can only understand techniques by reading and using working code. (It is to support people like them that I created the sample Stupid code.) Since I described most of the optimizations quite abstractly, they are particularly puzzled about copy propagation.

Write a method, `copyPropagate(rebelsky.pal.Instruction[] code)`, that performs copy propagation on `code`. You can assume that `code` is a basic block. You need only handle the following instructions: `Move`, `IAdd`, `IMult`, `Jump`, `IWrite`, `Push`, `Pop`, and `JumpEqual`.

You can find the details of those instructions at
`/home/rebelsky/Web/Courses/CS362/2002F/Examples/rebelsky/pal`.

Because the fields of those classes have package protection, in order to access them you will need to make the class that holds `copyPropagate` be part of the `rebelsky.pal` package. You can do that in two ways:

- Recommended: Create your own `rebelsky/pal` directory that contains only your class (which is listed as being in `rebelsky.pal`). Due to an interesting design decision in Java, that class can then access the package-protected fields and methods of my `rebelsky.pal` classes. You will have to make sure that both the examples directory and your directory are part of your `CLASSPATH`.

- Alternative: Make a copy of my `rebelsky/pal` directory.

Problem 10: Loop Unrolling

Larry and Lana Looper are suspicious of my claim that it's useful to unroll `for` loops by duplicating the body because such duplication provides additional chances for optimization in the duplicated body.

Write a `for` loop that when unrolled in this way provides opportunities for optimization that would not be possible without the unrolling. Indicate what opportunities it provides.

Problem 11: Write Your Own

Carl and Carla Creative always ask me to add one more question that gives them the opportunity to express themselves. So, here goes ...

Write and answer your own question. It should be relevant to the material we covered in the class, cover a topic not covered in the problems already on the exam, and be of an appropriate level of difficulty (no easier than the easiest question, no harder than the hardest question).

Extra Credit

Each of the following problems is worth of modicum of extra credit.

a. Identify errors of grammar or spelling in this exam. (I'd prefer that you email them to me as you find them so that I may correct them.) I will give these points to the class as a whole and limit the total to five.

- Extra Credit: "I"d" rather than "I'd" [Various; 1 pt]
- Problem 7: "`guess = i`" rather than "`guess := i`" [Various; 1 pt]
- Problem 7: Sam declared `tmp` instead of `guess`. [EBA, 1pt]
- Introductory notes "in in" should be "in". [EBA, 1 pt]
- "which has" should be "that has" [EBA, 1 pt]
- "which contains" should be "that contains" [EBA, 0 pt]
- "some language provide" should be "some languages provide" [MF,EBA, 0 pt]
- "X is a global variables" should be "X is a global variable". [MF, 0 pt]

b. There is a subtle error in the `selection-sort` procedure. Identify it.

c. Indicate how long each problem took you.