

Please print clearly:

Name: **SOLUTION**

Login: @ucsc.edu

No books; No calculator; No computer; No email; No internet; No notes; No phone. Neatness counts! Do your scratch work elsewhere and enter only your final answer into the spaces provided.

1. The ETF grammar is presented here. Rewrite it so that the + and * operators are both right-associative, and also make + have a higher precedence than *. [2✓]

$E \rightarrow E + T$
 $E \rightarrow T$
 $T \rightarrow T * F$
 $T \rightarrow F$
 $F \rightarrow (E)$
 $F \rightarrow i$

$E \rightarrow T * E$
 $E \rightarrow T$
 $T \rightarrow F + T$
 $T \rightarrow F$

$F \rightarrow (E)$
 $F \rightarrow i$

2. Draw abstract syntax trees (not parse trees) for each of the following C expressions. [3✓]

| $a * b + c * d$ | $(a + b) * c * d$ | $a / b * c + d$ |
|-----------------|-------------------|-----------------|
| | | |

3. Given the nondeterministic finite αὐτόματον shown here:

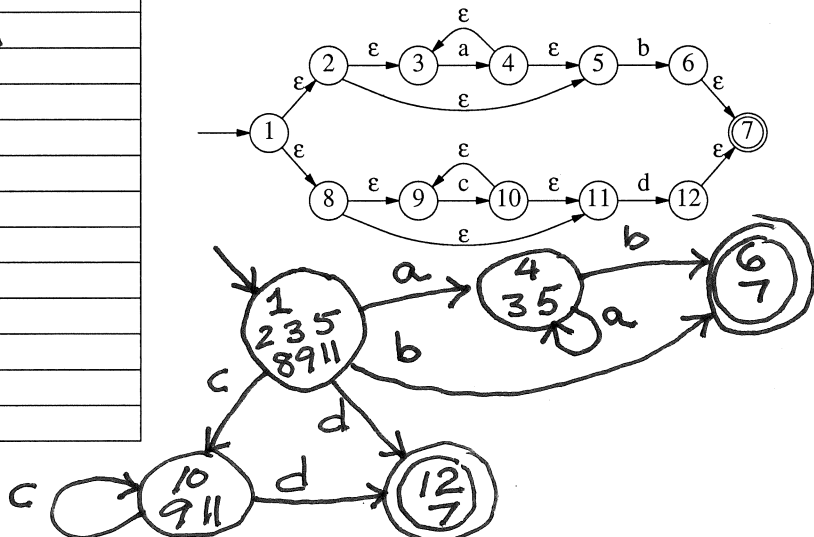
- (a) Write the regular expression that was used by Thompson's construction to create this NFA. [1✓]

$a * b | c * d$

- (b) Fill in the table of ε-closures for each state. [2✓]

- (c) Use the subset algorithm to construct the equivalent *deterministic* finite αὐτόματον. Inside each state of the DFA, write the numbers of the NFA states to which it corresponds. [2✓]

| state s | ϵ -closure (s) |
|-----------|-----------------------------|
| 1 | 1 2 3 5 8 9 11 |
| 2 | 2 3 5 |
| 3 | 3 |
| 4 | 4 3 5 |
| 5 | 5 |
| 6 | 6 7 |
| 7 | 7 |
| 8 | 8 9 11 |
| 9 | 9 |
| 10 | 10 9 11 |
| 11 | 11 |
| 12 | 12 7 |



4. Write **flex** regular expressions for each of the following descriptions of tokens. [3✓]

(a) An identifier consists of upper- and lower-case letters and digits and underscores. It may not begin with a digit or an underscore, nor end with an underscore. Underscores may not appear next to each other.

`[a-zA-Z](_?[a-zA-Z0-9])*`

(b) A string consists of a sequence of characters between either double quotes or single quotes, but the ending quote must be the same as the starting quote. In between the quotes may be any characters except for newlines and the quotes that begin and end the string.

$\backslash "[^"\backslash n]^*\backslash " | '[^'\backslash n]^*\backslash ' |$

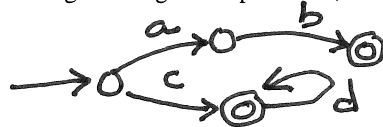
(c) A number consists of a sequence of decimal digits with a decimal point somewhere in between the digits, but the decimal point may not appear on the front or on the end. An optional exponent may appear, which consists of an **E** or an **e**, followed optionally by a + or - sign, followed by one or more digits.

an E or an e, followed optionally by a + or - sign, followed by one or more digits.

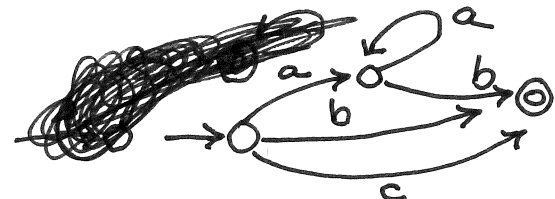
$[0-9]^+ \backslash \cdot [0-9]^+ ([Ee][+-]? [0-9]^+)^?$

5. For each of the following **flex** regular expressions, draw a **deterministic** finite $\alpha\upsilon\tau\omicron\mu\alpha\tau\omicron\nu$, using as few states as possible. [4✓]

