CS46 Lab 3

In typical labs this semester, you'll be working on a number of problems in groups of 3-4 students each. Lab problems are an opportunity for discussion and trying many different solutions. They are **not counted towards your grade**, and **you do not have to submit your solutions.** I encourage you to try to solve these problems on paper *first*, then trying with the Automata Tutor. Once you are ready to test your solutions, the Automata Tutor site will give you troubleshooting feedback.

Note that Automata Tutor uses slightly different notation for regular expressions.

1. **LaTeX** You'll be using LaTeXfor most of the homework assignments this semseter. For some of them you won't need much more than the math notation we've been seeing in class. However, some assignments will require you to construct a picture of a DFA.

   The purpose of this problem is to get more used to LaTeX and to learn how to build nice picturese of DFAs, NFAs, etc.

   (a) If you haven't done so recently, do a `git pull` on the lecture-notes repo. In that repo there is a `lab 3` directory containing a file called `LearningLaTeX.tex`. Open the file and scan through the comments to see some common LaTeX features.

   (b) **Compiling LaTeX.** The grader will always compile your submissions using a command called `pdflatex`. This program takes a .tex file, compiles the tex, and produces a pdf. Go ahead and use this command on `LearningLaTeX.tex`:

   ```
   pdflatex LearningLaTeX.tex
   ```

   You should get a single compilation error—that's ok. Just press return, and you should get a pdf you can read.

   (c) **Compiling Graphs.** In future homeworks this semester, you'll need to draw out DFAs or NFAs. You're welcome to do this on pen+paper and include the image in your submission. However, you can also use GraphViz to create nice looking graphs.

   Open `LearningLatex.tex` and scroll down to the section on creating graphs. Follow the instructions there and recompile `LearningLaTeX.tex`. You should now see a DFA we used last week.

   (d) Pick any DFA or NFA we've seen in lab 2 or lab 3, and try to modify dfa1 to create that DFA. Repeat the compilation process to ensure you can create your own automata pictures.

2. Construct an NFA for the language $\{w \mid w \text{ begins with } a \text{ and ends with } b\}$.

3. Construct an NFA that recognizes the language

$$\{w \mid w \text{ ends with } bb\}$$

   . Try to use the fewest number of states and transitions possible!

4. Construct an NFA for the language $L = L((aa \cup ab)^*)$ over the alphabet $\Sigma = \{a, b\}$.

5. Construct an NFA for each of the languages given by regular expressions:

   (i) `a(abb)`*`∪b`

   (ii) `a`$^+$`∪(ab)`*

   (iii) `(a∪b`$^+$`)a`$^+$`b`$^+$

6. Construct a regular expression for the language Σ^* over $\Sigma = \{a, b\}$.

7. Construct a regular expression for the language $\{w \mid w \text{ contains the substring } ab\}$ over $\Sigma = \{a, b\}$.

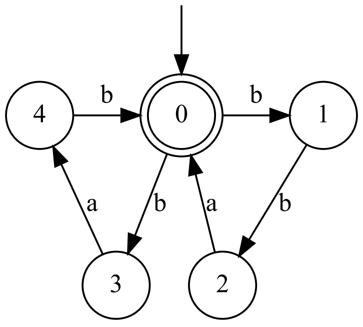
8. Construct a regular expression for the language $\{aa, abba\}$ over $\Sigma = \{a, b\}$.

9. Construct a regular expression for the language $\{w \mid w \text{ contains exactly two } a\text{s}\}$ over $\Sigma = \{a, b\}$.

10. Let $L = \{w \mid \text{every appearance of } c \text{ in } w \text{ is in a contiguous substring with at least two other } c\text{s}\}$ over $\Sigma = \{a, b, c\}$. So for example, L contains *baaccca*, *aaab*, ε, and *cccca*. L does *not* contain *abcccbaca*, *bccb*, *c*, or *cabaaaabcc*.

    - Construct a DFA that recognizes L.
    - Construct an NFA that recognizes L.
    - Construct a regular expression that recognizes L.

11. Construct a DFA equivalent to the following NFA:



    Write a regular expression for the language accepted by this machine.

12. **C++ comments**

    In C++, it is possible to write a comment as a string beginning with '`/*`' and ending with '`*/`'. In between, it can contain any characters, including '`/`' and '`*`', as long as it does *not* contain the substring '`*/`' before the end of the string. **For the sake of clarity in this problem, we will substitute '#' for the '*' character, so that it cannot be confused with the Kleene star operator.**

    For example, '`/#ab/ab/c/b####ac#/`' is a valid comment, but '`/#cab/baaaa###/a//bb#/`' is not a valid comment.

Construct a regular expression for the language L of this style of well-formed C++ comments over the alphabet $\Sigma = \{a, b, c, \#, /\}$.

$$L = \{w \mid w = \texttt{/\#}x\texttt{\#/},\ \text{where } x \in \Sigma^* \text{ does not contain the substring `}\texttt{\#/}\text{'}\}$$

You should explain your reasoning/show your work. A formal proof is not required, but you should be able to justify (at a high level) why your regular expression is correct.