CS 340, Fall 2014 — Dec 11th/13th — Final Exam Name: Solution

Note: in all questions, the special symbol ϵ (epsilon) is used to indicate the empty string.

Question 1. [5 points] Consider the following regular expression;

(a|ab|bac)*

For each of the strings below, circle it if it is in the language generated by the regular expression, and cross it out if it is not in the language generated by the regular expression.

- ϵ
- a
- ~~b~~
- aba
- ~~bab~~
- bac
- bacaa
- ~~babac~~
- abbac
- ~~abbbac~~

Question 2. [5 points] Specify a regular expression that generates the language over the alphabet {**a**, **b**, **c**} of all strings not containing the substring **ac**.

Examples of strings in the language:

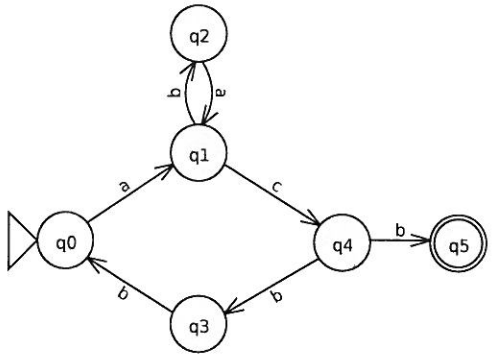
- ϵ
- **aaab**
- **caaaa**
- **abc**
- **bbc**

Examples of strings not in the language:

- **ac**
- **aac**
- **cac**
- **bacb**
- **bbac**

(b|c|a*b)* (a|ε)

Question 3. [5 points] Consider the following nondeterministic finite automaton (NFA):

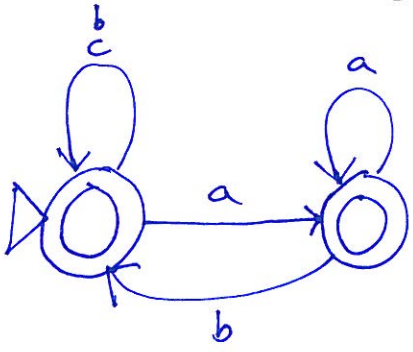


For each of the strings below, circle it if it is in the language accepted by the NFA, and cross it out if it isn't.

- ~~ϵ~~
- ~~a~~
- ~~ac~~
- acb
- ~~acbbac~~
- acbbacb
- abacb
- ~~abacbb~~
- ~~aabcb~~
- ababacb

Question 4. [10 points] Specify a *deterministic* finite automaton (DFA) that generates the language over the alphabet {**a**, **b**, **c**} of all strings not containing the substring **ac**. (See Question 2 for lists of example strings in the language and not in the language.)

In your DFA, be sure to indicate a start state and one or more final states, and make sure each transition specifies a direction and a single input symbol consumed.



Question 5. [5 points] Consider the following context-free grammar (CFG), where **P** is the start symbol:

$\mathbf{P} \rightarrow \mathbf{pP}$
$\mathbf{P} \rightarrow \mathbf{qP}$
$\mathbf{P} \rightarrow \mathbf{r}$

For each of the strings on the right, circle the string if is in the language generated by the CFG, and cross it out if it isn't.

- p
- q
- r
- ppr
- qqr
- prr
- pqrr
- pqqp
- pqqpr
- qq

Question 6. [10 points] Specify a context-free grammar (CFG) which generates the language of all *ALists*. An *AList* has the following properties:

- It is a sequence of terminal symbols chosen from ( ) a ,
- It begins with (and ends with)
- It contains (between the parentheses) a comma-separated list of 0 or more *AListMembers*

An *AListMember* is either the terminal symbol **a** or an *AList*.

Examples of strings in the language

()
(a)
(a, a)
((a, (a, (a), a)), a)

Examples of strings not in the language:

ϵ
a
a, (a)
(a
(a, a))

Be sure to indicate which nonterminal symbol in your grammar is the start symbol.

L → (M)
M → N
M → ε
N → I
N → I, N
I → a
I → L

L is the start symbol

Question 7. [10 points] Consider the following context-free grammar with start symbol A, nonterminal symbols A E F V N and terminal symbols ( ) λ . a b 1 2

A → E | E A
E → F | V | N
F → (λ V . E)
V → a | b
N → 1 | 2

Assume that you are writing a recursive descent parser for this grammar. Assume you have a lexer supporting `peek()`, `next()`, and `expect()` operations. (You can assume that the lexer operations are stateful, i.e., you don't have to assume that you are using a functional language.) Use pseudo-code to show how the following parse functions would be implemented. You don't have to build the parse tree, but do show all lexer operations and calls to parse functions. *Hint*: for `parseE`, think about how to use the lexer to decide whether tokens matching an occurrence of the F, V, or N nonterminal is about to begin.

(a) `parseE() {`

```
t = peek();
if (t == '(') { parseF(); }
else if (t == 'a' || t == 'b') { parseV(); }
else if (t == '1' || t == '2') { parseN(); }
else { error() }
```

}

(b) `parseA() {`

```
parseE()
if (peek() != null) {
    parseA()
}
```

}