

CSE-304 Compiler Design

Fall '98

Final Exam

December 17, 1998.

Duration: 2 hours, 50 minutes.

ID Number : _____

Name : _____

INSTRUCTIONS

Read the following carefully before answering any question.

- Please fill in your ID number and Name in the space provided above.
- *DO NOT OPEN THE EXAM BOOKLET UNTIL INSTRUCTED TO DO SO!*
- The exam for undergraduate students is different from the exam for the graduate students. Make sure you are reading the correct question paper.
- Read all questions carefully at the beginning of the exam. Doubts about the questions will be answered only in the first 30 minutes of the exam.
- Write your answers in the space provided. Keep answers brief and precise.
- The exam consists of 7 questions, in 11 pages (including this page) for a total of 100 points.

GOOD LUCK!

Question	Max. Points	Points Scored
1.	10	
2.	10	
3.	15	
4.	20	
5.	14	
6.	16	
7.	15	
Total	100	

1. [10 points] HTML and L^AT_EX are two document formatting languages. We want to build a translator to convert documents structured using HTML to L^AT_EX. In HTML, the text of the document to be formatted is specified between `<html>` and `</html>`; in L^AT_EX, the text is specified between `\begin{document}` and `\end{document}`. In HTML, a list of unnumbered bullet items is given using `` and ``; in L^AT_EX, the list is specified using `\begin{itemize}` and `\end{itemize}`. In HTML, a list of numbered bullet items is given using `` and ``; in L^AT_EX, the list is specified using `\begin{enumerate}` and `\end{enumerate}`. Each item in the list begins with `` in HTML, and `\item` in L^AT_EX.

Font changing commands in HTML are specified as follows: the text to be displayed in **bold face**, *italics*, or `teletype` are marked with ``, `<I>`, and `<TT>`, respectively, in the beginning, and ``, `</I>`, and `</TT>`, respectively, in the end. In L^AT_EX, the text with changed font begins with `\bf`, `\it` and `\tt` respectively, and the change of font persists until the matching `}` is found.

A sample translation of a HTML document to L^AT_EX is shown below:

HTML:	L ^A T _E X:	Output (Formatted) Text
<pre><html> This is a sample document. A list of bullet items can be created as follows: </html> Start an itemize environment; Place each item with an item command. </html></pre>	<pre>\begin{document} This is a sample document. A list of bullet items can be created as follows: \begin{itemize} \item Start an \bf itemize} environment; \item Place each item with an \em item} command. \end{itemize} \end{document}</pre>	<p>This is a sample document. A list of bullet items can be created as follows.</p> <ul style="list-style-type: none"> • Start an itemize environment; • Place each item with an <i>item</i> command.

a. (7 points) List the smallest set of tools you will use to construct a translator to convert input HTML documents into L^AT_EX documents. Explain your choice.

You may assume that all input documents are well formed HTML documents: i.e., they are correct with respect to HTML syntax.

b. (3 points) If the input documents may not all be well formed HTML documents, will the tools you chose in part [a] above be still sufficient to construct a translator? Explain.

Undergraduate Exam

2. [Total: 10 points] BigSoft Inc., the world's largest supplier of useless software, announces a new product called aritrustr. The program aritrustr converts documents (which are plain ASCII files) as follows: each input sentence is copied to the output file, *except those sentences that contain the word competition*.

BigSoft claims that aritrustr is written completely using Lex, with no C code other than printf's. Is this possible? If so, sketch the Lex specification; if not, explain why not.

3. [Total: 15 points] The "normal" way to write arithmetic expressions is called *infix* form, since operators are sandwiched between the operands. For example, in $1 + 2 * 3$, $*$ operates on 2 and 3, and $+$ works on 1 and the result of the product.

The same expression can be written in *postfix* form, as $1 2 3 * +$: the operands precede the operators. For instance the infix expression $(2 + 4) * 6$ will be written in postfix as $2 4 + 6 *$.

a. (2 points) Write the infix expression $-(4 * 6) + 7$ in postfix form.

b. (6 points) Can every infix expression be converted to postfix? Justify.

d. (7 points) Let L_3 be a language of postfix expressions over integer constants, with $*$ and $+$ as binary operators and $'-'$ as an unary operator.

Write a grammar that describes L_3 precisely.

Undergraduate Exam

Undergraduate Exam

4. [Total: 20 points] Let G_4 be an LL(1) grammar with Σ as the set of terminal symbols and Π as the set of nonterminal symbols.

