Name: _____ Solutions _____

Rules:

- No potty breaks.
- Turn off cell phones/devices.
- Closed book, closed note, closed neighbor.

- <u>WEIRD!</u> Do not write on the backs of pages. There are extra blank pages at the end of the exam. If you need more pages, ask me for some.

Reminders:

- Verify that you have all pages.
- Don't forget to write your name.
- Read each question <u>carefully</u>.
- Don't forget to answer <u>every</u> question.

1. [2pts] T or F? Language *syntax* pertains to the form in which sentences in the language are expressed, whereas language *semantics* pertain to the meaning of sentences in the language.

    (a.) True

    b. False

2. [2pts] T or F? A language *interpreter* is a program that reads sentences in a language and "executes" them.

    (a.) True

    b. False

3. [5pts] Fill in the blanks in this sentence with the appropriate letters below.

    A __e__ takes a __b__ as input, and produces as output a __c__, which is in turn

    passed to a __a__ that processes it and produces a __d__ as output.

    a. Parser

    b. Stream of characters

    c. Stream of tokens

    d. Parse tree

    e. Scanner

4. [2pts] Fill in the blanks in this sentence with the appropriate letters below.

    In ANTLR, to generate a parser, you specify __d__, and to generate a scanner, you specify __a__.

    a. Regular expressions

    b. A non-deterministic finite-state automaton (NFA)

    c. Irregular expressions

    d. A context-free grammar (CFG)

    e. A deterministic finite-state automaton (DFA)

5. [3pts] Which of these ANTLR productions is a regex, and why? Circle one choice and then explain.

    a. `blah : preBlah+ ':' blah '.' ;`

    (b.) `UGH : ([a-z]* '88')+ ;`

Choice b is a regex because, unlike choice a, it does not recursively reference itself, and its name starts with a capital letter (an ANTLR convention).

6. [9pts] Write an ANTLR regex rule called "Title" that matches only a string that is in the form of a capitalized title (as in a book or movie titles). Such titles may be made up of one or more words (letters only), separated by single spaces (no leading or trailing spaces). Each word must begin with a capital letter and the rest of the letters must be lowercase. For example, these are legal titles: "Jaws", "No Country For Old Men". On the other hand, these titles would not be legal: "ET", "500 Days Of Summer", "in bruge", "Lethal   Weapon".

```
Title : Word ('\ ' Word)* ;
Word:  [A-Z]([a-z]*) ;
```

7. [9pts] Circle all the strings below that following regex would match.

`BUG : ('Wolf' | 'Garden' | 'Scary')* 'Green'? ('Spider' | 'Roach' | [A-Z]([a-z]*));`

    a. "" (the empty string)

    (b.) "WolfSpider"

    (c.) "GreenRoach"

    (d.) "ScaryWolfSpider"

