2. 节点之和

编写C/C++程序，创建一个n个节点的单向链表，每个节点中都含有int型数据成员data，然后用递归调用的函数实现该单向链表的n个data之和sum的计算（假设非空链表的sum非0）。要求分别用函数实现链表的创建（含节点数据的输入）、输出和删除，并在 main 函数中输入n、输出sum。

输入格式

- 第一行：一个非负整数（代表n）
- 第二行：n个正整数（存储在单向链表中）

输出格式

- 一个整数（代表sum）

解法：链表的创建、遍历

```
const int N = 10;
typedef struct Node Node;
struct Node
{
    int data;
    Node *next;
};

Node *InsCreate( );
void Output(const Node *);
int Sum(Node *);
void DeleteList(Node *);
```

```
int main()
{
    Node *list = InsCreate();
    Output(list);
    cout << Sum(list);
    DeleteList(list);

    return 0;
}
```

解法：链表的创建、遍历

```
const int N = 10;
Node *InsCreate()
{
    Node *head = NULL;
    for(int i = 0; i < N; ++i)
    {
        Node *p = new Node;
        cin >> p -> data;
        p -> next = head;
        head = p;
    }
    return head;
}
```

解法：链表的创建、遍历

```
int Sum(Node *head)
{
    if(head == NULL)
            return 0;
    else
            return head->data + Sum(head->next);
}
```

解法：链表的创建、遍历

```
void Output(const Node *head)
{      while(head != NULL)
      {            cout << head -> data << " ";
                   head = head->next;
      }
      cout << endl;
}

void DeleteList(Node *head)
{     while(head)
      {     Node *current = head;
            head = head -> next;
            free(current);
      }
}
```

3. 老鹰抓小鸡

请你设计C/C++程序：输入一组整数，将其存储到链表中，代表一窝鸡的重量；其中有且只有一个最大值，代表老母鸡的重量，找到老母鸡对应的节点；将老母鸡对应的节点及其后的子链表整体平移拼接到原链表的开头；输出拼接后的链表，代表老鹰抓小鸡游戏的鸡方准备就绪。

输入格式

- 一行，n+1个整数（n 是链表的长度，输入 -1 表示结束）

输出格式

- 一行，拼接处理后的新链表（代表准备就绪的 n 只鸡的重量，用空格分隔）

200 100 300 200 **1700** 150 -1

1700 150 200 100 300 200

```
list = Concat(list, FindMaxNode(list));
Output(list);
```

解法：链表的创建、遍历、拼接

```
Node *Concat(Node *head, Node *max)
{
    Node *newhead = head, *newtail = head;
    while (newhead != max)
    {
        newtail = newhead;
        newhead = newhead -> next;
    }
    newtail -> next = NULL;    //第一个节点最大时，死循环
    while (newhead->next)
    {
        newhead = newhead->next;
    }
    newhead->next = head;
    return max;
}
```

解法：链表的创建、遍历、拼接

```
Node *Concat(Node *head, Node *max)
{
    Node *newhead = head, *newtail = NULL;
    while (newhead != max)
    {
        newtail = newhead;
        newhead = newhead -> next;
    }
    if (newtail) newtail -> next = NULL;
    else return head;
    while (newhead->next)
    {
        newhead = newhead->next;
    }
    newhead->next = head;
    return max;
}
```

4. 单链表分裂

请你设计C/C++程序将一个单向链表 L 分裂成两个单向链表 L1 和 L2 （保持原来的顺序）。假定 L 中有且只有两个重要节点，要求分裂后的 L1 和 L2 中各含一个重要节点，其中 L1 中的最后一个节点是重要节点。

输入格式

- 一行：若干个非负整数（存储在单向链表 L 中，其中有且只有两个0，代表重要节点），最后输入一个 -1

输出格式

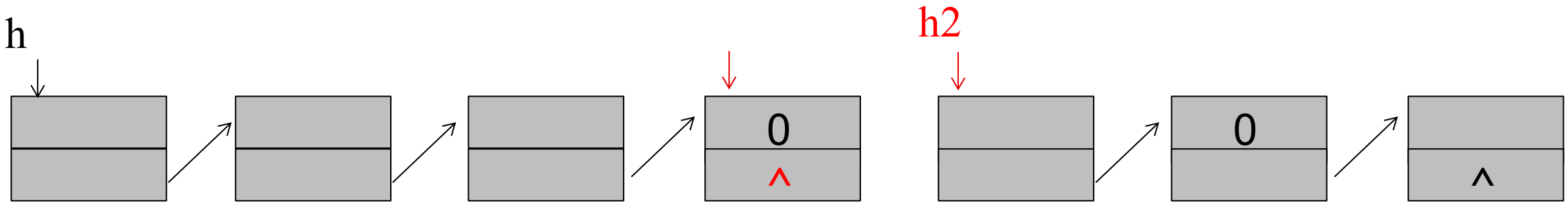
- 第一行：若干个正整数和一个0
  （存储在单向链表 L1 中）
- 第二行：若干个非负整数
  （其中有且只有一个0，存储在单向链表 L2 中）

```
1 2 3 0 5 0 7 -1


1 2 3 0
5 0 7
```

解法：链表的创建、遍历

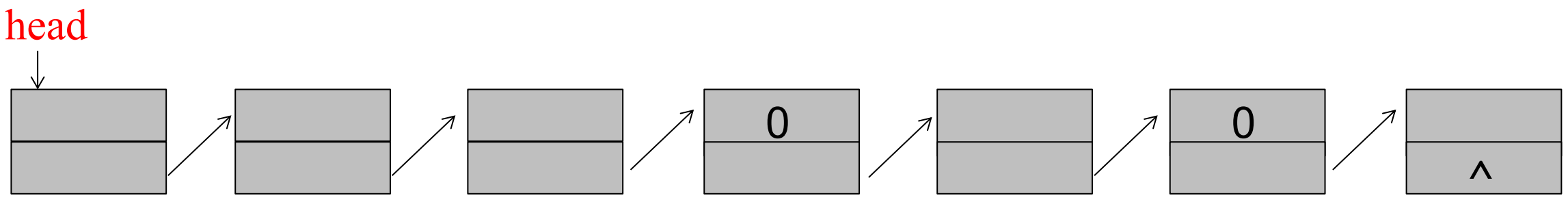
```
int main()
{
    Node *h = AppCreate();
    //PrintList(h);
    Node *h2 = Find0(h);
    PrintList(h);
    PrintList(h2);
    return 0;
}
```



解法：链表的创建、遍历

```
Node *Find0(Node *head)
{
    while (head -> data)
    {
        head = head -> next;
    }
    Node *head2 = head -> next;
    head -> next = NULL;
    return head2;
}
```

```
int main()
{
    Node *h = AppCreate();
    //PrintList(h);
    Node *h2 = Find0(h);
    PrintList(h);
    PrintList(h2);
    return 0;
}
```



解法：链表的创建、遍历

```
Node *Find0(Node *head)
{
    while (head -> data)
    {
        head = head -> next;
    }
    Node *head2 = head -> next;
    head -> next = NULL;
    return head2;
}
```

```
int main()
{
    Node *h = AppCreate();
    //PrintList(h);
    Node *h2 = Find0(h);
    PrintList(h);
    PrintList(h2);
    return 0;
}
```