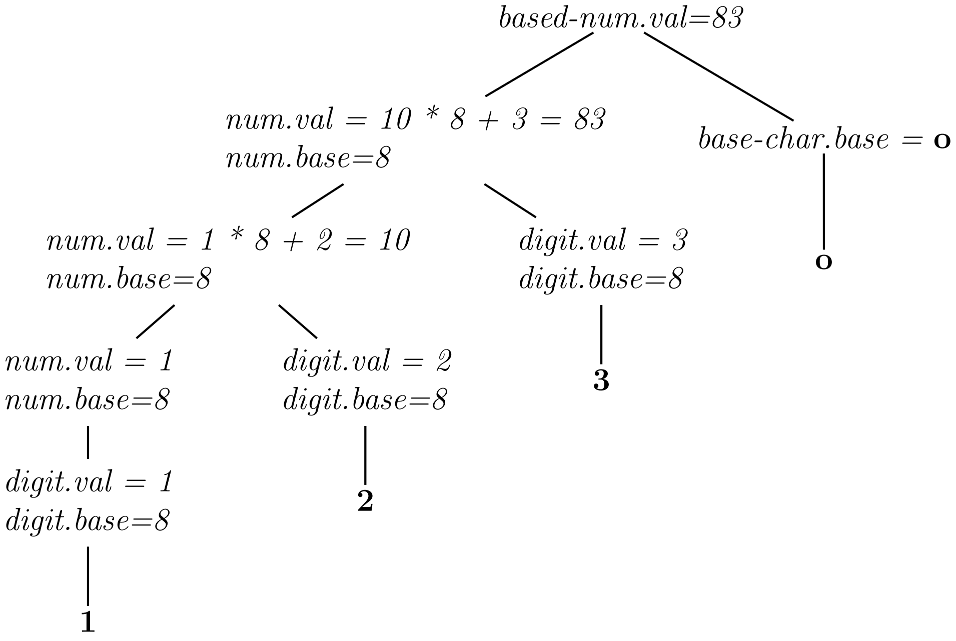
COMP3131/9102: Programming Languages and Compilers
Week 8 Tutorial Solutions
Attribute Grammars

(a)

Production			Semantic Rules
based-num	→	*num base-char*	*based-num.val* = *num.val* *num.base* = *base-char.base*
base-char	→	**o**	*base-char.base=8*
base-char	→	**d**	*base-char.base=10*
num_1	→	num_2 *digit*	$num_1.val$ = if ($num_2.val$ == error || $digit.val$ == error) then error else $num_2.val * num_1.base + digit.val$ $num_2.base = num_1.base$ $digit.base = num_1.base$
num	→	*digit*	*num.val* = *digit.val* *digit.base* = *num.base*
digit	→	**0**	*digit.val = 0*
digit	→	**1**	*digit.val = 1*
			. . .
digit	→	**7**	*digit.val = 7*
digit	→	**8**	*digit.val* = if (*digit.base == 8) then error else 8*
digit	→	**9**	*digit.val* = if (*digit.base == 8) then error else 9*

(b)



(c) The attribute grammar is not L-attributed because in the first production, *num.base* depends on *base-char.base*, where the nonterminal *base-char* is on the right of the nonterminal *num*.

(d) No. The grammar is not L-attributed. The value of a number cannot be computed unless the base is known. But the base is not known until after the number has been parsed.

(e)

```
void EvalNum (AST N) {
  if (N is a based-number node) {
    EvalNum (right child of N);
    Assign base of right child of N to base of left child of N;
    EvalNum (left child of N);
    Assign val of left child of N to N.val
  } else if (N is a num node) {
    Assign N.base to base of left child of N;
    EvalNum (left child of N);
    if (right child of N exists) /* num -> num digit */
      Assign N.base to base of right child of N;
      EvalNum (right child of N);
      if (vals of both left and right children of N are not error)
        N.val = (val of left child of N) * N.base + (val of right child of N)
      else
        error
    else /* num -> digit */
     N.val = val of left child
  }
}
```