Compiler Design Midterm

NAME: ______________________________

Directions

Read carefully. Work individually. Write legibly. Check work. Complete in 1 hour.

Beforehand Visit the restroom if necessary. Close your laptop. Clear your desk. Silence your phone.

DO Use pencil, eraser, pen, or scratch paper to complete this exam.

DO NOT Distract others, talk, use electronic devices, notes, smoke signals, gestures, Morse code, . . .

Confused? **Let me come to you.** I will clarify questions. I won't answer: "Is this right or wrong?"

1 Definitions and Examples

1. Match these terms to their definition or exemplars.
   A. Nondeterminism B. Ambiguity C. First set D. Follow set E. Compiler F. Interpreter

   (a) Translates a source language into a target language

   (a) **E**

   (b) Executes a source language

   (b) **F**

   (c) Having more than one option about which state to transition to

   (c) **A**

   (d) More than one parse tree is possible

   (d) **B**

   (e) The set of terminals appearing first in any string while deriving a nonterminal

   (e) **C**

   (f) The set of terminals appearing first in any string after deriving a nonterminal

   (f) **D**

   (g) Examples: `gcc`, `javac`

   (g) **E**

   (h) Examples: `sh`, `bash`, `cmd`

   (h) **F**

2 Chomsky Hierarchy

2. Match these languages to their constraints on productions, or equivalent automaton or parsing strategy.[1]
A. Context-free B. Context-sensitive C. $LL(k)$ D. $LR(k)$ E. Recursively-enumerable F. Regular

(a) No constraints on productions.

(a) **E**

(b) Productions are constrained to: $A \to a$ or $A \to aB$, where a is a terminal and A, B are nonterminals.

(b) **F**

(c) Finite automata.

(c) **F**

(d) Recursive-descent parsing.

(d) **C**

(e) Shift-reduce parsing.

(e) **D**

(f) Unambiguous.

(f) **D**

(g) Nonterminals derive sequences of terminals and nonterminals.

(g) **A**

(h) Equivalent to Brainfuck (linearly-bounded Turing machines).

(h) **B**

(i) Pushdown automata (finite state machine with a stack).

(i) **A**

(j) Turing machines.

(j) **E**

(k) No common prefixes are allowed in productions.

(k) **C**

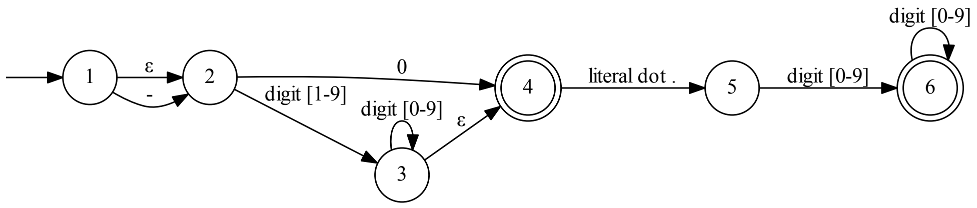
(l) No left recursion is allowed in productions.

(l) **C**

[1] If multiple languages match, choose the most inclusive (least constrained) language that matches.

3 Regular languages

Refer to this finite automaton to answer questions in this section.



3. Which of these strings does this finite automaton accept?
   A. `0.` B. `-.5` C. `007` D. `1.` E. `1`

3. **E**

4. Which regular expression matches the same language as the finite automaton?
   A. `[-]?(0|[1-9][0-9]*)\.[0-9]+`
   B. `[-]?0|[1-9][0-9]*\.[0-9]+`
   C. `[-]?(0|[1-9][0-9]*)(\.[0-9]+)?`
   D. `[-]?(0|[1-9][0-9]*)\.[0-9]+?`
   E. `[-]?0|[1-9][0-9]*\.[0-9]+?`
   F. None of the above

4. **C**

5. Let L be the language the finite automaton matches. What is the derivative of L, with respect to `3`?
   A. `[0-9]*\.[0-9]+`
   B. `[0-9]*(\.[0-9]+)?`
   C. `[0-9]*\.[0-9]+?`
   D. `[1-9][0-9]*\.[0-9]*`
   E. ∅

