CS 340, Fall 2014 — Sept 25th — Midterm 1 Name: Solution

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

Question 1. [10 points] Consider the regular expression **a(b(a|ac))***. For the following strings, circle the ones that are in the language generated by this regular expression, and cross out the ones that are *not* in the language generated by this regular expression.

- **a**
- ~~**b**~~
- ~~**ba**~~
- ~~**ab**~~
- ~~**abc**~~
- **abac**
- **aba**
- ~~**bac**~~
- **ababa**
- **abacba**

Question 2. [10 points] Specify a regular expression that generates the language of all strings over the alphabet {**a**, **b**, **c**} which have an odd number of symbols and which both start and end with **a**.

Examples of strings in the language:

- **a**
- **aba**
- **aca**
- **abaca**
- **aabca**

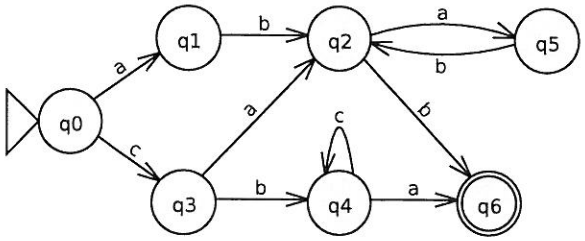
Examples of strings not in the language:

- ϵ
- **aa**
- **baa**
- **abaa**
- **abbab**

a | a ((a|b|c) ((a|b|c)(a|b|c))*) a

guaranteed to produce an odd-length string

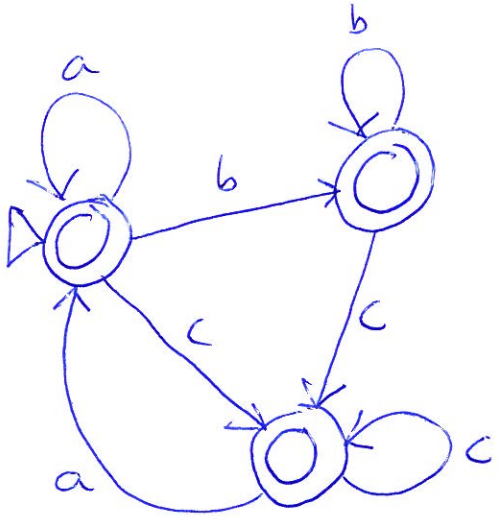
Question 3. [10 points] Consider the following finite automaton:



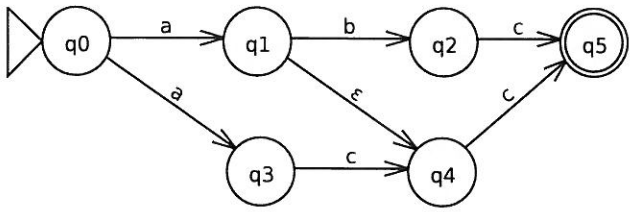
For each string below, circle the string if it is the language recognized by the finite automaton, and cross out the string if it is *not* in the language recognized by the finite automaton.

- ~~aba~~
- abb
- ~~abab~~
- ~~ababa~~
- ababb
- cba
- ~~cbcc~~
- cbca
- ~~caab~~
- caabb

Question 4. [10 points] Create a *deterministic* finite automaton that recognizes the language over the alphabet {**a**, **b**, **c**} of all strings that do not contain the substrings **ba** or **cb**. Be sure to indicate which state is the start state, and which state or states are final states. Also, make sure that each transition is labeled with a single input symbol.

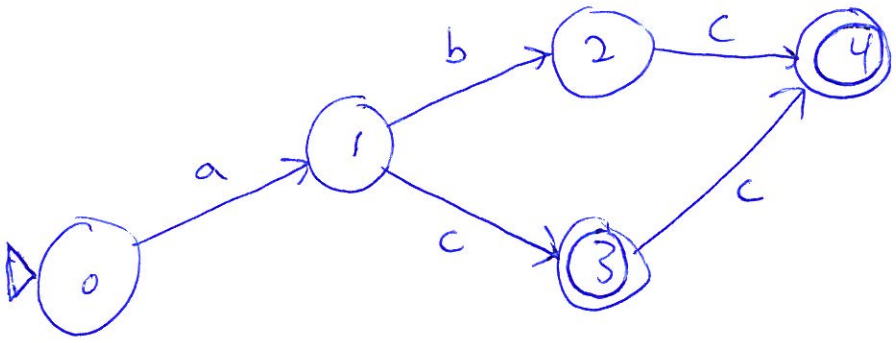


Question 5. [10 points] Convert the following nondeterministic finite automaton (NFA) to a *deterministic* finite automaton (DFA). As you do the conversion, create a table showing how sets of NFA states map to corresponding DFA states.

