IT 328, Introduction to the Theory of Computation

Programming Assignment 2

Convert NFA to an equivalent DFA

Due date: March 9, 2017, Thursday, 50 points (70% on programs, 30% on report)

This is not a team work. Every Student has to do their own work.

Write a Java program to convert any given NFA to an equivalent DFA. You don't have to minimize the resulting DFA for this assignment (this will be your next assignment). Your programs will be compiled, run, and graded on our Unix system.

1. In my `/home/ADILSTU/ccli/Public/IT328/nfa` directory, there are some text files `nfa1`, `nfa2`, ... Each such file is the specification for an NFA described as follows. The format of NFA files is shown in the following example:

```
3
   a b c d
0:  {0} {0,1} {} {} {1,2}
1:  {} {1} {1} {1} {}
2:  {} {1,2} {1} {1} {}
0
{1,2}
```

   - The first line contains the number of states of the NFA; in this case, there are 3 states.
   - The second line contains $\Sigma = \{a, b, c, d\}$.
   - After that, each line is a state and its transition. The number to the left of the colon is the label of the state. After the colon, each set is the set of next states on the input alphabet corresponding to the alphabets in Σ in the same order as shown in the second line plus the λ-transition at the end of the same line.

     For example, the transitions of the NFA shown above from state 0 are: $\delta(0, a) = \{0\}, \delta(0, b) = \{0, 1\}, \delta(0, c) = \{\}$, $\delta(0, d) = \{\}$, and $\delta(0, \lambda) = \{0, 1, 2\}$.

     Note that, the last column is referred to λ-transition, and coming back to self is always implied; therefore, 0 is included in $\delta(0, \lambda) = \{0, 1, 2\}$, although 0 is not shown in the last column of state 0.
   - After every state's transitions are specified, there is a line to indicate the starting state. In this case, 0 is the starting state.
   - The last line contains the set of final states. In this case, $\{1, 2\}$

2. Make a new directory under you `~/IT328/nfa` on your unix account. All files related to this assignment should be prepared in this directory. (Note: ~ denotes your home directory.) You have to copy the following six files from my `/home/ADILSTU/ccli/Public/IT328/nfa` as your program's input: `nfa1`, `nfa2`, `nfa3`, `nfa4`, `nfa5`, and `inputStrings.txt`. There are more NFA files in my public directory, but these five are the require test NFA's for your program.

3. Name your main program as `NFA.java`. I will compile and run your program from the Unix command line as follows:

```
javac NFA.java
java NFA nfa2 inputStrings.txt
```

The first argument is an NFA specification described above, and the second argument is the file of input strings. Your program should check each string in `inputStrings.txt` and print out those that are accepted the NFA. Your program will not parse the input string using the NFA directly, which is rather difficult since we do not have non-determinstic computers. Instead, your program will convert the NFA to an equivalent DFA first, then parse the string using the DFA.

Your program should print out the following text:

(1) The original NFA

(2) An equivalent DFA (no need to minimized)

(3) The list of strings in `inputStrings.txt` that are accepted by the NFA (don't list those that are rejected).

Here is the example output of for my program on `nfa2` and a string file.

```
Sigma:a b
------
0:  (a,{1}) (b,{4}) ( ,{2})
1:  (a,{5}) (b,{2}) ( ,{})
2:  (a,{5}) (b,{}) ( ,{})
3:  (a,{}) (b,{}) ( ,{0})
4:  (a,{}) (b,{1}) ( ,{3})
5:  (a,{}) (b,{}) ( ,{4})
------
0:  Initial State
5:  Accepting State(s)

To DFA:
 Sigma:     a     b
 ------------------
     0:     1     2
     1:     1     3
     2:     1     3
     3:     1     3
 ------------------
0:  Initial State
1:  Accepting State(s)
The following strings are accepted:
aabaa
aaaaa
....
```

Note: You have to submit a hard copy of the direct output of your programs, but you don't have to submit a hard copy of your codes. If the NFA or DFA has more then 30 states, you just have to print out the first 30 states on your hard copy for submission, but your program still has to print all states on the screen. You should write up a brief report from which I should be able to understand your methods, data structures, and the difficulties you'd faced and your solutions. Your score is based on the correctness and documentation of your program, 50 points (70% on programs, 30% on report).

Final Step: After you've finished your work, or have decided that what you have is the final version for me to grade, select a secret name, say "`peekapoo`" as an example (you should chose your own), and that will be the secret directory, and should not give to anyone except the instructor. Run the following bash script program:

```
bash /home/ADILSTU/ccli/Public/IT328/copy328.sh peekapoo
```

Note that your original works have to be in the required directory `~/IT328/nfa/` as described in step 1 in order to let this script program correctly copy your works into the secret directory.

Final Words:

You have to follow the submission guidelines, i.e., cover page (that contains assignment number, student's names, student **ULID**, and secret directory), summary, source code(optional), output, folder, and so on. **No late work will be accepted.**