

CSE-304 Compiler Design

Fall '09

Final Exam

December 14, 2009

Duration: 2 hours, 30 minutes.

Maximum Score: 30

Name: _____

USB ID Number: _____

INSTRUCTIONS

Read the following carefully before answering any question.

- Make sure you have filled in your name and USB ID number in the space above.
 - Write your answers in the space provided; Keep your answers brief and precise.
 - The exam consists of 8 questions, in 19 pages (including this page) for a total of **30** points. Pages 17–19 of the exam are intentionally blank.
-

Question	Max.	Score
1.	3	
2.	4	
3.	4	
4.	3	
5.	2	
6.	5	
7.	4	
8.	5	
Total:	30	

1. [3 points] $\text{\LaTeX}$ and HTML are two document formatting languages. We want to build a translator to convert documents structured using HTML into $\text{\LaTeX}$. In HTML, the text of the document to be formatted is specified between `<html>` and `</html>`. In $\text{\LaTeX}$, the text is specified between `\begin{document}` and `\end{document}`. In HTML, a list of unnumbered bullet items is given using `<ul>` and `</ul>`; in $\text{\LaTeX}$ such a list is specified using `\begin{itemize}` and `\end{itemize}`. Each item in a list of bullet items begins with `<li>` in HTML and `\item` in $\text{\LaTeX}$.

Font-changing commands in HTML are specified as follows: the text to be displayed in **bold face**, *italics* or **teletype** are marked with `<B>`, `<I>` and `<TT>`, respectively, in the beginning, and `</B>`, `</I>` and `</TT>`, respectively, at the end. In $\text{\LaTeX}$ text with changed font begins with `\textbf{}`, `\textit{}` and `\texttt{}` respectively, and the change of font ends with `}`.

A sample translation of HTML document to $\text{\LaTeX}$ is shown below:

HTML	$\text{\LaTeX}$	Output (formatted) text
<pre><html> This is a sample document. A list of bullet items can be created as follows: Start an itemize environment; Place each item with an <i>item</i> 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 \textbf{itemize} environment; \item Place each item with an \textit{item} command. \end{itemize} \end{document}</pre>	This is a sample document. A list of bullet items can be created as follows: • Start an itemize environment; • Place each item with an <i>item</i> command.

List the smallest set of tools you will use for building a translator to convert input HTML documents into $\text{\LaTeX}$ documents. Explain your choice clearly. In particular, you must justify the inclusion of each tool in the set, and show that your choice of tools is complete: i.e. sufficient to carry out the translation.

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

[THIS PAGE IS INTENTIONALLY BLANK]

2. [4 points] A grammar is said to be ϵ -free if either it has no ϵ -productions, or the only ϵ -production is $S \rightarrow \epsilon$ where S is the start symbol.

A grammar is said to be in Chomsky Normal Form (CNF) if it is (i) ϵ -free, (ii) its start symbol S does not appear on the right hand side of any production, and (iii) every production is of the form $A \rightarrow BC$ where B and C are some non-terminal symbols, or of the form $A \rightarrow a$ where a is some terminal symbol.

Let G_2 be an LL(1) grammar in CNF. Let G_2 have t terminal symbols, n non-terminal symbols and p productions.

- (a) [1 point] What is the size of the LL(1) parsing table for G_2 ? Why?

- (b) [2 points] Let s be a string of length k in $\mathcal{L}(G_2)$, i.e. the language of G_2 . What is the size of the parse tree for s ? Justify.

- (c) [1 point] Let s be a string of length k in $\mathcal{L}(G_2)$, i.e. the language of G_2 . How long does it take to parse s using the LL(1) parsing algorithm? Justify.

3. [4 points] Consider the following grammar G_3 :

$$\begin{aligned} S &\longrightarrow A \\ S &\longrightarrow B \\ A &\longrightarrow A \text{ c} \\ A &\longrightarrow \epsilon \\ B &\longrightarrow \text{c } A \text{ d} \end{aligned}$$

- (a) [1 point] Compute FIRST and FOLLOW sets for S , A and B .

- (b) [1 point] Show that G_3 is not SLR(1).

[CONTINUED ON NEXT PAGE]

[CONTINUED FROM PREVIOUS PAGE]

- (c) [2 points] Can the conflicts in the action table be resolved arbitrarily (i.e. always choosing one of the conflicting actions instead of the other) without affecting the language accepted by the parser? Explain.

4. [**3 points**] Consider the following grammar describing floating point numbers written in binary notation (with F as the start symbol):

$$F \longrightarrow N . N$$

$$N \longrightarrow N B$$

$$N \longrightarrow B$$

$$B \longrightarrow 0$$

$$B \longrightarrow 1$$

Write a **Yacc/Bison specification** of the above grammar, and use that specification to compute the *value* of a binary floating point number. For example the value of 101010.101 is 42.625.

5. [2 points] Consider a system with 4GB address space, of which 1GB (total) is allocated to stack, static area, and code.
- (a) What is the maximum allowable heap size if *Mark-and-Sweep* garbage collection is used?
 - (b) What is the maximum allowable heap size of *Copying* garbage collection is used?
 - (c) If the total size of all live heap objects is 256MB, and it takes 4 nanoseconds to inspect 4 bytes of heap for marking, sweeping or copying, how much time will be taken by a *mark-and-sweep* collector?
 - (d) If the total size of all live heap objects is 256MB, and it takes 4 nanoseconds to inspect 4 bytes of heap for marking, sweeping or copying, how much time will be taken by a *copying* collector?

