CSE 322: Formal Models in Computer Science **Extra Credit**
May 28, 2008 Due: **Monday**, June 9, 2008, Midnight

This extra credit involves building automata and Turing machines using JFLAP, a Java-based simulator. You can download the program from the following web page: `http://www.jflap.org/`. Documentation for the program can be found at the same page.

For this extra credit you will create DFAs, NFAs, CFGs, and Turing Machines using JFLAP. Once you have constructed your machine or grammar, you can simulate it on input. Be sure to test your designs well before submitting them! To submit your work, save each machine or grammar that you design (using the Save command in the File menu) and put all of your files into a ZIP archive. Also include a text file in your ZIP archive containing a documentation for each of the problems. Your documentation for each problem should be a brief explanation for how your design works. For each problem, save your solution as a file of the name "<your-last-name><problem number>.jff". For example, for me, if I was solving problem 2, I would save this as "bacon2.jff". Save the documentations as "<your-last-name>doc.txt" Note that JFLAP uses λ where we would use ε.

Please submit your ZIP archive online to before the due date, Monday, June 9 at midnight: `https://catalysttools.washington.edu/collectit/dropbox/dabacon/2525`

Problems

1. Use JFLAP to construct a six-state DFA which accepts a string if it has a number of a's that is a multiple of 2 or a multiple of 3 (or both). Work over the alphabet $\Sigma = \{a\}$.

2. Use JFLAP to construct a NFA which accepts the language of the regular expression $(01 \cup 001 \cup 010)^*(0 \cup 1)^*$. Work over the alphabet $\Sigma = \{0, 1\}$.

3. Use JFLAP to construct a PDA which accepts the language $L = \{w \in \Sigma^* |$ the number of a's is greater than the number of b's in $w\}$. Use the alphabet $\Sigma = \{a, b\}$.

4. Use JFLAP to construct a CFG for the language $L = \{x\#y | x, y \in \{0, 1\}^*$ and $x \neq y\}$.

5. Use JFLAP to design a Turing machine which adds two numbers in unary. That is if the Turing machine tape initially contains "1111+11111=", then the machine should halt with a tape containing "1111+11111=111111111", representing $4 + 5 = 9$.