P1:	of 10	P5:	of 10
P2:	of 10	P6:	of 10
P3:	of 10	P7:	of 15
P4:	of 10	P8:	of 25

CS 411 Final
April 24th, 2004

Exam number:

Student Name:

CPSC Login Username:

READ THIS CAREFULLY BEFORE PROCEEDING
DO NOT TURN THE PAGE UNTIL WE TELL YOU TO DO SO

CAUTION -- Candidates suspected of any of the following, or similar, dishonest practices will be immediately dismissed from the examination and will be liable to further disciplinary action:

a) Making use of any books, papers or memoranda.

b) Speaking or communicating with other candidates.

c) Purposely exposing written papers to the view of other candidates. The plea of accident or forgetfulness will not be received.

Instructions

1. Each candidate should be prepared to produce his/her student-ID.

2. READ AND OBSERVE THE FOLLOWING RULES:

- No candidate will be permitted to ask questions of the invigilators except in case of supposed errors or ambiguities in exam questions.
- No candidate will be permitted to enter the examination room after the expiration of one half hour, or to leave during the first half hour of the examination.
- No candidate shall be permitted to leave during the last half hour of the examination. Please remain in your seat until your paper has been collected.
- Smoking is not permitted during examinations.

3. You have 2 hours to complete this exam.

4. Give concise answers. Long answers will be looked down upon with disregard - you will get points deducted for fluff. Put all your answers on the exam. Do any rough work on the back of the exam pages. All rough work must be handed in.

1) ASTs (10 points)

Given this fragment of a mini Java method

```
    int x, y = 1;
    int x;
```

there are several reasonable AST structures we could use to represent it. Show at least two, describe the difference, and list their relative strengths and weaknesses. You can assume the existence of a Sequence AST node class that can be used to keep a sequence of any other kind of AST node.

2) Visitors (10 points)

Sometimes we can write compiler passes that just do style checking, to warn programmers of questionable style in their code.

Implement a visitor that checks a program to find field and local variable declarations that have no initializers. You can assume the existence of the reusable Visitor infrastructure used in the class compiler.

3) Runtime Issues (10 points)

Java defines a hashCode() method on java.lang.Object. This method returns an integer that can be used in hash tables and other code that must hash on objects. The documentation of the hashCode method is somewhat subtle, but for our purposes, assume that the contract of the method is that:

- For any given object, hashCode always returns the same value.
- For two different objects, hashCode should be very likely to return different values.

A simple way to implement the hashCode method is to just return the object's address in memory. But this can have interactions with the memory management and garbage collection scheme used by the runtime. For each of the following garbage collection scheme, say whether this simple implementation of hashCode would work, and if not, explain why:

(A) mark and sweep

(B) stop and copy

4) Code Generation (10 points)

Show the jasmin code the mini Java compiler should generate for this method. Assume that “friend” is a field in the class that contains this method.

```
int foo(int x) {
    return 2 * friend.foo(x);
}
```

5) Optimization (10 points)

Following are three programs, written in three different languages you have studied this semester. Each program is amenable to at least one of the optimizations discussed in class. For each program, show the optimized code AND write the name of the optimization you applied to produce that code.

```
int sum(int [] arr) {
  result = 0;
  for(int i=0; i< arr.length; i++) {
      result += arr[i];
  }
  return result;
}
```

```
push 1
loadl 5
loadl 30
call mult
store 0[sb]
pop 1
halt
```

```
.method public foo(II)I
.limit stack 50
.limit locals 3
iload 1
iload 2
iadd
iload 1
iload 2
iadd
imul
ireturn
.end method
```

6) Extending the Compiler (10 points)

The mini Java compiler architecture is extensible, in that we can add new features to the language, in a well-defined way.

Consider adding for loop statements to your compiler. Please describe the changes you would have to make to the compiler to add this feature. For each of the following describe the salient characteristics of what you would have to do: AST, scanner/parser, name checking, type checking, code generation. You probably don’t need to write code, but you can if you want.

7) Generative Programming (15 points)

The infrastructure for visitors in the compiler is scattered across a number of files, and is also boring (very repetitious).

(A) Describe a scheme for generating this code automatically. Sketch out all the key parts, how it would work, when the code generator would have to run, where the generated code would go etc. (10 points), AND

(B) Assume you are part of a team building a large compiler. What are the pros AND cons of using such a mechanism in the compiler? (5 points)

8) Building a Parser Generator (25 points)

In this question you must consider the implementation of a parser generator, such as Java Cup. This is a tool that you could then use in the implementation of a compiler such as mini Java.

Assume you need to implement the parser generator from scratch. (You just have Java, a Java IDE and Java compiler.) Please describe the final architecture of your parser generator, as well as describing any key issues that arise during its construction. Hint: Take your time and sketch this out on scratch paper (backs of pages) first.

The best answers to this question will include material from every chapter of the book. Your answer does not need to be an essay. It can just be a list of the key points, in a coherent order, perhaps supplemented with figures and small amounts of code. You will be graded on coverage of the key issues, as well as on clarity of the overall answer.