CSci 3501 Lab 9
30 points (See canvas for due date)
Work in pairs

- All lab submissions should be done by canvas. Please be sure to include your group members.
- When working on the lab, please comment your work so that it is clear what the contributions from each person are.
- At the end of the lab each group should submit the results of their in-class work. Please indicate if this is your final submission. Don't forget to answer all the questions below.
- If your submission at the end of the lab time was not final, please submit a final copy before the due time.

Lab overview and goals

The goal of the lab is to get practice with converting DFAs to regular expressions, the pumping lemma, and simple context-free grammars.

Convert a DFA to a regular expression (8 points)

JFLAP guides you through the process of converting a DFA to a regular expression via a generalized NFA (GNFA), as described in the tutorial *Converting a FA to a Regular Expression*. JFLAP uses a slightly different version of a GNFA: it allows self-loops in the starting and the final state. The empty set transitions are added just like in the book, and the number of states is reduced by the procedure described in the book. The resulting regular expression then is combined as R1*R2R3*, where R1 is the self-loop expression in the start state, R2 is the expression on the transition from the start state to the final state, and R3 is the loop in the final state.

As you are transforming your DFA to a 2-state GNFA, write down (in a plain-text file) all transition changes that result in non-empty-set expressions.

Below are the DFAs to convert:

- The language of all strings with 00 pattern
- The language of all strings that either start with 0 and don't have any more 0s, or start with 1 and don't have any more 1s.

Please export and submit your resulting expression.

Play the "Pumping Lemma Game" (6 points)

The pumping lemma in JFLAP is implemented as a two-player "game" when one player is trying to prove that a language is regular by representing strings as required by the pumping lemma, and the other player is trying to disprove it, as described in the tutorial *Regular Pumping Lemmas*.

Go to "Regular Pumping Lemma" in the JFLAP start menu (careful: you don't want Context-Free Pumping Lemma). There is a list of languages, some are regular, some aren't. The alphabet is a,b. The pumping length is denoted as m. JFLAP allows you to save the file with the log of all your attempts. Please submit these files for the two cases below and additionally write down your conclusions in a plain-text file or an e-mail message.

Your conclusions should include:

1. Whether the language satisfies the pumping lemma. This is a bit tricky since you are testing a pumping length, but your conclusions should be general. For a language that satisfies the pumping lemma show its pumping length and explain how you would always be able to break down a string in a way that satisfies the pumping lemma. If the language does not satisfy the pumping lemma, show how to construct a string that breaks the pumping lemma. Show it for at least two values of candidates for the pumping length.
2. Whether it is regular (note that some languages that satisfy the pumping lemma may still be non-regular). Justify your answer, either by explaining why the language is regular, or by giving your intuition for why it is not regular. You don't need to do a proof.

The languages to try:

- The 6th example (the language $a^n b^j a^k$, where $n > 5$, $j > 3$, and $k \leq j$). Choose the "computer goes first" option so that you are trying to prove that the language is non-regular.
- The 8th example (the language $a^n\ b^k$, n is odd or k is even). Choose the "you go first" option so that you are trying to prove that the language satisfies the pumping lemma.

Context-free grammars (16 points)
Please refer to the corresponding sections of the JFLAP tutorial, namely Entering grammars (just pressing "enter" in RHS enters an empty string), Brute Force Parser for constructing parse trees, and Constructing a push-down automaton.

Your tasks are as follows:

- A context free grammar for the language of strings $a^n\ b^m$, where $n >= m$.
- A context free grammar for the language of strings a^k followed by any number of b followed by c^k.
- A context-free grammar for odd-length strings of alternating zeros and ones. It can start with either zero or one.
- A context-free grammar of strings of a,b that has more occurrences of "a" than occurrences of "b". The order of letters is arbitrary. Test your automaton on strings baaba and aaab and export the corresponding parse trees as jpg files.

What to submit

1. Submit your JFLAP files as attachments, CC your group. Make sure to submit your automata files (as .jff) and your input data (as .txt). Make sure to follow the naming requirements! Make it clear which data refers to which automaton.