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));
}
```