Automata Theory

Homework 1: due 12 October 2017

You are to write a program for regular expression matching:

```
rexp  'exp'  file
```

where `exp` is a regular expression and `file` is a file name. The file contains a list of strings over $\Sigma = \{0, 1\}$, one in each line. Your program should check if each string in `file` is a string represented by `exp`, and it consists of two subtasks.

1. Constructing an NFA from a regular expression

   - Input: a regular expression r
   - Output: an NFA equivalent to the regular expression r

   We assume that the input regular expression is completely parenthesized. An example regular expression is `((0.((0+1)*)).1)`.

   The first step is to change a regular expression to postfix form by using a stack for operators. The postfix form of the example is `001+*.1.` where an operator is in the position of the corresponding right parenthesis.

   The second step is to construct an NFA (transition table) from the postfix form by using a stack for intermediate NFAs. Use only *one* transition table to store *all* intermediate NFAs during the construction. For each intermediate NFA, you maintain its initial state and final state.

2. Running an NFA with an input string

   - Input: input string x and an NFA $N = (Q, \Sigma, \Delta, q_0, F)$
   - Output: yes if N accepts x; no otherwise

```
Run($N, x$)
    $C \leftarrow E(q_0)$
    for $i \leftarrow 1$ to $n$ do
        $C' \leftarrow \Delta(C, x_i)$
        $C \leftarrow E(C')$
    od
    if $C$ contains a final state
        then print “yes”
        else print “no” fi
end
```

- Run your program with at least two regular expressions. For each regular expression, run your NFA with at least two "Yes" strings and at least two "No" strings.
- Explain how your program works in your report.
- Hand in your report, programs, executable files, and an example running (with at least two regular expressions) by email to `sghong@theory.snu.ac.kr`.
- Write down the environment you run your program.
- Write comments appropriately in your program.