CS 340, Fall 2016 — Dec 15th/17th — Final Name: _____

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|ba)*

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

- ϵ
- a
- aa
- aab
- bba
- abab
- ababa
- abba
- aaaab
- aabab

Question 2. [5 points] Write a regular expression that generates the language over the alphabet {**a**, **b**} of all strings having an even number of **b**s.

Examples of strings in the language:

- ϵ
- a
- aabab
- bbabab
- abba

Examples of strings not in the language:

- b
- ab
- abbba
- bbab
- babab

Question 3. [5 points] Write a regular expression that generates the language over the alphabet {**a**, **b**, **c**} of all strings not containing the substring **cc**.

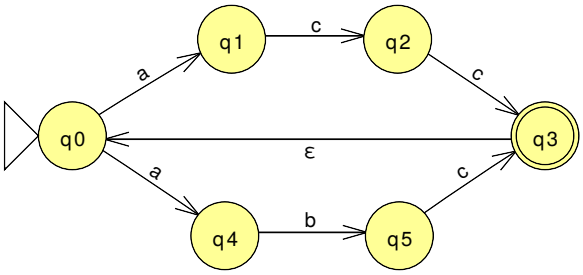
Examples of strings in the language:

- ϵ
- c
- ab
- abcac
- cabc

Examples of strings not in the language:

- cc
- ccc
- ccbc
- abccb
- baccccbc

Question 4. [5 points] Consider the following nondeterministic finite automaton (NFA):



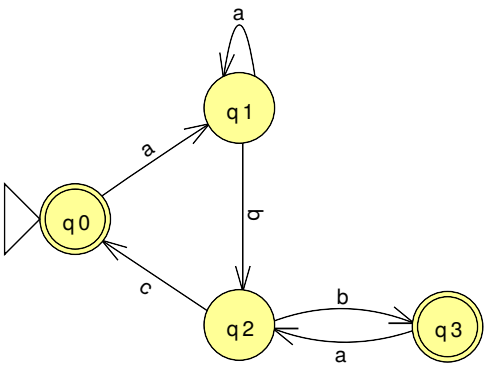
For each of the following strings, circle the string if it is in the language recognized by the NFA, or cross it out if it is not in the language.

- ϵ
- a
- acc
- abb
- abc
- accacc
- accabc
- abcacc
- accab
- abbabbabc

Question 5. [5 points] Create a *deterministic* finite automaton (DFA) that recognizes the language described in Question 2. Make sure that the start state and final state(s) are labeled, that each transition is labeled with *exactly one* input symbol, and that the direction is indicated for each transition.

Question 6. [5 points] Create a *deterministic* finite automaton (DFA) that recognizes the language described in Question 3. Make sure that the start state and final state(s) are labeled, that each transition is labeled with *exactly one* input symbol, and that the direction is indicated for each transition.

Question 7. [5 points] Consider the following deterministic finite automaton (DFA):



Create a DFA that recognizes the complement of language recognized by the DFA shown above. In other words, given any string over the alphabet {**a**, **b**, **c**}, if the above DFA accepts the string, yours should reject it, and if the above DFA rejects the string, yours should accept it. Make sure that the start state and final state(s) are labeled, that each transition is labeled with *exactly one* input symbol, and that the direction is indicated for each transition.

Question 8. [5 points] Write a context-free grammar for the language over the alphabet **a b c ,** of all comma-separated lists of 0 or more **a**s, **b**s, or **c**s.

Examples of strings in the language:

- ϵ
- a
- a , a , b
- c , b , c
- c , c , a , c , b

Examples of strings not in the language:

- a a
- , a
- c ,
- a , , b
- b c , a

Make sure that you indicate what the start symbol of your grammar is, and that each production has a single nonterminal symbol on the left hand side.

Question 9. [5 points] Consider the following context-free grammar (CFG), with nonterminal symbol S and terminal symbols a , :

S → S , a
S → a

Rewrite this grammar so that it accepts the same language, but is suitable for recursive descent parsing.

Question 10. [10 points] Consider the following context-free grammar (CFG) with nonterminal symbols S A I, terminal symbols ; ( ) = a b f, and start symbol S:

S → A	A → I
S → S ; A	I → a
A → I (A)	I → b
A → I = A	I → f

(a) Show a derivation for the string a = f (a) ; b. Continue on the right-hand side if necessary:

String	Production	String	Production
S			

(b) Draw the parse tree for the derivation you found in part (a). Make sure that each node represents a single nonterminal or terminal symbol.

Programming Questions

First things first: the following web page specifies which resources you may use during the exam, and links to the permitted resources:

`http://ycpcs.github.io/cs340-fall2016/assign/final.html`

Start by downloading the exam zipfile using the command

`wget` *zipfileURL*

where *zipfileURL* is the URL of the exam zipfile.

Import the zipfile as an Eclipse project. The project will be called `cs340-final`. You will be editing the file `src/cs340_final/core.clj`.

There are four functions to complete: **swap-weird**, **conjn**, **replicate-members**, and **count-delimiters**. Each has a detailed comment explaining the requirements, expected behavior, and a point value.

Make sure you meet all of the requirements of each function: e.g., if the function is required to be tail recursive, make sure your function is tail recursive.

Some general hints:

- `(empty? s)` tests whether sequence `s` is empty
- `(first s)` gets the first element in sequence `s`
- `(second s)` gets the second element in sequence `s`
- `(rest s)` gets a sequence with all but the first element of sequence `s`
- `[]` is an empty vector
- `(conj v elt)` creates a vector which results from appending `elt` onto the vector `v`
- `(conj v elt1 elt2)` returns a vector which results from appending `elt1` and `elt2` onto the vector `v`
- `(concat s1 s2)` returns a sequence containing all of the elements of sequence `s1` followed by all of the elements of sequence `s2`

You can run the command `lein test` in a terminal window to run unit tests. Note that as you are working on a function, testing it interactively in a REPL (using the example inputs described in the function comment) will probably be the most useful form of testing.

When you are done, run the command `make submit` in a terminal. Type your Marmoset username and password when prompted.

Most importantly:

Have fun!