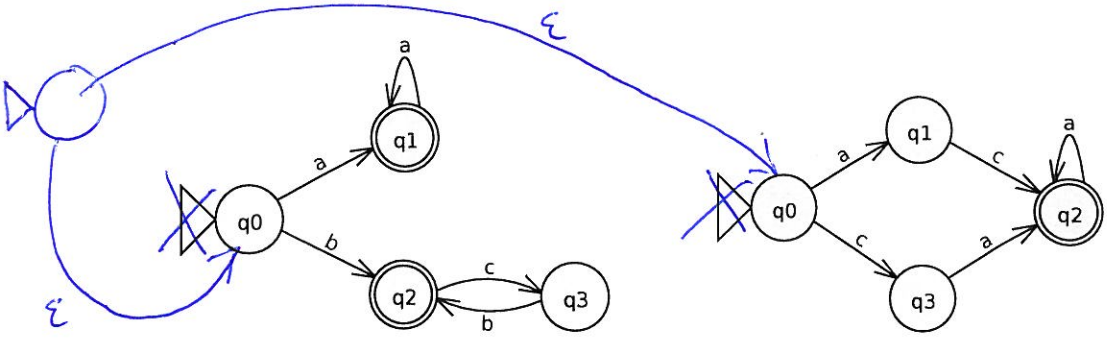


Note that ϵ denotes a transition that does not consume an input symbol.

NFA states	DFA state
{0} ✓	0
{1,3,4} ✓	1
{2} ✓	2
{4,5} ✓	3
{5} ✓	4



Question 6. [10 points] Consider the following deterministic finite automata (DFAs):



Show how to construct a finite automaton that recognizes the union of the languages accepted by these two automata. Hint: your automaton can be nondeterministic.

Create new start state,
create ε-transitions to
original start states

Question 7. [10 points] Consider the following context-free grammar (CFG):

start symbol → P → abQ
Q → bQa
Q → c

For the following strings, circle the ones that are in the language generated by this CFG, and cross out the ones that are *not* in the language generated by this CFG.

- ab
- abc
- abba
- abaca
- abbca
- abbbbcaa
- abbbbcaaa
- c
- bca
- abbbaa

Question 8. [10 points] Consider the following context-free grammar, where the terminal symbols are **a b { } : ,** and the start symbol is **V**:

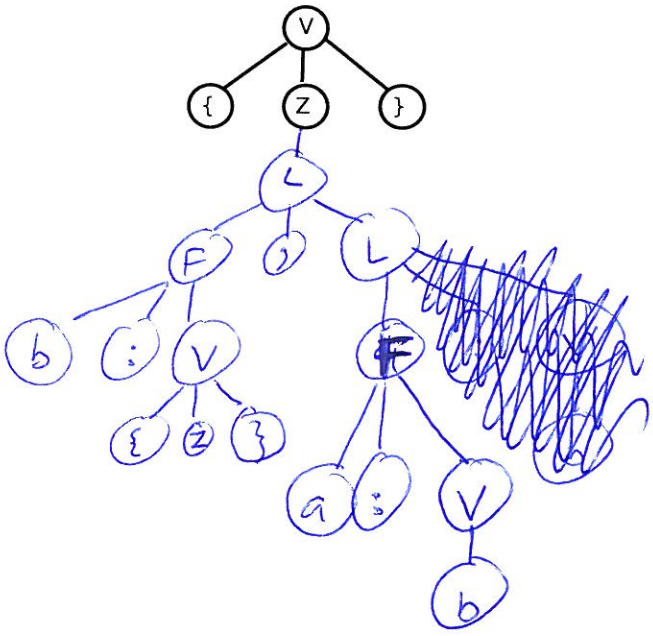
V → **a**	**L** → **F , L**
V → **b**	**L** → **F**
V → **{ Z }**	**F** → **a : V**
Z → L	**F** → **b : V**
Z → ϵ	

(a) Show a derivation for the string **{ b : { } , a : b }**. (The first step is given.) Continue on the right if necessary.

String	Production
V	**V** → **{ Z }**
{ Z }	Z → L
{ L }	L → F, L
{ F, L }	F → b : V
{ b : V, L }	V → { Z }
{ b : { Z }, L }	Z → ε
{ b : { }, L }	L → F
{ b : { }, F }	F → a : V

String	Production
{ b : { }, a : V }	V → b
{ b : { }, a : b }	

(b) Complete a parse tree for the derivation in part (a).



Question 9. [10 points] Consider the following productions in a context free grammar:

R → **a** S
R → **b** T

Assume that **a** and **b** are terminal symbols, and R, S, and T are nonterminal symbols. Show the pseudo-code for a recursive descent parse function for the nonterminal R (*i.e.*, a `parseR` function).

Important: Show how to use the lexical analyzer to choose between the two possible productions. Also, show how to raise an error if neither of the productions can be applied.

```
parseR() {
    t = peek()
    if (t == 'a') {
        expect('a')
        parseS()
    } else if (t == 'b') {
        expect('b')
        parseT()
    } else { error }
}
```

Question 10. [10 points] Create a Turing Machine that takes an input string of **a** and **b** symbols and changes the last symbol in the input string to whatever the first symbol was. For example, if the input string is **abbab**, the resulting string (when the Turing Machine halts) should be **abbaa**. As a special case, if the input string is empty, the Turing Machine should halt immediately.

Be sure to indicate which state is the start start and which state or states are final (halt) states. Also, make sure each transition is labeled with an input symbol, output symbol, and direction. (Use the back of the page if necessary.)