CS400 Compiler Construction
Fall 2008, Dr. Sheldon Liang

Homework & Quizzes #03

(25 points)

Due Date: One Week Away from today
(*Look at schedule in the syllabus*)

Your Name: _____ Your Score: _____

Objectives (*Lexical Analysis*):

☞ A token is a pair consisting of a token name and an optional attribute value. Automat is based on tokens, and regular expressions are combining token and automata with formalism processing

- A lexical unit is represented by a token, i.e., "identifier"
- The parser processes the token as input symbols
- Automata and regular expressions

Questions and points

Number of questions:	[10]
Positive points per question:	[2.5]
Negative points per question:	[0.5]

Make sure your name is on this handout before turning it in

⇨ Since we stress "learning through lecturing and reading", some questions designed for homework & quizzes stimulate students to listen, think, and read the text in our lectures. Be careful to find answer from the handouts and text or lab-testing (that is, seeking answer through program). ↵

Chapter 3: Formal Theory, Regular Expressions, and Finite Automata

Strings, Token, and their description
DFA's, automata with epsilon-rules.
Regular expressions and conversions with finite automata.

1. Divide the following C code into tokens:

```
float limitedSquare(x) float x {
        /* returns x-squared, but never more than 100 */
        return (x<=-10.0||x>=10.0)?100:x*x;
}
```

Which of the following is NOT one of the tokens?

- ☐ a) /* returns x-squared but never more than 100 */
- ☐ b) float
- ☐ c) >=
- ☐ d) ?

2. What are all the (a) prefixes (b) proper prefixes of the string abacabad?
Identify one of these from the list below.

- ☐ a) abacabad is a prefix and a proper prefix.
- ☐ b) ε is a prefix and a proper prefix.
- ☐ c) ε is a prefix but not a proper prefix.
- ☐ d) acada is a prefix and a proper prefix.

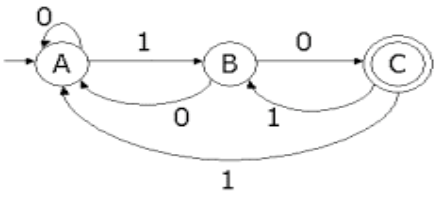
3. Here is a simple Lex program:

```
/* regular definitions */
letter    [A-Za-z]
uletter   [A-Z]
x         {uletter}+{letter}*
y         {letter}{uletter}
z         {letter}*

%%

{x}       {return(1);}
{y}       {return(2);}
{z}       {return(3);}
```

Suppose the input consists of a string of (upper- or lower-case) letters
w followed by a nonletter (e.g., a blank). Determine for which values of w the Lex program above will return 1, will return 2, and will return 3. Then, select from the list below, a correct pair (w,i), where the program returns *i* when the input is ***w***.

a) (a, 3) b) (Ab, 3) c) (a, 1) d) (Ab, 2)

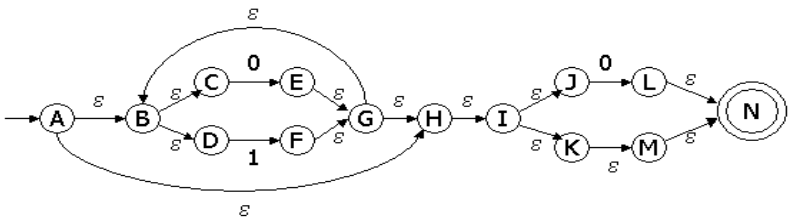
4. The following nondeterministic finite automaton:



accepts which of the following strings?