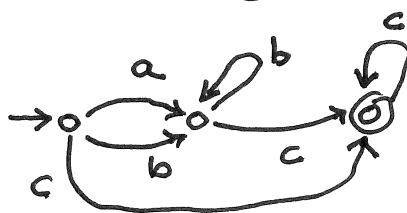
(a) $ab|cd^*$



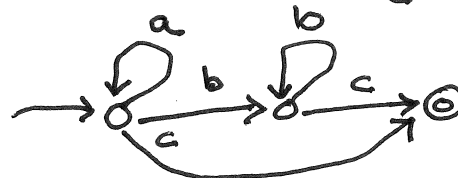
(b) $a^*b|c$




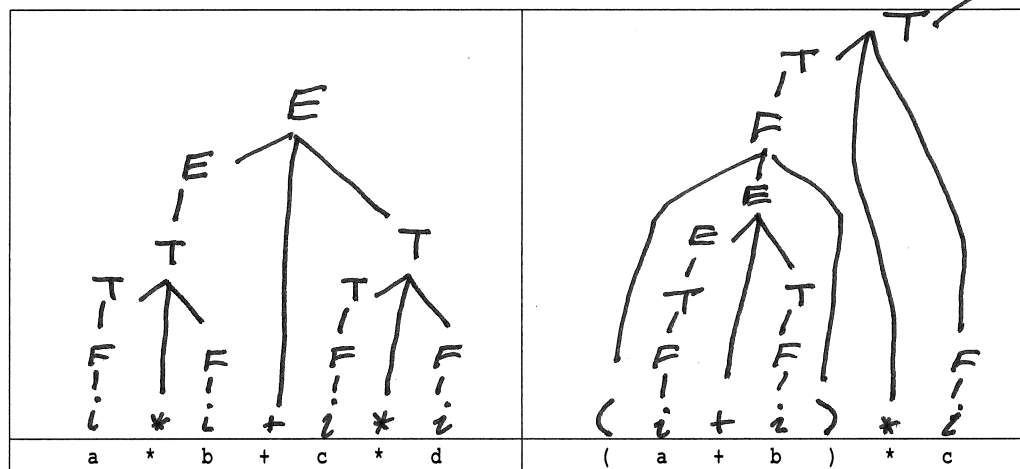
(c) $a^?b^*c^+$



(d) $a * b * c$



6. Draw parse trees (not abstract syntax trees) for the following expressions, given the ETF grammar at the left. The lexical symbols are below the middle line. Match each with a terminal symbol above the middle line and draw the trees. [2✓] 

$$\begin{aligned} E &\rightarrow E + T \\ E &\rightarrow T \\ T &\rightarrow T * F \\ T &\rightarrow F \\ F &\rightarrow (E) \\ F &\rightarrow i \end{aligned}$$


7. A context-free grammar $G = \langle V_N, V_T, P, S \rangle$. If P is the ETF grammar, fill in the blanks for the other three components. [1✓]

$$V_N = \{ \underline{E} \quad \underline{T} \quad \underline{F} \}, \quad V_T = \{ + \quad * \quad (\quad) \quad i \}, \quad S = \underline{E}$$

Multiple choice. To the *left* of each question, write the letter that indicates your answer. Write **Z** if you don't want to risk a wrong answer. Wrong answers are worth negative points. **[12✓]**

| | | | |
|--------------------------------------|----|------------------------|-------|
| number of correct answers | | $\times 1 =$ | $= a$ |
| number of wrong answers | | $\times \frac{1}{2} =$ | $= b$ |
| number of missing answers | | $\times 0 =$ | 0 |
| column total $c = \max(a - b, 0)$ | 12 | | $= c$ |

1. Thompson's construction :

B

- (A) converts C code into assembly language.
(B) converts a regular expression into an NFA.
(C) converts an NFA into a DFA.
(D) minimizes a DFA.

2. Given a regular expression r used to construct a DFA, and a scanned string s , the worst-case size of a DFA is :



- (A) $O(2^{|r|})$
 (B) $O(2^{|s|})$
 (C) $O(|r|^2)$
 (D) $O(|r| \times |s|)$

3. Which pattern might match a string of zero characters?

A

- (A) a^*b^*
(B) a^*b^+
(C) a^+b^*
(D) a^+b^+

4. Which pattern will match the following string?
bbaaaabaaaabababa

B

- (A) $(ba)^+$
(B) $(b^+a^+)^*$
(C) $(ba^+)^*$
(D) b^+a^+

5. Given a regular expression r used to construct an NFA, and a scanned string s , the speed of the scan is : _____

D

- (A) $O(2^{|r|})$
 (B) $O(2^{|s|})$
 (C) $O(|r|^2)$
 (D) $O(|r| \times |s|)$

6. The **flex** pattern . (dot) is the same as

D

- (A) [/n]
(B) [\n]
(C) [^/n]
(D) [^\n]

7. What is not possible in an NFA ?

C

- (A) epsilon transitions
(B) multiple final states
(C) multiple initial states
(D) multiple outgoing transitions from a given state with the same label

8. How many tokens in the following Java statement?

B

```
out.printf( "%d\n",n);
```

- (A) 7
(B) 9
(C) 11
(D) 13

9. Which is equivalent to the following regex ?

 $a|bc^*$

D

- (A) $((a|b)c)^*$
 (B) $(a|b)(c^*)$
 (C) $a|((bc)^*)$
 (D) $a|(b(c^*))$

10. Determining that `int` is a reserved word, not an identifier, is done by :

B

- (A) preprocessor
(B) scanner
(C) parser
(D) magic numbers

11. The value returned by `yylex` is :

D

- (A) a non-terminal symbol
(B) a pointer to a struct token
(C) a pointer to the lexical buffer
(D) an integer token code

12. Which of the following regexes will produce the smallest DFA (DFA with the fewest states) ?

B

- (A) a
(B) a^*
(C) a^\dagger
(D) ab