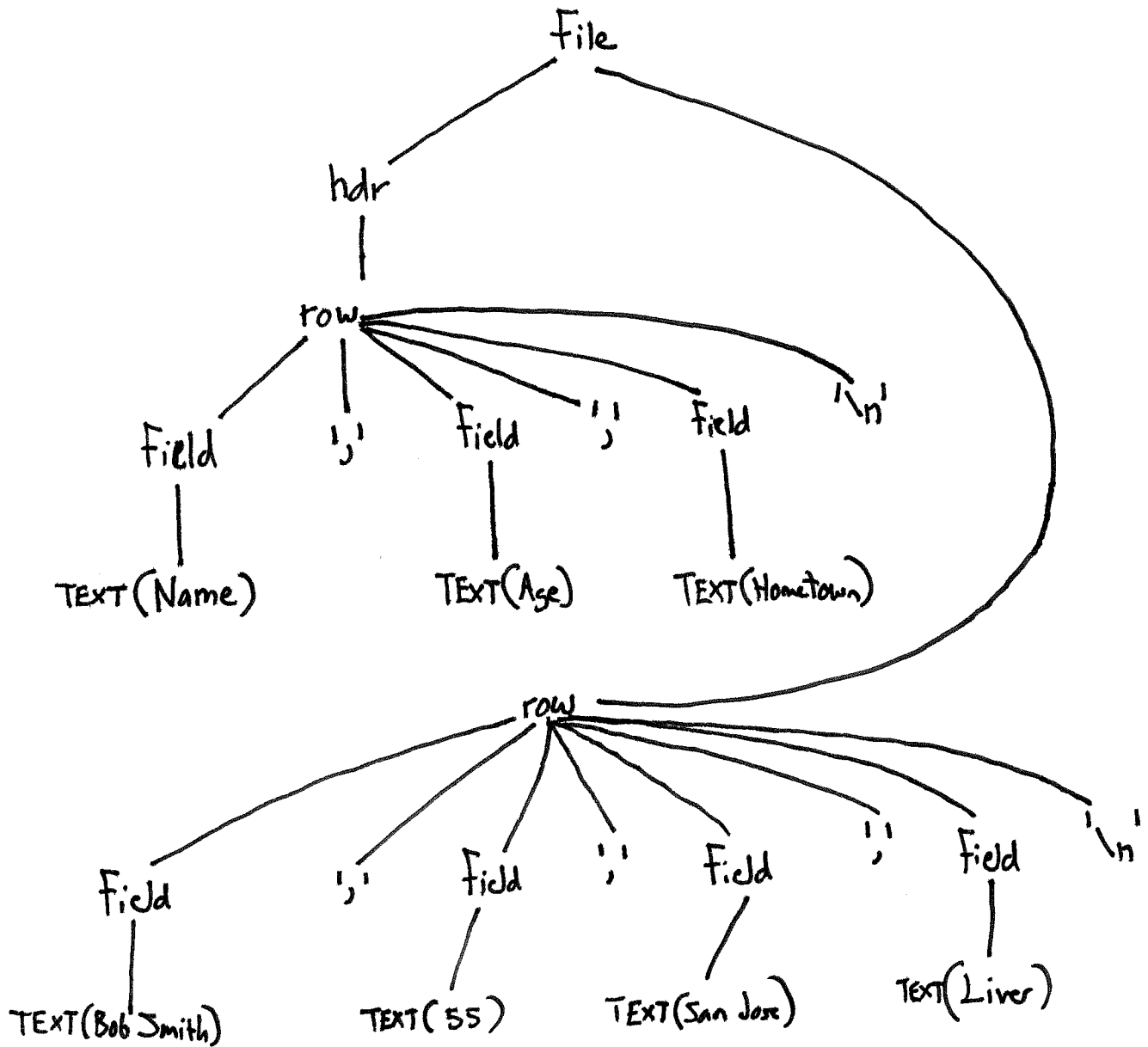   (e.) "WolfWolfWolfWolfWolfWolfRoach"

    (f.) "Zipper"

3

Consider this ANTLR grammar:

```
grammar CSV;

file : hdr row+ ;
hdr : row ;
row : field (',' field)* '\r'? '\n' ;
field : TEXT ;
TEXT : ~[,\n\r"]+ ;   // The ~ means "not"
```

8. [13pts] Using the CSV grammar, draw a parse tree for the following input string (which you'll note is a little weird). Assume that each line ends with a '\n' character.

```
Name, Age, Hometown
Bob Smith, 55, San Jose, Liver
```

9. [13pts] Assume that you are given (i.e., you need <u>not</u> define) the tokens NUMBER (for any kind of number), STRING (for any quoted string), and IDENTIFIER, and that whitespace characters are being skipped. Define an ANTLR rule (or rules) that will match function calls (similar to Java). A function call has a name (an identifier) followed by a parameter list (in parentheses, and comma separated). A parameter can be a string, a number, or another function call. Here are some examples that your rule(s) should match:

