数据结构 hw6

刘良宇 PB20000180

5.34

这里采用拓展链表的广义表实现方式。

```
int ReverseGList(GList& L) {
    // 考虑先将表当前层所有元素逆置。
    // 如果过程中遇到结点为广义表，则递归执行。
    GList* head = L->hp, *pre = nullptr;
    while (head) {
        // 如果是广义表，递归
        if (head->tag == 1) {
            ReverseGList(*head);
        }
        // 链表逆置
        GList* temp = head->tp;
        head->tp = pre;
        pre = head;
        head = temp;
    }
    // 此时 pre 成为了表的第一个元素
    L->hp = pre;
    return 1;
}
```

6.36

```
int BiTreeSim(BiTree T1, BiTree T2) {
    if (!T1 && !T2) {
        return 1;
    }
    if (!T1 || !T2) {
        return 0;
    }
    // 此时都非空
    return BiTreeSim(T1->lchild, T2->lchild) &&
        BiTreeSim(T2->rchild, T2->rchild);
}
```

6.43

```
int ReverseBiTree(BiTree T) {
    if (!T) {
        return 1;
    }
    BiTree temp = T->lchild;
    T->lchild = T->rchild;
    T->rchild = temp;
    return ReverseBiTree(T->lchild) &&
        ReverseBiTree(T->rchild);
}
```

6.48

```
// 返回的结点写入到 e 中
// 该函数返回 1 表示 p 在 root 树中，返回 2 表示 q 在 root 树中，返回 3 表示都在
int CommonParent(BiTree& e, BiTree root, BiTree p, BiTree q) {
    if (!T)  {
        return 0;
    }
    int l = CommonParent(e, root->lchild, p, q),
        r = CommonParent(e, root->rchild, p, q);
    if (l == 1 && r == 2 || l == 2 && r == 1) {      // 如果 p, q 分别在左右
        e = root;
    }
    int now = 0;
    if (root == p) {     // 判断 q 是不是 p 子孙
        if (l == 2 || r == 2) {
            e = root;
        }
        now = 1;
    }
    if (root == q) {     // 判断 p 是不是 q 子孙
        if (l == 1 || r == 1) {
            e = root;
        }
        now = 2;
    }
    return now + l + r;
}
```