☐ a) 10001010 ☐ b) 1011101 ☐ c) 01010011 ☐ d) 00110100

5. Here is an epsilon-NFA:

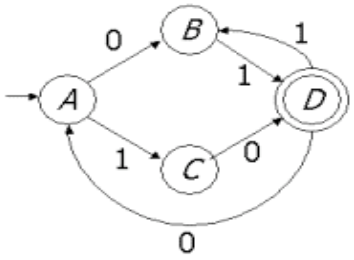


Suppose we construct an equivalent DFA by the construction of Algorithm 3.20 (p. 153). That is, start with the epsilon-closure of the start state A. For each set of states *S* we construct (which becomes one state of the DFA), look at the transitions from this set of states on input symbol 0. See where those transitions lead, and take the union of the epsilon-closures of all the states reached on 0. This set of states becomes a state of the DFA. Do the same for the transitions out of *S* on input 1. When we have found all the sets of epsilon-NFA states that are constructed in this way, we have the DFA and its transitions.

Carry out this construction of a DFA, and identify one of the states of this DFA (as a subset of the epsilon-NFA's states) from the list below.

☐ a) BCD ☐ b) BCDFGHIJK ☐ c) BCDEGHIJKLMN ☐ d) IJKMN

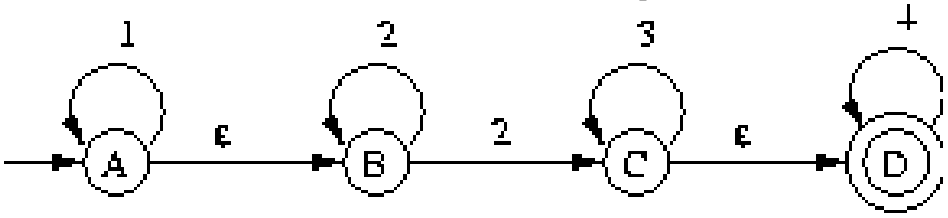
6. Examine the following DFA:



Identify in the list below the string that this automaton accepts.

a) 01111 b) 1001 c) 1011000 d) 10001

7. Here is a nondeterministic finite automaton with epsilon-transitions:



Suppose we use the subset construction of Algorithm 3.2 (p. 118) to convert this epsilon-NFA to a deterministic finite automaton with a dead state, with all transitions defined, and with no state that is inaccessible from the start state. Which of the following would be a transition of the DFA?
Note: we use S-x->T to say that the DFA has a transition on input x from state S to state T.

☐ a) {B,C,D}-3->{C} ☐ b) {A,B}-2->{B,C,D} ☐ c) {B}-2->{B,C} ☐ d) {A,B}-1->{A}

8. The UNIX-style regular expression [a-e]*f?bc* generates which of the following strings?

☐ a) edcffb ☐ b) dcebfbcc ☐ c) Bbc ☐ d) a-ea-efbc

9. Apply the McNaughton-Yamada-Thompson construction in Section 3.7.4 (p. 159) to convert the regular expression (0+1)*(0+ε) to an epsilon-NFA. Count

1. The number of states.
2. The number of states that have more than one out-arc.
3. The number of states that have more than one in-arc.
4. The number of arcs labeled ε

Then, identify the true statement about your epsilon-NFA from the list below:

☐ a) There are 12 states with more than one arc in.
☐ b) There are 16 states.
☐ c) There are 14 arcs labeled ε.
☐ d) There are 6 states with more than one arc out.

10. Suppose that in some language a real number is represented by the following elements, which must be in order, if they occur at all (i.e., are not optional):

1. An optional minus sign (-).
2. One or more digits (0 through 9).
3. A decimal point.
4. Zero or more digits.
5. An optional exponent, consisting of:
   - An "E" in upper or lower case.
   - An optional minus sign.
   - One or more digits.

Write a regular expression for real numbers in the form described below, one of the possible regular expressions that denotes all these real numbers and nothing else.

☐ a) [-]?[0-9]*.[0-9]*([Ee][-]?[0-9]+)?
☐ b) -?[0-9]*.[0-9]*((E|e)-?[0-9]+)?
☐ c) [-]?[0-9]+.[0-9]*([Ee][-]?[0-9]+)?
☐ d) -?[0-9][0-9]*.[0-9]*(E|e)-?[0-9][0-9]*