SECTION 10.3 Tree Traversal

2. See the comments for the solution to Exercise 1. The order is $0 < 1 < 1.1 < 1.1.1 < 1.1.1.1 < 1.1.1.2 < 1.1.2 < 1.2 < 2$.

4. **a)** The vertex is at level 5; it is clear that an address (other than 0) of length l gives a vertex at level l.

b) We obtain the address of the parent by deleting the last number in the address of the vertex. Therefore the parent is 3.4.5.2.

c) Since v is the fourth child, it has at least three siblings.

d) We know that v's parent must have at least 1 sibling, its grandparent must have at least 4, its great-grandparent at least 3, and its great-great-grandparent at least 2. Adding to this count the fact that v has 5 ancestors and 3 siblings (and not forgetting to count v itself), we obtain a total of 19 vertices in the tree.

e) The other addresses are 0 together with all prefixes of v and the all the addresses that can be obtained from v or prefixes of v by making the last number smaller. Thus we have 0, 1, 2, 3, 3.1, 3.2, 3.3, 3.4, 3.4.1, 3.4.2, 3.4.3, 3.4.4, 3.4.5, 3.4.5.1, 3.4.5.2, 3.4.5.2.1, 3.4.5.2.2, and 3.4.5.2.3.

6. **a)** The following tree has these addresses for its leaves. We construct it by starting from the beginning of the list and drawing the parts of the tree that are made necessary by the given leaves. First of course there must be a root. Then since the first leaf is labeled 1.1.1, there must be a first child of the root, a first child of this child, and a first child of this latter child, which is then a leaf. Next there must be the second child of the root's first grandchild (1.1.2), and then a second child of the first child of the root (1.2). We continue in this manner until the entire tree is drawn.

b) If there is such a tree, then the address 2.4.1 must occur since the address 2.4.2 does (the parent of 2.4.2.1). The vertex with that address must either be a leaf or have a descendant that is a leaf. The address of any such leaf must begin 2.4.1. Since no such address is in the list, we conclude that the answer to the question is no.

c) No such tree is possible, since the vertex with address 1.2.2 is not a leaf (it has a child 1.2.2.1 in the list).

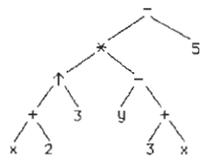
8. See the comments in the solution to Exercise 7 for the procedure. The only difference here is that some vertices have more than two children: after listing such a vertex, we list the vertices of its subtrees, in preorder, from left to right. The answer is $a, b, d, e, i, j, m, n, o, c, f, g, h, k, l, p$.

10. The left subtree of the root comes first, namely the tree rooted at b. There again the left subtree comes first, so the list begins with d. After that comes b, the root of this subtree, and then the right subtree of b, namely (in order) f, e, and g. Then comes the root of the entire tree and finally its right child. Thus the answer is d, b, f, e, g, a, c.

12. This is similar to Exercise 11. The answer is $k, e, l, m, b, f, r, n, s, g, a, c, o, h, d, i, p, j, q$.

14. The procedure is the same as in Exercise 13, except that some vertices have more than two children here: before listing such a vertex, we list the vertices of its subtrees, in postorder, from left to right. The answer is $d, i, m, n, o, j, e, b, f, g, k, p, l, h, c, a$.

16. **a)** We build the tree from the top down while analyzing the expression by identifying the outermost operation at each stage. The outermost operation in this expression is the final subtraction. Therefore the tree has $-$ at its root, with the two operands as the subtrees at the root. The right operand is clearly 5, so the right child of the root is 5. The left operand is the result of a multiplication, so the left subtree has $*$ as its root. We continue recursively in this way until the entire tree is constructed.

