COMP 370, Spring 2018
Program #2, NFA to DFA Conversion
Date Assigned: 3/2/18
Date Due: The night of Friday, 3/16/18, 3AM (so 3/17/18, 3AM)
Possible Points: 20

Write a program that converts an NFA to a DFA. Your program must implement the algorithm described in textbook for doing the conversion. The behavior of your program should be exactly as described below.

The program should read input from a file that is specified as the first command line argument to the program, and write output to a file whose name is specified as the second command line argument to the program.

The format of the NFA description is as follows (this is close to the same as the DFA specification in the first programming assignment, except that there can be nondeterminism, epsilon transitions, and a transition does not have to be specified for all state/symbol combinations):

1. An integer that is the number of states in the NFA. This appears by itself on the first line of input. (In the remainder of the description, each state must be referred to by an integer in the range $[1, 2, ..., n]$, where n is the number of states.)

2. The alphabet of the NFA. This appears by itself on the second line of input. Every character in the line (not including the terminating newline character) is a symbol of the alphabet. The alphabet cannot include the letter e, as this is reserved for specifying epsilon transitions.

3. The transition function of the NFA. There will be one line of input per possible transition in the transition function, starting with the third line of input. The format of an entry is

   ```
   qa 'c' qb
   ```

   This entry indicates that if the NFA is in state `qb` and the next symbol scanned is a `c`, then the NFA can transition to `qb`. `c` can also be the letter e, in which case is represents an epsilon transition.

   `qa` and `qb` must be valid states; that is, qa $\in \{1, 2, \ldots, n\}$ and qb $\in \{1, 2, \ldots, n\}$. `c` must be in the alphabet or be the letter e (representing epsilon), and the single quotes must be present (even for the letter e)

   The entries of the transition function can appear in any order.

4. A blank line terminates the transition function entries. We need this because we don't know in advance how many transition function entries there will be.

5. An integer that is the start state of the NFA. This appears by itself on the first line after the transition function lines.

6. The set of accept states of the NFA. These appear together on the line following the line containing the start state.

The format of the DFA description is as follows (this is the same as for the first programming assignment):

1. An integer that is the number of states in the DFA. This appears by itself on the first line of input. (In the remainder of the description, each state must be referred to by an integer in the range $[1, 2, ..., n]$, where n is the number of states.)

2. The alphabet of the DFA. This appears by itself on the second line of input. Every character in the line (not including the terminating newline character) is a symbol of the alphabet.

3. The transition function of the DFA. There should be one line of input per entry in the transition function table, starting with the third line of input. The format of an entry is

   ```
   qa 'c' qb
   ```

   This entry indicates that if the DFA is in state `qb` and the next symbol scanned is a `c`, then the DFA transitions to `qb`.

   `qa` and `qb` must be valid states; that is, qa $\in \{1, 2, \ldots, n\}$ and qb $\in \{1, 2, \ldots, n\}$. `c` must be in the alphabet, and the single quotes must be present.

   The entries of the transition function can appear in any order.

4. An integer that is the start state of the DFA. This should appear by itself on the first line after the transition function lines.

5. The set of accept states of the DFA. These should appear together on the line following the line containing the start state.

Multiple entries on a line should be separated by 1 space character. No whitespace characters should precede the first entry on a line, and no characters should follow the last entry on a line (not counting the newline character).

Your program can assume that the input file will be correctly formatted, with no inconsistent data. (For example, if there are only 6 states in the NFA, there will be no transition function entries that specify a state of greater than 6 or less than 1.)

You can program in Java, C, or Python.

You should work with your programming partner, using the pair programming software development technique.

Use good programming style in writing your program, as specified by the document on blackboard.

Upload your source code only onto blackboard, on the page for the assignment, by the due date/time. Remember, no late assignments.

Sample input and correct output will be posted on the assignment webpage. Because the numbering of your states in the dfa could be different than mine, your output dfa's might appear different than mine. That does not mean they are not correct. So, the sample input/output folder has 4 files for each test case. For example, for test case 1, the files will be:

- `nfa1.txt`: contains the nfa specification
- `dfa1.txt`: contains the dfa equivalent to the nfa in `nfa1.txt`. If your dfa matches this exactly, great. If not, the numbering of states may just be different than mine. In that case, the next two files are helpful
- `dfa1Input.txt`: contains test inputs for the dfa you have created.
- `dfa1Output.txt`: contains correct results for the test inputs in the `dfa1Input.txt` file.