1.

```
void inorder(TREE T)
{
    if (T != NULL) {
        inorder(T->leftchild);
        printf("%c\n", T->label);
        inorder(T->rightchild);
    }
}
```

2.

```
BOOLEAN isoperator(char sym) {
    if (sym == '+' || sym == '-' || sym == '*' || sym == '/')
        return TRUE;
    else return FALSE;
}

void inorder(TREE T)
{
    if (T != NULL) {
        if (isoperator(T->label)) {
            putchar('(');
            inorder(T->leftchild);
            printf("%c", T->label);
            inorder(T->rightchild);
            putchar(')');
        }
        else {
            inorder(T->leftchild);
            printf("%c", T->label);
            inorder(T->rightchild);
        }
    }
}
```

3.

```
BOOLEAN isoperator(char sym) {
    if (sym == '+' || sym == '-' || sym == '*' || sym == '/')
        return TRUE;
    else return FALSE;
}

BOOLEAN need_parentheses(char sym) {
    if (sym == '*' || sym == '/')
        return TRUE;
    else return FALSE;
```

```
}

void inorder(TREE T)
{
    if (T != NULL) {
        if (need_parentheses(T->label)) {
            if (T->leftchild != NULL && isoperator(T->leftchild->label)) {
                putchar('(');
                inorder(T->leftchild);
                putchar(')');
            }
            else inorder(T->leftchild);

            printf("%c", T->label);

            if (T->rightchild != NULL && isoperator(T->rightchild->label)) {
                putchar('(');
                inorder(T->rightchild);
                putchar(')');
            }
            else inorder(T->rightchild);
        }
        else {
            inorder(T->leftchild);
            printf("%c", T->label);
            inorder(T->rightchild);
        }
    }
}
```

4.

```
int max(int a, int b) {
    if (a >= b) return a;
    else return b;
}

BOOLEAN isleaf(TREE T) {
    if (T->leftchild == NULL && T->rightchild == NULL)
        return TRUE;
    else return FALSE;
}

int height(TREE T) {
    if (T == NULL) return 0;
    else if (isleaf(T)) return 0;
    else return 1 + max(height(T->leftchild), height(T->rightchild));
}
```

5. We prove the following statement by structural induction.

STATEMENT $S(T)$: The number of full nodes in T is 1 fewer than the number of leaves.

BASIS. Where T is a single node n, n is a leaf, and there are zero full nodes.

INDUCTION. Let n be the root of T. Assume the inductive hypothesis holds for the children of n. Let $fn(c)$ be the number of full nodes in the binary tree rooted at c. Let $leaves(c)$ be the number of leaves in the binary tree rooted at c.

Suppose n is a full node. Then by the inductive hypothesis we must have

$$\begin{aligned}1 + fn(c_1) + fn(c_2) &= 1 + (leaves(c_1) - 1) + (leaves(c_2) - 1)\\ &= leaves(c_1) + leaves(c_2) - 1\end{aligned}$$

full nodes in T, which is 1 fewer than the number of leaves.

Suppose n is not a full node. Then n must have one and only one child. Therefore by the inductive hypothesis we must have

$$fn(c_1) = leaves(c_1) - 1$$

full nodes in T, which gives the same result. This proves the inductive step. ✦

6. We shall prove the following statement by structural induction.

STATEMENT $S(T)$: The number of `NULL` pointers in T is 1 greater than the number of nodes.

BASIS. Where T is a single node n, then n has 2 `NULL` pointers.

INDUCTION. Let n be the root of T having at least one child. Let $np(c)$ be the number of `NULL` pointers in the binary tree rooted at c. Let $nodes(c)$ be the number of nodes in the binary tree rooted at c.

Consider the case where n has exactly one child c_1. Since c_1 is the root of a binary tree, then the inductive hypothesis holds for c_1. We know that there are $1 + nodes(c_1)$ nodes in T. Thus by the inductive hypothesis we must have

$$1 + nodes(c_1) =$$