5. **B**

6. Let L be the language the finite automaton matches. What is the derivative of L, with respect to `a`?
   A. `-?(0|[1-9][0-9]*)\.[0-9]+`
   B. `-?0|[1-9][0-9]*\.[0-9]+`
   C. `[0-9]+`
   D. `\.[0-9]+`
   E. `0|[1-9][0-9]*\.[0-9]+`
   F. ∅

6. **F**

7. This finite automaton is: A. deterministic B. non-deterministic

7. **B**

4 Grammars and parsing

Refer to this grammar to answer questions in this section.

$$\begin{aligned}
\mathit{Start} &\rightarrow \mathit{Stmt} \\
\mathit{Stmt} &\rightarrow \mathit{id} = \mathit{Expr} \\
&\mid \texttt{if}\ \mathit{Expr}\ \texttt{then}\ \mathit{Stmts}\ \texttt{end} \\
&\mid \texttt{if}\ \mathit{Expr}\ \texttt{then}\ \mathit{Stmts}\ \texttt{else}\ \mathit{Stmts}\ \texttt{end} \\
&\mid \texttt{while}\ \mathit{Expr}\ \texttt{do}\ \mathit{Stmts}\ \texttt{end} \\
&\mid \texttt{begin}\ \mathit{Stmts}\ \texttt{end} \\
\mathit{Stmts} &\rightarrow \mathit{Stmts}\ ;\ \mathit{Stmt} \\
&\mid \mathit{Stmt} \\
\mathit{Expr} &\rightarrow \mathit{Expr} + T \\
&\mid T \\
T &\rightarrow \mathit{id} \\
&\mid \mathit{num} \\
\mathit{id} &\rightarrow [\texttt{a} - \texttt{zA} - \texttt{Z}]+ \\
\mathit{num} &\rightarrow [\texttt{0} - \texttt{9}]+
\end{aligned}$$

8. This grammar is:

A. Regular B. LL(k) C. Context-free D. Context-sensitive E. Context-free, Context-sensitive

8. **E**

9. What is $\mathit{First}(\mathit{Stmt})$?

A. *id*, *num* `if`, `while`, `begin`, `+`, `;`, `=`

B. *id*, `if`, `while`, `begin`, `+`, `;`, `=`

C. *id*, `if`, `while`, `begin`, `+`, `;`

D. *id*, `if`, `while`, `begin`, `+`

E. *id*, `if`, `while`, `begin`

9. **E**

10. What is $\mathit{Follow}(\mathit{Stmt})$?

A. `end`

B. `;`, `end`

C. `;`, `end`, `+`

D. `;`, `end`, `+`, *num*

E. `;`, `end`, `else`, end of string

10. **E**

11. Which sequence of shifts and reductions will eventually parse `x = 2`?

A. shift *id* `x`, shift `=`, shift *num* `2`, reduce $\mathit{num} \rightarrow T$, ...

B. shift *id* `x`, reduce $\mathit{id} \rightarrow T$, reduce $T \rightarrow \mathit{Expr}$, shift `=`, ...

C. shift *id* `x`, shift `=`, reduce $\mathit{id} \rightarrow T$, shift *num* `2`, ...

D. shift *id* `x`, shift `=`, shift *num* `2`, reduce $\mathit{num} \rightarrow \mathit{Expr}$, ...

11. **A**

5 Compiler phases

12. Which compiler phase is typically implemented as a list of regular expressions?

A. Scanner/Lexer/Tokenizer B. Parser C. Type checker D. Optimizer E. Code generator

12. **A**

13. Which compiler phase(s) is/are typically implemented as visitor?

A. Scanner/Lexer/Tokenizer

B. Parser

C. Type checker

D. Optimizer

E. Code generator

F. Type checker, optimizer, code generator

13. **F**

14. From source to machine code, what are the compiler phases, in order?

A. Scanning, Parsing, Optimization, Type checking, Code generation

B. Parsing, Scanning, Optimization, Type checking, Code generation

C. Scanning, Parsing, Type checking, Optimization, Code generation

D. Optimization, Scanning, Parsing, Type checking, Code generation

14. **C**

This exam has 33 answers total (including a blank for your name), so each answer is worth ≈ 3 points. Check your work. See you tomorrow!