CS 388: Compiler Design Test 1

Professor Ellen Walker

Fall 2007 12 Week

NAME: ________________________________________________ KEY

Answer all questions in the space provided. Use the backs of the pages if you need extra space. You may have a single-page one-side crib sheet for this exam.

Please make sure that you have all 8 pages before beginning the test.

If you are stuck on a question, move on and come back to it later.

<table>
<thead>
<tr>
<th>No.</th>
<th>Pts.</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>
1. (16 points) Which phase (or phases) of a compiler...

a. directly read(s) the input characters?

**scanner**

b. is/are considered the “back end”?

**code generator, target code optimizer**

c. access(es) the symbol table (for reading or writing)?

**scanner, semantic analyzer, code generator**

d. check(s) for type mismatches?

**semantic analyzer**

e. generate(s) a syntax tree?

**parser**

f. recognize(s) and eliminates comments?

**scanner**

g. perform(s) constant folding and loop unrolling?

**source code optimizer**

h. is/are independent of the underlying machine?

**scanner, parser, semantic analyzer, source code optimizer**
2. (10 points) Consider the language of all strings from the alphabet \{a,b,c\} containing the substring “abcabb”.

a. Write a regular expression that describes this language

\((\text{abc})^*\text{abcabb}(\text{albc})^*\)

b. Draw an NFA or DFA that describes this language
3. (8 points) Consider the following regular grammar: S is the start symbol.

   S -> aA | bS
   A -> aS | bB | ε
   B -> bS | ε

   a. Draw a DFA that is equivalent to this grammar.

   ![DFA Diagram]

   b. Describe the language accepted by this grammar in English or as a regular expression.

   **As a regular expression:** \((b+aa+abb)^* (a+ ab)\)

   **Strings must end in a or “ab”, a’s (except the last one) must come in pairs.**

4. (8 points) Create a DFA or NFA that accepts the same language as the regular expression: a (b+bc)*a

   ![DFA Diagram for a (b+bc)*a]
5. (10 points) Answer the questions for the following FA. State 1 is the initial state.

a. Is it a DFA or NFA? Justify your answer.

**NFA because it has epsilon transitions, and also because state q0 has two outgoing transitions on the same characters (b, to q0 and q3).**

b. Give a string of length 2 and a string of length 4 that are accepted by the FA.

**Length 2 accepted: aa, ab, ca (only)**

**Length 4 accepted: abba, abbb, acaa, acab, acca, baba, bbba, bbab, bbbca, caba, cabb, caca, ccaa, ccab, ccca**

c. Give a string of length 2 and a string of length 4 that are not accepted by the FA.

**Strings not mentioned in the answer to part b**
6. (12 points) Find the minimized DFA that is equivalent to the FA from problem 5. Show all work!

The following DFA is generated by considering all possible combinations of states that can be simultaneously entered for each transition. The DFA is already minimized. (The trap state is not shown. It would be entered by a from q0, c from q1, or c from q2).
8. (10 points) The C++ language has two kinds of comments. One goes from the pair of characters \// to the end of the line (\n character). The other is of the form /*...*/, where the ... can consist of any characters except the “*” followed by the “/”.

a. Draw a DFA that will recognize any C++ comment. You may use the notation “other” for an arrow that covers all characters except those with specific arrows of their own (as was in the textbook).

b. Write a regular expression that would match both forms of comment. You may use the “extended” form of regular expression used by FLEX if you desire. If you don’t remember the exact notation, make up your own (and explain it) for partial credit.

c. `\^* (\[^\*\] | [\^\*] )* \^*/ | [\^\n] *\n`
9. (10 points) Prove that the language consisting of decimal numbers that are multiples of 4 is a regular language.

**Proof by FA**

10. (10 points) Prove that the language consisting of strings of the form $a^n b b^* a^n$ is not a regular language. (The number of a’s before the b’s equals the number of a’s after the b’s).

**Proof by Pumping lemma.**

Assume the language is regular, then the pumping lemma must hold.

Given the pumping lemma constant $n$, we pick the string with $n$ a’s followed by a b, followed by $n$ more a’s.

We must divide this string into 3 parts (xyz) so that $|xy| \leq n$ and $|y| \geq 1$. Since $|xy| \leq n$, y must consist entirely of a’s before the b. Therefore, the pumped string xyyz will have more a’s before the b than after the b and it is not in the language. This is a contradiction (i.e. the pumping lemma does not hold for this language), so the language cannot be regular.