CMSC 331 Midterm Exam, Fall 2013

Name: _________________________________

UMBC username: _______________________________

You will have seventy-five (75) minutes to complete this closed book exam. Use the backs of these pages if you need more room for your answers. Describe any assumptions you make in solving a problem. We reserve the right to assign partial credit, and to deduct points for answers that are needlessly wordy.

1. True/False [30 points]

For each of the following questions, indicate if the statement is TRUE or FALSE. (2 points each)

   1.1 The “Von Neumann” computer architecture is still used as the basis for most computers today.
   1.2 Any finite language can be defined by a regular expression.
   1.3 In Attribute Grammars, synthesized attributes allow information to be passed from the root of the parse tree down to the other nodes.
   1.4 Most modern programming languages are too complex to be specified using regular expressions alone.
   1.5 A deterministic finite state automaton (DFSA) requires that for each node, a symbol be associated with no more than one arc that leaves that node.
   1.6 The C++ programming language got a lot of its ideas from the older versions of Java.
   1.7 The assert macro in C (and C++) can be used to show that a program is running correctly.
   1.8 In a regular expression, the Kleene star operator “*” indicates that an expression can appear one or more times.
   1.9 In most languages, parentheses can be used to override the rules of operator associativity, whatever they happen to be.
   1.10 The Python scripting language generates code that is then run using the Java Virtual Machine.
   1.11 In a context-free grammar, no symbol can be both a terminal and a non-terminal symbol.
   1.12 In a recursive descent parser, care must be taken to make sure that left recursive productions are converted to another form.
   1.13 The COBOL language was designed for scientific and engineering applications.
   1.14 Lexical analyzers often use regular expressions to describe the tokens in a language.
   1.15 Every DFSA has exactly one start state and one final state.
2. Parsing Expressions [20]

2.1 (10 points) Using the grammar shown below, draw a parse tree for the expression \(2 * 7 + 4 * 3\)

\[
\begin{align*}
\text{Expr} & \rightarrow \text{Term} + \text{Term} \\
\text{Expr} & \rightarrow \text{Term} \\
\text{Term} & \rightarrow \text{Factor} * \text{Factor} \\
\text{Term} & \rightarrow \text{Factor} \\
\text{Factor} & \rightarrow 0|1|2|3|4|5|6|7|8|9
\end{align*}
\]

2.2 (10 points) Consider the attribute grammar shown below, where the symbols Term and Factor are subscripted to distinguish different instances of those symbols. Add the numerical value of the val attribute to each interior node in the parse tree you wrote for the previous problem.

What is the value associated with the root node? □□□□□□□□□□

Re-write the tree here, or on the tree you wrote above.

\[
\begin{align*}
\text{Expr} & \rightarrow \text{Term}_1 + \text{Term}_2 \\
& \{ \text{Expr.val} = \text{Term}_1.val + \text{Term}_2.val \} \\
\text{Expr} & \rightarrow \text{Term} \\
& \{ \text{Expr.val} = \text{Term.val} \} \\
\text{Term} & \rightarrow \text{Factor}_1 * \text{Factor}_2 \\
& \{ \text{Term.val} = \text{Factor}_1.val * \text{Factor}_2.val \} \\
\text{Term} & \rightarrow \text{Factor} \\
& \{ \text{Term.val} = \text{Factor.val} \} \\
\text{Factor} & \rightarrow 0|1|2|3|4|5|6|7|8|9 \\
& \{ \text{Factor.val} = \text{the corresponding integer} \}
\end{align*}
\]
3. Finite State Machines (and just a little C++) (20 Points)

In the C++ handout we described a rational numbers class. In the printed representation of rational numbers, we could have an optional negative sign, followed by any number of digits, a division sign (/) and any number of digits, not all of which are zero. (That prevents the denominator from being zero, of course) By using this definition

4.1 (10 points) Draw a DFA for rationals using the description above.

4.2 (10 points) Provide a regular expression that describes the rationals, again according to the description above.
4. Axiomatic Semantics (10 Points)

3.1 (5 points) Show the weakest precondition (i.e. the \{ ?? \}) for the following statement:

\[
\begin{align*}
&\{ ?? \} \\
&\text{if } (x < 3) \text{ then} \\
&\quad y = x + 2 \\
&\text{else } \{ x \geq 3 \} \\
&\quad y = x - 1; \\
&\{ y > 0 \}
\end{align*}
\]

3.2 (5 points) Show the weakest precondition (i.e. the \{ ?? \}) for this code segment:

\[
\begin{align*}
&\{ ?? \} \\
&x = (y / 10) + 2; \\
&y = 4 \times x; \\
&\{ y > 12 \}
\end{align*}
\]
5. Understanding Scheme (20 points)

Consider the following Scheme function. Assume it is syntactically correct and executes to completion as shown. What is the output for the expressions given?

```
(define aFunc(n)
  (cond
   ((= n 0) 0)
   ((= n 1) 1)
   (else (+ n (aFunc (- n 1))))))
```

(5 points) The expression (aFunc 3) evaluates to ________________

(5 points) The expression (aFunc (car (cdr '(2 4 6 4 2)))) evaluates to _______

(10 points) Write a Scheme (or Lisp) function that takes two arguments: a list (of any type) called aList, and a positive integer n. The function returns the nth element of the list. The expression (nth aList 1) returns the first element of the input list, or the null list ‘() if aList itself is empty. Here is the first line of code:

```
(define nth(aList, n)
```