6. [5 points] For this question, consider the following class definitions in a Java-like language:

<pre>class One { private int i; public int j; static int k; public One a; }</pre>	<pre>class Two extends One { public int n; public int m; }</pre>
<pre>class Three extends One { public int j; public int n; public int m; private One b; }</pre>	<pre>class Four extends Three { public int k; static One c; private int n; }</pre>

- (a) [1 point] What is the minimum amount of memory that should be allocated for an object (instance) of class `One`? Justify your answer.

Assume that `int`'s and object references take up 4 bytes of memory.

- (b) [1 point] Show the memory layout of an object (instance) of class `One`. Show each field in the object and write the offset of each field relative the base address of the object, assuming `int`'s and object references take up 4 bytes of memory.

[CONTINUED ON NEXT PAGE]

[CONTINUED FROM PREVIOUS PAGE]

- (c) [1 point] Show the memory layout of an object (instance) of class **Two**. Show each field in the object and write the offset of each field relative the base address of the object, assuming `int`'s and object references take up 4 bytes of memory.

- (d) [1 point] Show the memory layout of an object (instance) of class **Three**. Show each field in the object and write the offset of each field relative the base address of the object, assuming `int`'s and object references take up 4 bytes of memory.

[CONTINUED ON NEXT PAGE]

[CONTINUED FROM PREVIOUS PAGE]

- (e) [1 point] Show the memory layout of an object (instance) of class **Four**. Show each field in the object and write the offset of each field relative the base address of the object, assuming **int**'s and object references take up 4 bytes of memory.

7. [4 points] Consider extending Decaf with a *Do-While* statement, whose syntax is given by the following production:

$$Stmt \longrightarrow \text{do } Stmt \text{ while } Expr$$

where *Stmt* and *Expr* are non-terminals representing Decaf statements and expressions, respectively. The meaning of a do-while statement is standard: execute the body statement *one or more times*, exiting the loop when the guard expression is false.

- (a) [2 points] Write syntax-directed definition to generate code for *Do-While* statements. You may generate CREAM code (as in HW4) or generate three-address code.

[CONTINUED ON NEXT PAGE]

[CONTINUED FROM PREVIOUS PAGE]

- (b) [2 points] When generating code for *while* statements, the text book claims that it is better to consider

`while Expr Stmt`

as

`if Expr do Stmt while Expr`

The code generated for the second form above (`if ...`) is more efficient than the code generated for the first form above (`while ...`). Give a clear argument to show why the claim in the text book is valid.

8. [5 points] For this question, recall the following CREAM instructions (as in the CREAM manual, “...” represents untouched values on stack):

- **ildc(n):** $\dots \Rightarrow \dots, n$
Push integer constant n on stack.
- **load:** $\dots, n \Rightarrow \dots, v$
Push the value v of local variable n on stack. n is the offset of the local variable from the base of the current activation record.
- **store:** $\dots, n, v \Rightarrow \dots, v$
Place the value v in the local variable n . Remove n from stack but retain v on top of stack.
- **aload:** $\dots, a, i \Rightarrow \dots, v$
 a is a handle to an array in heap; the value v of the i th element of the array is retrieved and pushed on stack.
- **astore:** $\dots, a, i, v \Rightarrow \dots, v$
 a is a handle to an array in heap; value v is assigned to the i th element of the array.
- **iadd:** $\dots, i, j \Rightarrow \dots, i + j$
Add integer values of the top two elements on stack, and replace them with their sum.
- **dup:** $\dots, v \Rightarrow \dots, v, v$
Duplicate the value on top of stack.
- **pop:** $\dots, v \Rightarrow \dots$
Pop the value on top of stack.
- **swap:** $\dots, v_1, v_2 \Rightarrow \dots, v_2, v_1$
Interchange the top two elements of stack.

For each of the following Decaf expressions, give the corresponding CREAM code. Your code should only use the instructions listed above. Note that not all above instructions may be necessary.

(a) $j = i+1$ where i and j are local integer variables at offsets 2 and 3 respectively.

[CONTINUED ON NEXT PAGE]

[CONTINUED FROM PREVIOUS PAGE]

(b) $j = ++i$ where i and j are local integer variables at offsets 2 and 3 respectively.

(c) $j = a[++i]$ where i and j are local integer variables at offsets 2 and 3 respectively, and a is a local variable at offset 1 that contains an array reference.

[CONTINUED ON NEXT PAGE]

[CONTINUED FROM PREVIOUS PAGE]

(d) `j = i++` where `i` and `j` are local integer variables at offsets 2 and 3 respectively.

(e) `j = a[i++]` where `i` and `j` are local integer variables at offsets 2 and 3 respectively, and `a` is a local variable at offset 1 that contains an array reference.

END OF EXAM

[THIS PAGE IS INTENTIONALLY BLANK]

[THIS PAGE IS INTENTIONALLY BLANK]

[THIS PAGE IS INTENTIONALLY BLANK]