- foo()
- start(40)
- getX(bar())
- setY("hi", 30)
- run(stop(30), getSpeed(getConfig("speed-level")))

```
FuncCall :  IDENTIFIER  '('  argList  ')'  ;

argList  :  (arg  argListTail)?  ;

arg  :  STRING | NUMBER | FuncCall  ;

argListTail  :  (','  argList)?  ;
```
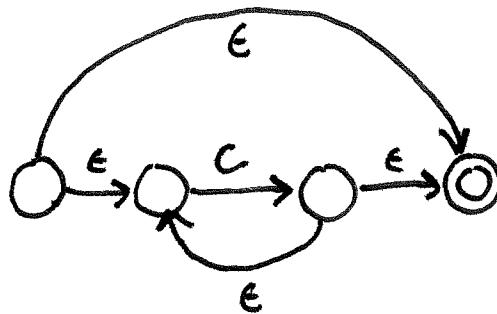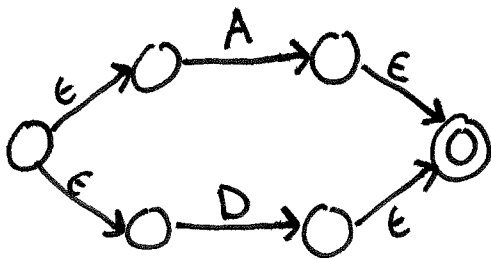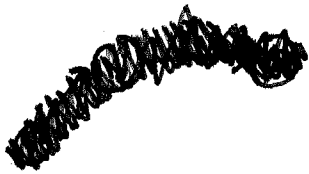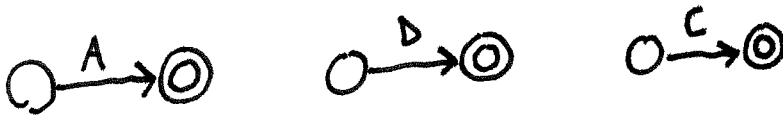
10. [12pts] Create a non-deterministic finite automaton for the following regex:

```
ACDC : ('A' | 'D') 'C'* ;
```

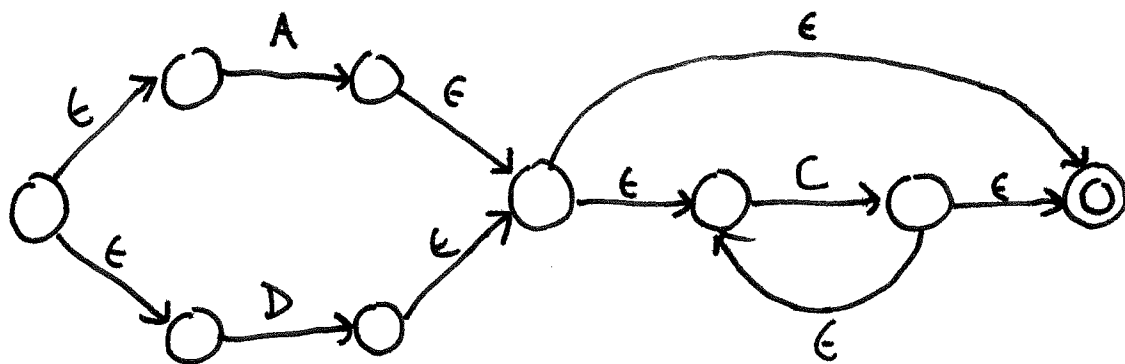As a hint, here are the NFA templates from class, but with the names elided.

← AB

← A|B

← A*
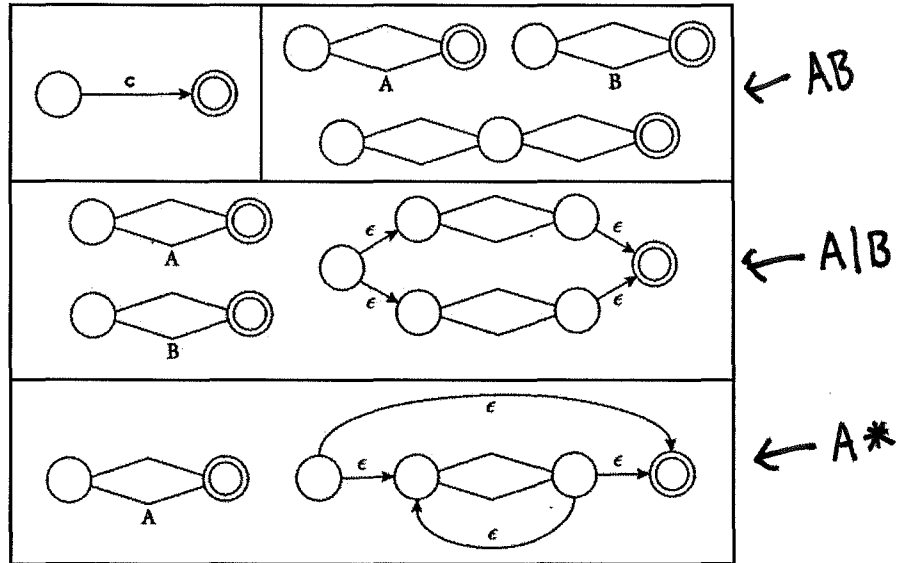
Also, there is additional space for you to draw on the next page.

(Final answer next page)

(extra space)

8