CS 540 Programming Fall 2014
Assignment 1: Using Lex

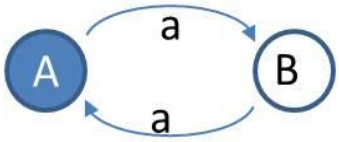
Due: Sun, Sept 21 at end of day (midnight)

As discussed in class, any DFA (or NFA) can be described using a context free grammar where there are only two different types of rules:

```
Nonterminal --> Terminal Nonterminal
Nonterminal -->
```

For example, the CFG

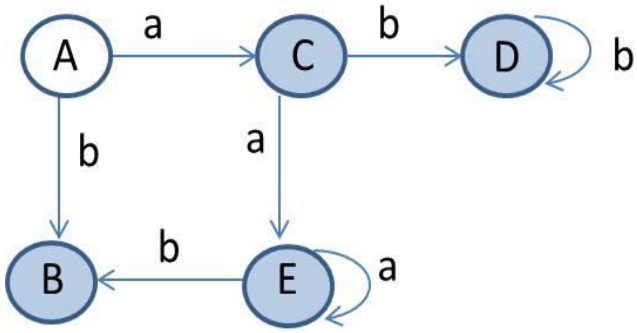
```
A --> a B
B --> a A
A -->
```



describes a DFA that recognizes an even number of 'a' characters.
In the diagram on the right, A is the start state and the only final state.

As a second example, the CFG

```
A --> a C
A --> b B
C --> a E
C --> b D
C -->
D --> b D
D -->
E --> a E
E -->
E --> b B
B -->
```



describes a DFA for (a*b | ab*) where A is the start state and Final = {B,C,D,E}.

While CFLs are a superset of regular languages, this restricted CFL can be recognized using regular languages. The DFA given at the end of this handout (states 1-5) will recognize lists of these two rule types (assuming that they are terminated by an end of line -- EOL -- marker).

For this assignment, we are going to process DFAs that have been specified using the above language (the described subset of CFGs). You are going to use Lex to read in and save DFAs specified using this format. For the purposes of this assignment, all non-terminals will be a string of 1 or more alpha-numerics, starting with a capital (uppercase) letter. All terminals are a single lowercase character. After reading in the DFA, you are going to read in 1 or more strings of input and see if each string is in the language. This last part is recognized using states 6 and 7 of the DFA below.

Going back to the first example, the input file could look like

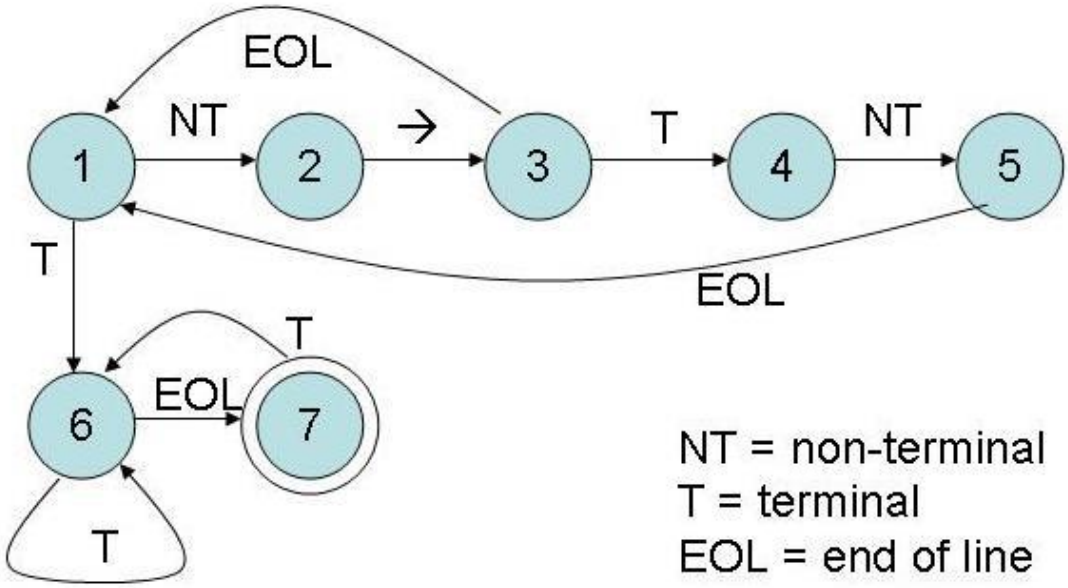
```
A --> a B
B --> a A
A -->
aaaaa
aaaa
aaaaaaaaaa
```

Your tool would read in this input file and output:

```
String rejected
String accepted
String accepted
```

since the first string has an odd number of a's and the next two have even numbers of a's. You can assume that the first production in the input gives you the start symbol.

Note: The input DFAs are guaranteed to be correct syntactically and are also guaranteed to be DFAs (as opposed to NFAs).



Creating this assignment

You will need to do this assignment in several steps. BE SURE the previous steps work before going on to the next step.

1. Using a lex tool, write REs for the tokens of the above DFA.
2. Once this works correctly, you will want to use this starting lexer to build something that processes the input as a whole. You can use Lex states to decide what the appropriate actions are for a given token or build your own state machine in the actions. Don't worry initially about saving the DFA itself – get the recognition part working first.

3. Now that you can process the input, add to your actions code that saves information about the DFA as you process it. The simplest way to keep track of the transitions is a 2-D array. You will also need to record the start state (first state defined) and final states (states with transitions to epsilon.
4. Write actions to execute the saved DFA for the given inputs. Be sure to continue to 'process' a given input line even once you have already determined the string is not in the language.

Be sure to test your program on different inputs. TRY to break your tool; this is preferable to having me break it ☺

Submitting this assignment

Electronic submissions on blackboard. Be sure to include information in the comments

- on what language you used (C, C++ or Java),
- build instructions and
- the system (osf1, zeus) where you want me to build/test your assignment.

For this and all other assignments, but sure you give me ALL files needed to build your executable. The TA will not test your program on any executables you include.