

Put **your name** & number *on this page only*. Name \_\_\_\_\_ Number \_\_\_\_\_  
 Read this exam from beginning to end before you start. Write your answers on blank sheets of paper. Start each question on a new sheet of paper. Use both sides of the paper if necessary. Clearly **identify which question** you are answering. Put the last four digits of your **number** (**not** your name) on **every** sheet.

Closed book. Bring a copy of the Minsky Language Specification and an 8.5x11 cheat sheet of notes(both sides). No access to computers (any machine that can access the Internet or compile, interpret, check, parse, or analyze any computer language discussed in this course). No wireless communication. There are eight(8) questions needing long answers, mathematical derivations, and short essays. Each contributes a maximum of 25 points. You can get 100 %, in this exam, you must complete all 8 questions. I give credit for working, do not erase it.

### 1. Language Research Presentations Chapter 1

Show that you recall at least one presentation of a term paper made by another student in this class.

### 2. Syntax Chapter 2

Show that you understand BNF, EBNF, Chomski types, ... Parse strings according to given grammar/BNF. Recall mathematical definition of a grammar. Show how to apply productions to calculate strings in a language. Translate EBNF into BNF. Correct a simple ambiguity in BNF.

### 3. Semantics Chapter 3

Given the formal abstract syntax and semantics of an unusual language show how to determine the meaning of a program/expression/statement in the language.

### 4. Translation Chapter 4

Define and relate: lexers, parser, interpreters, compilers, quads, syntax improvement, recursive descent. Given a production write the associated subprogram/function from a typical recursive descent parser or interpreter for it

### 5. Imperative Language Chapter 5

Describe effect of Von Neumann architecture on imperative languages. Provide names of typical imperative languages. Show how a given imperative language fits the paradigm. List differences between two imperative languages. Describe features of imperative languages: I/O, arithmetic, GOTOs, control structures, loops, special commands/directives, data structures, records, pointers, runtime errors, functions and procedures, activation records and the run time stack, parameter passing. Give a rough translation of a small FORTRAN program into a language like C++ or Java with only approximate I/O formatting.

### 6. Object Oriented Languages Chapter 6

Explain why OO languages evolved: what needs did they meet? Use imperative to OO transition as an example of language evolution. Describe the features that distinguish OO from imperative. Give examples of OO features in a language like C++, Java, or C#(your choice). Compare C++ and Java. Describe 2 or 3 features in Java that serve concurrency. Give an example of a difficulty with concurrent programming that does not occur in normal programs. Show how to express a simple data structure (stack, queue,...) in an OO language. How successful was the shift from imperative to OO paradigm? Write a short essay on the future of C++(or Java) that states your opinions and gives reasons for them

### 7. Functional Languages Chapter 7

Name the most Long lasting and famous functional language, and its two main variants. Describe LISP data in terms of C++/Java. Describe S-expressions and give examples. Explain a COND function in terms of C++/Java. Given a small LISP function definition and a call trace the resulting evaluation. Describe association lists: form of data, typical operations. Describe the Property list: typical function calls. Know whether you can describe the semantics of LISP using LISP as a metalanguage. Show how to express a simple data structure (stack, queue,...) in LISP. Write a short essay on the future of LISP that states your opinions and gives reasons for them.

### 8. Logic Programming Chapter 8

Describe the propositional calculus, first order (lower) predicate calculus. Give an example of Modus Ponens, truth table, quantifier, predicate, proposition, and connectives. Give an example of a Prolog database containing facts and rules and trace how a typical query accesses these to give results. Given a Prolog program plus some queries, show what Prolog responds. Show how to express a simple data structure (stack, queue,...) in Prolog. Distinguish '=' and 'is' in Prolog -- with examples. Define and trace a Prolog recursive definition.

**CS620**

**Comprehensive Final 200pts Max**

**Spring 2004 Vn. A**