第二章作业答案

```
#include <stdlib.h>
#include <stdio.h>
#define ElemType int //类型定义

enum Status//状态类型定义
{
    OK = 1, ERROR = 0, OVERFLOW = -2
};

#define LIST_INIT_SIZE 100 // 线性表存储空间的初始分配量
#define LISTINCREMENT 10 // 线性表存储空间的分配增量

typedef struct
{
    ElemType *elem; // 首地址指针
    int length; // 长度
    int listsize; // 当前分配的存储容量
}SqList;

int problem2_12(SqList A, SqList B)//当ElemType中定义减法运算时
A<B返回负值 A=B 返回0 A>B返回正值
{
    for (int i = 0; i<A.length&&i<B.length; i++)
        if (A.elem[i] != B.elem[i])
            return A.elem[i] < B.elem[i]?-1:1; //当ElemType中只定
义比较运算时可以这样用
            //return A.elem[i]-B.elem[i]; //当ElemType中定义减法运
算时可以这样用
    return A.length - B.length;
}

Status problem2_21_a(SqList &A)
{
    int *q = A.elem;
    int *p = A.elem + A.length - 1;
    ElemType t;
    while (q<p)//分别从头尾开始向中间遍历
    {
        t = *q;//遍历的同时交换对应变量顺序
        *q = *p;
        *p = t;
```

```
        q++;
        p--;
    }
    return OK;
}

Status problem2_21_b(SqList *A)
{
    int *q = A->elem;
    int *p = A->elem + A->length - 1;
    ElemType t;
    while (q<p)//分别从头尾开始向中间遍历
    {
        t = *q;//遍历的同时交换对应变量顺序
        *q = *p;
        *p = t;
        q++;
        p--;
    }
    return OK;
}

Status problem2_25(SqList A, SqList B, SqList *C)
{
    if (C->elem != NULL) free(C->elem);//给C分配空间
    C->elem = (ElemType*)malloc(A.length * sizeof(ElemType));
    if (NULL == C->elem) exit(OVERFLOW);
    C->length = 0;
    C->listsize = A.length;
    int i = 0, j = 0;
    while (i<A.length&&j<B.length)//i,j分别遍历表A B
    {
        if (A.elem[i] == B.elem[j])//当对应元素相同时将改元素加入表
A 注意此时++号的位置只能放在对应变量之后
            C->elem[C->length++] = A.elem[i++];
        else if (A.elem[i]<B.elem[j]) //当对应元素不同时 利用数组元
素的有序性 将对应元素较小的那个向后遍历
            i++;
        else
            j++;
    }
    return OK;
}
```

```
Status problem2_29(SqList *A, SqList B, SqList C)//O(n^2)时间复杂度版本
{
    int a = 0, b = 0, c = 0;
    while (a<A->length&&b<B.length&&c<C.length)
    {
        while (B.elem[b]<A->elem[a])b++;//判断B中对应元素是否与A中对应元素相等 不等利用元素有序性 遍历B中对应元素
        while (C.elem[c]<A->elem[a])c++;//判断C中对应元素是否与A中对应元素相等 不等利用元素有序性 遍历C中对应元素
        if (B.elem[b] == A->elem[a] && C.elem[c] == A->elem[a])//A中同时存在BC中元素 则删除
        {
            ElemType *q = A->elem + a;//顺序表删除操作 时间复杂度O(n)
            ElemType *p = A->elem + A->length - 1;
            for (q++; q <= p; q++)
                *(q - 1) = *q;
            A->length--;
            //注意 因为删除操作之后整表前移 所以不需要a++
        }
        else //不存在则继续判断下一个元素
            a++;
    }
    return OK;
    //值得一提的是 由于线性表删除操作时间复杂度为 O(n) 所以该方法整体时间复杂度O(n^2)
    //但是可以通过一些方法完成一次遍历中执行全部删除操作
    //这样将使算法只执行一次遍历+一次删除 将整体时间复杂度降低到O(n)
    //有兴趣的同学可以自行思考
}

//以下为函数实际测试部分
Status ListInit(SqList *L)//顺序表初始化
{
    L->elem = (int *)malloc(LIST_INIT_SIZE * sizeof(int));
    if (NULL == L->elem) exit(OVERFLOW);
    L->length = 0;
