Instructions: This is an open book, open note, 110 minute exam. There are 2 questions worth a total of 100 points. You must answer both of the questions. Put your name on all pieces of paper that you turn in. Show all of your work, write legibly and good luck.

1) (25 points) The definition of a programming language maps the text of a program onto a computation. It does this by way of a variety of rules about what constitutes a *well-formed* and *legal* program.

a) **(10 points)** Describe at least four types of rules for the JOOS or WIG language. Multiple instances of the same kind of rule are unacceptable.

b) **(5 points)** Why aren't all of those rules described in a single representation or formalism?

c) **(5 points)** What components of a compiler handle the processing of those rules?

d) **(5 points)** Why aren't all of those rules processed by a single part of a compiler?

In answering these questions be as detailed as possible and give examples where appropriate, e.g., of a kind of rule that is too "complex" for one kind of processing but for which another kind of processing is suitable.

2) (75 points) You are to add arrays to JOOS for integers and references. For example, you can define and manipulate an array of 3 integers as follows:

```
int[] ai = new int[3];
ai[0] = 1;
ai[1] = ai[0]+1;
ai[2*ai[0]] = ai[0] + ai[ai[0]];
```

The JVM supports integer arrays through `iaload` and `iastore` bytecodes. `iaload` expects an integer on the top of the stack that is used as an array index and a reference to an integer array below it; it pops those and pushes the indexed array element. `iastore` takes a value, then an index, and finally an array reference on the top of the stack; it pops them all and updates the array's value. Reference arrays are supported with the `aaload` and `aastore` bytecodes, which work in a similar way. In either case, the index expression is checked to ensure that is between 0 and the `arraylength`, which is a bytecode that takes an array reference on the top of the stack, pops that reference and pushes an integer equal to it's length; note that bounds checking is implicit in the semantics of array bytecodes and one doesn't have to explicitly encode the checks in the bytecode program. In addition, for array store byte codes the assignment compatibility of stored elements is checked.

In solving the following the questions, be sure to describe your solution conceptually first and then provide details with, e.g., code fragments, examples, etc. Describe the changes you would make to your compiler in the following areas:

a) **(10 points)** Extend the scanner to handle any new lexical elements.

b) **(10 points)** Extend the parser to handle any new syntactic elements.

c) **(10 points)** Extend the AST for arrays. Sketch the AST for the above example.

d) **(15 points)** Describe the type checking rules for arrays of object references. For full points, describe the logical type rule.

e) **(15 points)** Describe a general JVM byte-code template for array reference and assignment. Illustrate how the template is expanded for the above example.

f) **(15 points)** Describe a peephole optimization to use unchecked versions of array loads and stores. Assume that there are bytecodes `unchecked_iaload` and `unchecked_iastore` that are identical to `iaload` and `iastore`, except that they do not perform array bounds checks and therefore are faster to execute. For full points, give a peephole pattern, a transformation and illustrate how they improve the byte-codes for the example generated in the previous question.