Note: For each of the following, you may state your answers in big- O notation (e.g., $O(x)$) if the quantity is linear in x .

a. (4 points) What is the maximum size of LL(1) parsing table for G_4 ? Why?

b. (4 points) What is the size of the largest parse tree produced by a grammar in CNF for a string of length n ? Why?

You may assume that G_4 has no ϵ -productions, and each production in G_4 is of the form $A \rightarrow BC$ (where B and C are nonterminals) or $A \rightarrow a$ (where a is a terminal).

c. (4 points) What is the maximum size of SLR(1) parsing table for G_4 ? Why?

Undergraduate Exam

d. (4 points) How long does it take to parse a string of length n with respect to G_4 using the LL(1) parsing algorithm? Why?

e. (4 points) How long does it take to parse a string of length n with respect to G_4 using the shift-reduce parsing algorithm with an SLR(1) table? Why?

Undergraduate Exam

5. [Total: 14 points] Consider the following grammar, G_5 , that defines expressions over integers and a special identifier x :

$E \rightarrow E + T$
 $E \rightarrow T$
 $T \rightarrow T * F$
 $T \rightarrow F$
 $F \rightarrow (E)$
 $F \rightarrow \text{int}$
 $F \rightarrow x$

The expressions generated by G_5 can be viewed as polynomials of x .

Consider the problem of differentiating the polynomials with respect to x . For instance, for an input expression x the translation should yield 1. The output need not be simplified. For example, $x*x$ may be translated into $x*1 + x*1$.

Let dx be an attribute of nonterminal symbols E , T and F that holds the derivative (i.e., $\frac{d}{dx}$) of the corresponding expression.

a. (3 points) For each of the following nonterminal symbols state whether dx is an inherited or synthesized attribute:

E :

T :

F :

b. (7 points) Write a syntax directed definition to compute the value of attribute dx for grammar

G_5 .

Hint: Use the two laws of differentiation: $\frac{d(u+v)}{dx} = \frac{d_u}{dx} + \frac{d_v}{dx}$ and $\frac{d(u*v)}{dx} = u * \frac{d_v}{dx} + v * \frac{d_u}{dx}$.

Undergraduate Exam

c. (4 points) Can the attribute dx be computed during parsing when a shift-reduce parser is used? Why or why not?

(continued on next page)

6. [Total: 16 points] Consider the following Decaf program:

```

class A {
    public int i;
    public int m(int i) { return i;}
}
class B extends A {
    public float m;
    public float i(float x) { return x;}
}
class C extends B {
    public float m(float x){ return m;}
    public int i(int i){ return i(i);}
}
class D extends C {
    float i;
    float i;
    void f(){
        int m;
        boolean i;
        A a; B b; C c; D d;
        a = new A();
        b = new B();
        c = new C();
        d = new D();
    }
    Out.print(EXAMPLE_EXPRESSION /****/);
}
    
```

The table below lists expressions that may appear in the program in place of EXAMPLE_EXPRESSION (marked with /****/). In the space provided in the table, fill the type of each expression.

Fill at least 8 entries. Each correct entry in the table is worth 2 points (maximum 16 points).

	EXAMPLE_EXPRESSION	Type
1	a.i	
2	d.i	
3	i	
4	this.i	
5	i(m)	
6	m(i)	
7	m(i*m(i))	
8	c.m(i)	
9	c.m(b.i)	
10	c.m(this.i)	

7. [Total: 15 points]

a. (7 points) Give a syntax directed definition to generate code for the for statement (C or Decaf) whose syntax is given by the following grammar rule.

$$S \rightarrow \text{for } (E ; E ; E) S$$

In the above E stands for expressions and S for statements. Assume that *code* is an attribute of E and S that represents the generated code. You may generate C/C++ code (as in the last project) or Three-address code (as in the text).

State any assumptions you may make about the attributes of S and E clearly. You may (but need not) assume that short-circuit code is generated for E .

Undergraduate Exam

b. (5 points) Consider the `continue` statement:

`S` \rightarrow `continue`

Semantics of `continue` is same as in C or Pascal, i.e., begin the next iteration of the loop.

Give a syntax directed definition to generate code for the `continue` statement. List any modification(s) to the definition in part [a] (previous page) needed to support the `continue` statement.

c. (3 points) Consider the extension of `continue` statement of the form `continue(E)`, which means: begin next iteration of loop if E is true. Sketch how you will modify the syntax directed definition of part [b] above to handle this extension.

END OF EXAM