CSE 401 Midterm Exam – Autumn 2007 Snyder

Name: _____

CSE Email: _____

This is a “closed everything” test. Answer all questions.

Keep this page up until told to start

In this test the following alphabetic sets can be used.

Alpha ::= a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z
| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z

Num ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

1. [20] Specify the lexical structure of HTML. Here is an example of HTML

```
<html><head><title>Any sequence of letters numbers </title>
      </head>
      <body>
      <p width="100">Tags are reserved words, like body,
         enclosed in brackets; ending tags have a slash
         before the reserved word. Inside of tags are
         attribute keywords, like width, which are followed
         by an equal sign delimiter and a value in quotes.
         Values are either all numerical, all alphabetic or
         a mixture of each. Tags bracket text that is a
         mix of numbers and letters and for paragraph (p)
         tagged text, bracketed text. Note overall head-
         body structure.
      </p>
      </body>
</html>
```

Using the example as a guideline, define the LEXICAL structure of HTML by completing the following definition. (Note, the check that tags match is a *syntactic* part of the language.) HINT: The underlined words above refer to lexically important concepts that could correspond to a set of tokens. It is sufficient to handle the example, not all of HTML.

```
<program> ::= (<bracked_unit> | <white_space>)*
```

2. [5] Lexical analysis is the only pass that “sees” the actual text of a program, and therefore the only pass that can know which line an identifier is on. Line numbers are useful for error messages. Explain (briefly!) how a compiler can know the line number of each occurrence of an identifier in the program text. (Explaining how `jflex` does it would be a good answer.)

3. [15] Tables in HTML are specified with matching table tags containing a sequence of one or more rows, which are matching table-row tags, containing a sequence of one or more matching table-data tags, which can contain a mix of letters and numbers, or tables. White space is allowed. For example,

```
<table>
  <tr><td>stuff</td><td>goes </td><td>here   </td></tr>
  <tr><td>good </td><td>stuff</td><td>usually</td></tr>
</table>
```

Give an EBNF specification for HTML tables.

4. [7] Given the following grammar, form the *closure* for the production `S ::= {.L}` for an LR parser.

```
S’::= S $
S ::= beep | { L }
L ::= S | L ; S
```

```
S ::= {.L }
```

5. [5] The following grammar is suspected to be ambiguous. Show that it is, or argue that it can't be.

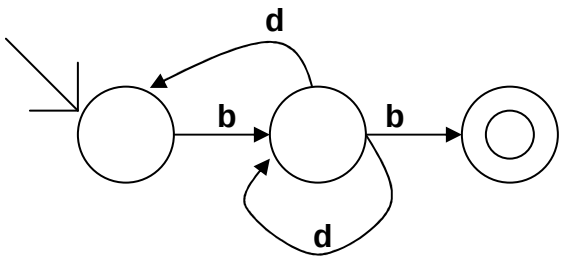
```
S ::= beep |  b + B |  B | { S }
B ::= bop + B  | bop  | { beep }
```

6. [10] Give (a) the concrete syntax tree and (b) the abstract syntax tree for:
`(a || b) && c` using the grammar and MiniJava-like nodes.

```
E ::= E || T | T
T ::= T &&  F | F
F ::= id | !id | ( E )
```

Derivation AST

7. [10] Given the following NFA, use the subset construction to find an equivalent DFA.



8. [8] Semantic analysis is a left-most traversal of the abstract syntax tree. Say, briefly, what the activity the semantic analysis pass performs as it moves around the tree:
a) At internal nodes on the “down sweep”

b) At leaf nodes

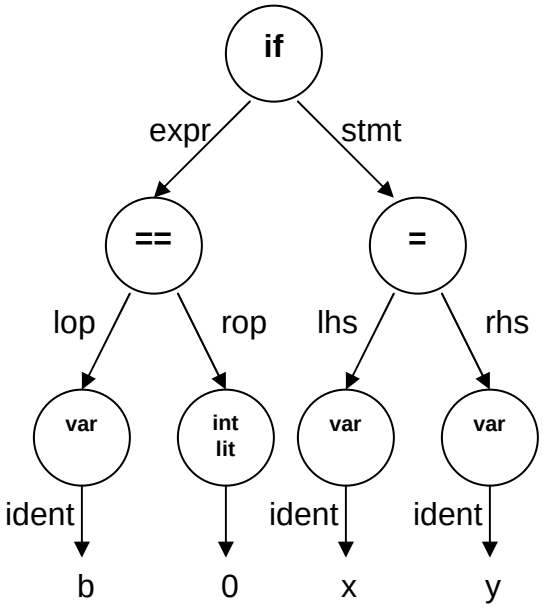
c) At the internal nodes on the “up sweep”

9. [5] Your friend is new to programming and is having trouble understanding the EBNF for MiniJava:

```
Stmt ::= Type ID ;
       | { {Stmt} }
       | if ( Expr ) Stmt else Stmt
       | while ( Expr ) Stmt
       | System.out.println ( Expr ) ;
       | ID = Expr ;
```

Answer your friend's questions about the language: "How do you know when to put in a semicolon after a statement?"

10. [20] Given the following AST, list **each** semantic check that would be performed during a semantic analysis pass. Suggestion: List them bottom up.



1.

2.