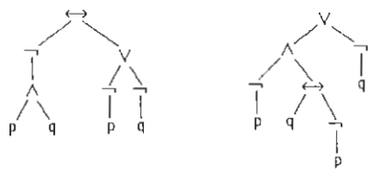


b) We can read off the answer from the picture we have just drawn simply by listing the vertices of the tree in preorder: First list the root, then the left subtree in preorder, then the right subtree in preorder. Therefore the answer is $-*\uparrow+x\,2\,3-y+3\,x\,5$.

c) We can read off the answer from the picture we have just drawn simply by listing the vertices of the tree in postorder: $x\,2+3\uparrow y\,3\,x+-*5-$.

d) The infix expression is just the given expression, fully parenthesized: $((((x+2)\uparrow 3)*(y-(3+x)))-5)$. This corresponds to traversing the tree in inorder, putting in a left parenthesis whenever we go down to a left child and putting in a right parenthesis whenever we come up from a right child.

18. **a)** This exercise is similar to the previous few exercises. The only difference is that some portions of the tree represent the unary operation of negation ($\neg$). In the first tree, for example, the left subtree represents the expression $\neg(p \wedge q)$, so the root is the negation symbol, and the only child of this root is the tree for the expression $p \wedge q$.



Since this exercise is similar to previous exercises, we will not go into the details of obtaining the different expressions. The only difference is that negation ($\neg$) is a unary operator; we show it preceding its operand in infix notation, even though it would follow it in an inorder traversal of the expression tree.

b) $\leftrightarrow \neg \wedge pq \vee \neg p \neg q$ and $\vee \wedge \neg p \leftrightarrow q \neg p \neg q$

c) $pq \wedge \neg p \neg q \neg \vee \leftrightarrow$ and $p \neg q p \neg \leftrightarrow \wedge q \neg \vee$

d) $((\neg(p \wedge q)) \leftrightarrow ((\neg p) \vee (\neg q)))$ and $(((\neg p) \wedge (q \leftrightarrow (\neg p))) \vee (\neg q))$

20. This requires fairly careful counting. Let us work from the outside in. There are four symbols that can be the outermost operation: the first $\neg$, the $\wedge$, the $\leftrightarrow$, and the $\vee$. Let us first consider the cases in which the first $\neg$ is the outermost operation, necessarily applied, then, to the rest of the expression. Then there are three possible choices for the outermost operation of the rest: the $\wedge$, the $\leftrightarrow$, and the $\vee$. Let us assume first that it is the $\wedge$. Then there are two choices for the outermost operation of the rest of the expression: the $\leftrightarrow$ and the $\vee$. If it is the $\leftrightarrow$, then there are two ways to parenthesize the rest—depending on whether the second $\neg$ applies to the disjunction or only to the p. Backing up, we next consider the case in which the $\vee$ is outermost operation among the last seven symbols, rather than the $\leftrightarrow$. In this case there are no further choices. We then back up again and assume that the $\leftrightarrow$, rather than the $\wedge$, is the second outermost operation. In this case there are two possibilities for completing the parenthesization (involving the second $\neg$). If the $\vee$ is the second outermost operation, then again there are two possibilities, depending on whether the $\wedge$ or the $\leftrightarrow$ is applied first. Thus in the case in which the outermost operation is the first $\neg$, we have counted 7 ways to parenthesize the expression:

$$(\neg(p \wedge (q \leftrightarrow (\neg(p \vee (\neg q))))))$$

$$(\neg(p \wedge (q \leftrightarrow ((\neg p) \vee (\neg q)))))$$

$$(\neg(p \wedge ((q \leftrightarrow (\neg p)) \vee (\neg q))))$$

$$(\neg((p \wedge q) \leftrightarrow (\neg(p \vee (\neg q)))))$$

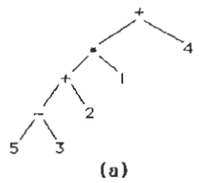
$$(\neg((p \wedge q) \leftrightarrow ((\neg p) \vee (\neg q))))$$

$$(\neg((p \wedge (q \leftrightarrow (\neg p))) \vee (\neg q)))$$

$$(\neg(((p \wedge q) \leftrightarrow (\neg p)) \vee (\neg q)))$$

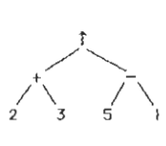
The other three cases are similar, giving us 3 possibilities if the $\wedge$ is the outermost operation, 4 if the $\leftrightarrow$ is, and 5 if the $\vee$ is. Therefore the answer is $7 + 3 + 4 + 5 = 19$.

