Program 1 - Bit Manipulation

Due Date: -- 11/13/14 --

The Grammar shown below describes the bitwise manipulation of Hexidecimal numbers. It is left-recursive and indicates proper associativity and precedence.

E → E "|" A	bitwise OR
E → A	
A → A "^" B	bitwise XOR
A → B	
B → B "&" C	bitwise AND
B → C	
C → "<" C	bitwise shift left 1
C → ">" C	bitwise shift right 1
C → "~" C	bitwise NOT
C → "(" E ")"	
C → hex	

The corresponding *right-recursive* attribute Grammar is provided on the second page. You are to write a program using **recursive descent parsing that implement the designated Inherited and Synthesized attributes shown on page 2** to compute correctly the expressions shown below. *You cannot use more than 2 global / non-local variables, and they should be to hold the Operator and HexNumber as detected by the Lexical Analyzer.*

f&a
b|3
f^1
~0
>>f
<1
3|6&c
(3|6)&c
(3|c)&6
~~f
f^>f
c&3&f
<3|3
~(e^7)
>>>>~(a^c)
~(>1|>2|>4|>8)^~5
(d^2|1)&(<<2|c)
((f&>9)|(~3^8)|(~c|b))
>f|<f&1
(>(<1&>f)|8|9)^(~3&7)
~(><8|<>1)

Bitwise Manipulation of Hexidecimal Numbers

$E \rightarrow A\ EE$
$EE.st = A.val$
$E.val = EE.val$

$EE_1 \rightarrow |\ A\ EE_2$
$EE_2.st = EE_1.st \mid A.val$ ("|" bitwise OR)
$EE_1.val = EE_2.val$

$EE \rightarrow \varepsilon$
$EE.val = EE.st$

$A \rightarrow B\ AA$
$AA.st = B.val$
$A.val = AA.val$

$AA_1 \rightarrow \wedge\ B\ AA_2$
$AA_2.st = AA_1.st \wedge B.val$ ("^" bitwise XOR)
$AA_1.val = AA_2.val$

$AA \rightarrow \varepsilon$
$AA.val = AA.st$

$B \rightarrow C\ BB$
$BB.st = C.val$
$B.val = BB.val$

$BB_1 \rightarrow \&\ C\ BB_2$
$BB_2.st = BB_1.st \& C.val$ ("&" bitwise AND)
$BB_1.val = BB_2.val$

$BB \rightarrow \varepsilon$
$BB.val = BB.st$

$C_1 \rightarrow <C_2$
$C_1.val = C_2.val << 1$ ("<<" bitwise shift left one)

$C_1 \rightarrow >C_2$
$C_1.val = C_2.val >> 1$ (">>" bitwise shift right one)

$C_1 \rightarrow \sim C_2$
$C_1.val = \sim C_2.val$ ("~" bitwise NOT)

$C \rightarrow (\ E\ )$
$C.val = E.val$

$C \rightarrow hex$
$C.val = hex.val$