    L->listsize = LIST_INIT_SIZE;
    return OK;
}

Status ListAdd(SqList *L, int e)//顺序表尾部插入
{
```

```
    if (L->length> = L->listsize)
    {
        int *newbase = (int *)realloc(L->elem, (L->listsize +
LISTINCREMENT) * sizeof(int));
        if (!newbase) exit(OVERFLOW);
        L->elem = newbase;
        L->listsize += LISTINCREMENT;
    }
    L->elem[L->length] = e;
    L->length++;
    return OK;
}

void ListTraversal(SqList L)//顺序表遍历显示
{
    for (int i = 0; i<L.length; i++)
        printf("%d ", L.elem[i]);
    printf("\n");
}

int main()
{
    //表赋值与定义
    SqList A, B, C, P;
    ListInit(&A);
    for (int i = 1; i <= 10; i++)
        ListAdd(&A, i);
    ListInit(&B);
    for (int i = 2; i <= 6; i++)
        ListAdd(&B, i);
    ListInit(&C);
    for (int i = 3; i <= 7; i++)
        ListAdd(&C, i);
    ListTraversal(A);
    ListTraversal(B);
    ListTraversal(C);

    //2.12
    printf("%d\n", problem2_12(B, C));

    //2.21
    problem2_21_a(A);
    ListTraversal(A);
    problem2_21_b(&A);
```

```
    ListTraversal(A);

    //2.25
    problem2_25(B, C, &P);
    ListTraversal(P);

    //2.29
    problem2_29(&A, B, C);
    ListTraversal(A);

}

//2.33
//循环链表也可以用LinkList，最好是自己再定义一个别的加以区分，例如CList
typedef struct CLNode{
    char data;
    struct CLNode *next;
}CLNode, *CList;
Status problem2_33(LinkList &L, CList& Letter, CList &Digit, CList &Other)
{//L就是题目给的线性链表
    Letter = (CList)malloc(sizeof(CLNode));
    Digit = (CList)malloc(sizeof(CLNode));
    Other = (CList)malloc(sizeof(CLNode));
    la = Letter;
    lb = Digit;
    lc = Other;
    p = L->next;
    while (p){
        //很多同学没有如何写判断字母字符和数字字符的这部分
        if (p->data >= 'a' &&p->data <= 'z' || p->data >= 'A' &&p->data <= 'Z'){
            la->next = p;
            la = p;
        }else if (p->data >= '0' &&p->data <= '9'){
            lb->next = p;
            lb = p;
        }else{
            lc->next = p;
            lc = p;
        }
        p = p->next;
    }
    //把三个链表最后一个结点的next域指向头结点
    la->next = Letter;
    lb->next = Digit;
```

```
    lc->next = Other;
    return OK;
}
//这题主要扣分点在于没有处理好循环链表，以及判断字母数字字符的函数没有写出来

//2.38
DuLNode *problem2_38(DuLinkedList &L, ElemType x){
    p = L->next;
    while (p->data != x&& p != L)
        p = p->next;
    if (p == L)
        return NULL; //没找到，有部分同学没有这样的判断
    p->freq++;
    q = p->pre;
    while (q->freq <= p->freq && q != L)
        q = q->pre; //查找插入位置
    if (q != p->pre) {//如果q==p->pre，表明p的位置不需要移动
        p->pre->next = p->next;
        p->next->pre = p->pre;
        q->next->pre = p;
        p->next = q->next;
        q->next = p;
        p->pre = q; //调整位置
    }
    return p;
}
出现的问题：
//1. 插入位置找的不对，没考虑到需要移动该结点的判断条件
//2. 双向循环链表里面交换两个结点或者是把一个结点挪到链表中的某个地方需要动六个指针
//3. 双向循环链表最后一个结点它的next指向的是头结点L，而不是L->next，L->pre指向最后一个结点
//PS：有少数同学在处理移动时直接把结点的data域交换...最好掌握一下链表的指针挪动
```