22. We work from the beginning of the expression. In part (**a**) the root of the tree is necessarily the first $+$. We then use up as much of the rest of the expression as needed to construct the left subtree of the root. The root of this left subtree is the $*$, and its left subtree is as much of the rest of the expression as is needed. We continue in this way, making our way to the subtree consisting of root $-$ and children 5 and 3. Then the 2 must be the right child of the second $+$, the 1 must be the right child of the $*$, and the 4 must be the right child of the root. The result is shown here.

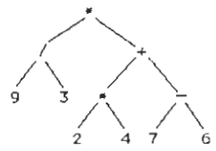


(a)

In infix form we have $((((5 - 3) + 2) * 1) + 4)$. The other two trees are constructed in a similar manner.



(b)



(c)

The infix expressions are therefore $((2 + 3) \uparrow (5 - 1))$ and $((9/3) * ((2 * 4) + (7 - 6)))$, respectively.

24. We exhibit the answers by showing with parentheses the operation that is applied next, working from left to right (it always involves the first occurrence of an operator symbol).

a) $5\,(2\,1\,-) - 3\,1\,4 + + * = (5\,1\,-)\,3\,1\,4 + + * = 4\,3\,(1\,4\,+) + * = 4\,(3\,5\,+) * = (4\,8\,*) = 32$

b) $(9\,3\,/)\,5 + 7\,2 - * = (3\,5\,+)\,7\,2 - * = 8\,(7\,2\,-) * = (8\,5\,*) = 40$

c) $(3\,2\,*)\,2\uparrow 5\,3 - 8\,4\,/ * - = (6\,2\uparrow)\,5\,3 - 8\,4\,/ * - = 36\,(5\,3\,-)\,8\,4\,/ * - = 36\ 2\,(8\,4\,/) * - = 36\,(2\,2\,*) - = (36\ 4\,-) = 32$

26. We prove this by induction on the length of the list. If the list has just one element, then the statement is trivially true. For the inductive step, consider the beginning of the list. There we find a sequence of vertices, starting with the root and ending with the first leaf (we can recognize the first leaf as the first vertex with no children), each vertex in the sequence being the first child of its predecessor in the list. Now remove this leaf, and decrease the child count of its parent by 1. The result is the preorder and child counts of a tree with one fewer vertex. By the inductive hypothesis we can uniquely determine this smaller tree. Then we can uniquely determine where the deleted vertex goes, since it is the first child of its parent (whom we know).

28. It is routine to see that the list is in alphabetical order in each case. In the first tree, vertex b has two children, whereas in the second, vertex b has three children, so the statement in Exercise 26 is not contradicted.

30. **a)** This is not well-formed by the result in Exercise 31.

b) This is not well-formed by the result in Exercise 31.

c) This is not well-formed by the result in Exercise 31.

d) This is well-formed. Each of the two subexpressions $\circ xx$ is well-formed. Therefore the subexpression $+\circ xx\circ xx$ is well-formed; call it A. Thus the entire expression is $\times Ax$, so it is well-formed.

32. The definition is word-for-word the same as that given for prefix expressions, except that "postfix" is substituted for "prefix" throughout, and $*XY$ is replaced by $XY*$.

34. We replace the inductive step (ii) in the definition with the statement that if $X_1, X_2, \ldots, X_n$ are well-formed formulae and $*$ is an n-ary operator, then $*X_1X_2\ldots X_n$ is a well-formed formula.