第一章 线性表

1. **编写一个将给定的线性链表逆转的 C 函数，只允许改变结点的指针值，不允许移动结点值。给出调用方法。**

```
/*结点结构
typedef struct node
{
    int data; //存储数据
    struct node* next; //指向下一个节点的指针
} NODE;
*/
Node * reverse_LinkList (NODE    * head)
{
//单向链表，请说明自己用的单向链表是否带表头结点，必须考虑链表为空的情况

}
```

2. **对于给定的线性链表，编写一个把值为 a 的结点插在值为 b 的结点的前面的 C 函数。若值为 b 的结点不在线性链表中，则把 a 插在链表的最后。若有多个值为 b 的结点，则擦在第一个出现的 b 结点之前。**

```
//分两种情况：带表头结点和不带表头结点，并分别给出调用方法。考虑链表为空的情况。
/*
typedef struct node
{
    int data; //存储数据
    struct node* next; //指向下一个节点的指针
} NODE;
*/
```

3. **假设 A 和 B 是两个按结点值从小到大排列的有序环形链表（带表头结点，表头结点中不存放有效信息）。试编写一个将这两个有序的环形链表归并为一个按节点值从小到大排列的有序环形链表的 C 函数。**

```
NODE    * MergeLinkList（NODE* A，NODE * B）
{
    //要求返回新链表的 head 指针，不为新链表申请新的存储空间
    //要考虑链表为空的情况。注意如何判断链表已访问到末尾
}
```

4. 求广义表的深度。我们规定空的广义表的深度为 0，而一般地有

$$\text{depth(s)} = \begin{cases} 0, & \text{若s是一个原子} \\ 1 + max\{depth(\alpha_0), \cdots, depth(\alpha_{n-1})\}, & \text{若s是广义表}(\alpha_0, \alpha_1, \cdots, \alpha_{n-1}) \end{cases}$$

编写一个求解广义表 s 的深度的 C 函数。

```
/*  结点结构

    struct node
    {
        int tag;   // 当 tag==0 时，第二个字段为 data，当 tag==1 时，第二个字段为 dlink
        union
        {
            struct node * dlink;
            char data;
        } dd;
        struct node * link ;
    };
    typedef struct node NODE;
*/
```

思考：1）如果是一个递归表

广义表 C=(a,C) //a 为原子，C 为一个广义表，你的函数会输出什么结果？

2）在写递归程序时，要特别注意哪些情况？

5. 利用一个栈进行火车车厢调度，每节车厢有不同的编号，并将被单独拖进站，可以在任何时候出站。若有 n 节车厢依次进站，则可能的出站情况有多少种（请写出递推公式）？