$Id: cmps104a-2015q4-exam1.mm,v 1.53 2015-10-20 15:05:57-07 - - $

page 1 page 2 page 3 Total / 32

***Please print clearly :***

**Name :**

**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. Draw ***abstract syntax trees*** for each of the following C expressions : **[3✔]**

| `a * b / c * d - e` | `a = b * c + d` | `a + b / c + d * e` |
|---|---|---|
| | | |

2. Write `flex` regular expressions for each of the following : **[5✔]**

   (a) A number which consists of a sequence of decimal digits, possibly with a decimal point. If a decimal point appears, it must be preceded and followed by digits. A number has an optional exponent, which is the letter "e" in upper- or lower-case, followed by an optional minus sign, and then one or more digits.

   (b) A quoted string which starts and ends with a double quote. Between the quotes may be zero or more occurrences of any character except a newline. If a backslash or a quote appears in the string, it must be excaped by a backslash.

   (c) Write two patterns in the correct order. One is the keyword `if`, which may not be recognized as an identifier. The other is an identifier which consists of one or more upper- or lower-case letters or decimal digits, but which may not begin with a digit.

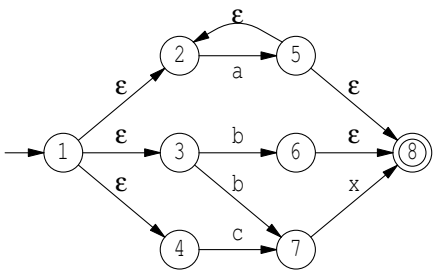
   (d) Write two patterns : one matches a correct octal constant in C, and the other matches a correct hexadecimal constant in C.

   (e) A pattern which recognizes a list-extracting function in Scheme. The first letter is always a lower-case "c" and the last letter is always a lower-case "r". Between them are one or more occurrences of either the lower-case letters "a" or "d". Examples : `car`, `cdr`, `caar`, `cadr`, `cdar`, `cddr`, etc.

3. Using ***Thompson's construction***, draw the NFA from the following regular expression. **[2✔]**

```
ab*|cd
```

4. Given the NFA shown here, compute the ε-closure of each state and fill in the table. **[2✔]**




| state $s$ | ε-closure($s$) |
|---|---|
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| 6 | |
| 7 | |
| 8 | |

5. Given the ETF grammar shown at the left, draw ***parse trees*** for each of the following expressions: **[3✔]**

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

| `a * b + c` | `( a + b ) * c` | `a * b * c` |
|---|---|---|

6. Given the ETF grammar in the previous question and the fact that a grammar $G = \langle V_N, V_T, P, S \rangle$, fill in each of the following: **[1✔]**

$V_N = \{$_____$\}$, $V_T = \{$_____$\}$, and $S =$ _____.

7. Rewrite the ETF grammar from the previous question so that both the `+` and `*` operators are right associative, and have the same precedence. **[2✔]**

8. Draw deterministic finite automata for each of the following regular expressions. Use the ***minimum*** possible number of states. **[2✔]**

   (a) `ab*c|d`

   (b) `a+b+c+`

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 | | × 1 = | $= a$ |
| number of wrong answers | | × ½ = | $= b$ |
| number of missing answers | | × 0 = | 0 |
| column total $c = \max(a - b, 0)$ | 12 | | $= c$ |

1. The function **yyparse** implements what kind of machine ?
   (A) Turing machine
   (B) finite state automaton
   (C) linear bounded automaton
   (D) pushdown automaton

2. The function **yylex** implements what kind of machine ?
   (A) Turing machine
   (B) finite state automaton
   (C) linear bounded automaton
   (D) pushdown automaton

3. What kind of language is recognized by **flex** ?
   (A) context-free
   (B) context-sensitive
   (C) recursively enumerable
   (D) regular

4. What kind of language is recognized by **bison** ?
   (A) context-free
   (B) context-sensitive
   (C) recursively enumerable
   (D) regular

5. The regex `ab|c*d` is equivalent to :
   (A) `(a(b|c))*d`
   (B) `(ab)|((c*)d)`
   (C) `a((b|c)*)d`
   (D) `a(b|(c*))d`

6. Whenever **yylex** returns, what variable points at the lexeme most recently matched ?
   (A) **yydebug**
   (B) **yyin**
   (C) **yylexeme**
   (D) **yytext**

7. Given the ETF grammar discussed in class, which rule unambiguously shows that the operator + is left associative and can appear multiple times in an expression ?
   (A) $E \to E + E$
   (B) $E \to E + T$
   (C) $E \to T + E$
   (D) $E \to T + T$

8. Access in time $O(1)$ to the string table is provided by :
   (A) **map<string>**
   (B) **set<string>**
   (C) **unordered_map<string>**
   (D) **unordered_set<string>**

9. A DFA is constructed from a regular expression $r$ and used to scan a string $s$. How fast is the scan ?
   (A) $O(1)$
   (B) $O(2^{|r|})$
   (C) $O(|r| \times |s|)$
   (D) $O(|s|)$

10. If D is the set of languages recognizable by a DFA, and N is the set of languages recognizable by an NFA, then :
    (A) $D \equiv N$
    (B) $D \subset N$
    (C) $D \supset N$
    (D) Not enough information to decide, because it depends on the particular grammar in question.

11. Given a grammar $G = \langle V_N, V_T, P, S \rangle$, tokens returned by **yylex** deal strictly with which set ?
    (A) $V_N$
    (B) $V_T$
    (C) $P$
    (D) $S$

12. Which pattern will recognize a Java or C++ comment ?
    (A) `"//"[^/n]*`
    (B) `"//"[^\n]*`
    (C) `"\\"[^/n]*`
    (D) `"\\"[^\n]*`