```cpp
#include<stdio.h>
#include<stdlib.h>

#define LIST_INIT_SIZE 100  //存储空间初始分配量
#define LIST_INCREMENT 10   //存储空间分配增量

typedef int ElemType;
typedef struct
{
    ElemType *base;     //存储空间基址
    int length;         //当前表长
    int listsize;       //当前分配的存储容量
} SqList;

enum  Status
{
    OVERFLOW = -1,
    ERROR = 0,
    OK = 1
};

//函数声明
Status InitList_Sq(SqList& l);
Status ListInsert_Sq(SqList& l, int i, ElemType e);
Status ListDelete_Sq(SqList& l, int i, ElemType& e);
Status GetElem(SqList& la, int i, int& ai);
Status ListSort(SqList& l);
void MergeList(SqList& la, SqList& lb, SqList& lc);
Status PrintList(SqList& l);
void PrintStar(void);

//主函数
int main()
{
    char chos;
    ElemType elem;
    int pos;

    SqList la;
    SqList lb;

    SqList lc;  //a表和b表的合并

    InitList_Sq(la);
    InitList_Sq(lb);
    lc.base = NULL;

    while(1)
    {
        printf("1.Insert a new elem to la.\n");
        printf("2.Insert a new elem to lb.\n");

        printf("3.Delete ist elem in la.\n");
        printf("4.Delete ist elem in lb.\n");

        printf("5.Sort la.\n");
        printf("6.Sort lb.\n");

        printf("7.Merge la & lb to lc.\n");

        printf("8.Print list.\n");

        printf("9.Quit.\n\n");

        printf("Please choose what you want to do: ");

        scanf_s("%c", &chos);
        if(chos == 10)
        {
            scanf_s("%c", &chos);
        }

        switch(chos)
        {
            case '1':
                {
                    printf("Please input the position & value of the elem: ");
                    scanf_s("%d,%d", &pos, &elem);

                    ListInsert_Sq(la, pos, elem);
                }break;
            case '2':
                {
                    printf("Please input the position & value of the elem: ");
                    scanf_s("%d,%d", &pos, &elem);

                    ListInsert_Sq(lb, pos, elem);
                }break;
            case '3':
                {
                    printf("Please input the position of the elem: ");
                    scanf_s("%d", &pos);

                    if(ListDelete_Sq(la, pos, elem))
                    {
                        printf("elem %d in la = %d\n", pos, elem);
                    }
                }break;
            case '4':
                {
                    printf("Please input the position of the elem: ");
                    scanf_s("%d", &pos);

                    if(ListDelete_Sq(lb, pos, elem))
                    {
                        printf("elem %d in lb = %d\n", pos, elem);
                    }
                }break;
            case '5':
                {
                    ListSort(la);
                    printf("la has been sorted.\n");
                }break;
            case '6':
                {
                    ListSort(lb);
                    printf("lb has been sorted.\n");
                }break;
            case '7':
                {
                    MergeList(la, lb, lc);
                }break;
            case '8':
                {
                    printf("Please choose which list you want to printf: ");
                    scanf_s("%c", &chos);
                    if(chos == 10)
                    {
                        scanf_s("%c", &chos);
                    }

                    switch(chos)
                    {
                        case 'a':
                            {
                                PrintStar();
                                printf("la: ");
                                PrintList(la);
                                PrintStar();
                            }break;
                        case 'b':
                            {
                                PrintStar();
                                printf("lb: ");
                                PrintList(lb);
                                PrintStar();
                            }break;
                        case 'c':
                            {
                                PrintStar();
                                printf("lc: ");
                                PrintList(lc);
                                PrintStar();
                            }break;
                        default :
                            {
                                printf("Choose2 error!\n");
                            }break;
                    }
                }break;
            case '9':
                {
                    free(la.base);
                    free(lb.base);
                    if(!lc.base)
                    {
                        free(lc.base);
                    }
                    exit(0);
                }break;
            default :
                {
                    printf("Choose1 error!\n");
                }break;
        }

        //system("pause");
        putchar(10);
    }

    return 0;
}

//函数定义
Status InitList_Sq(SqList& l)
{
    l.base = (ElemType *) malloc ((LIST_INIT_SIZE) * sizeof(ElemType));
    if(!l.base) exit(OVERFLOW);

    l.length = 0;
    l.listsize = LIST_INIT_SIZE;

    return OK;
} //InitList_Sq;

Status ListInsert_Sq(SqList& l, int i, ElemType e)
```

```cpp
{
    if(i<1 || i>l.length+1)
    {
        printf("Pos Error!\n");
        return ERROR;
    }

    ElemType* p;

    if(l.length >= l.listsize)
    {
        ElemType* newbase;

        newbase = (ElemType *)
                  realloc (l.base, (l.listsize + LIST_INCREMENT) * sizeof
                    (ElemType));
        if(!newbase) exit(OVERFLOW);
        l.base = newbase;
        l.listsize += LIST_INCREMENT;
    }

    if(i < l.length + 1)    //不是在表尾插入元素
    {
        ElemType* q;
        q = &l.base[i - 1];

        for(p = &l.base[l.length - 1]; p>=q; --p)
        {
            *(p+1) = *p;
        }
        *q = e;
    }
    else
    {
        l.base[i - 1] = e;
    }

    ++l.length;

    return OK;
} //ListInsert_Sq;

Status ListDelete_Sq(SqList& l, int i, ElemType& e)
{
    if((i < 1) || (i > l.length))
    {
        printf("Pos Error!\n");
        return ERROR;
    }

    ElemType* p;
    ElemType* q;

    p = &l.base[i - 1];
    e = *p;
    q = l.base + l.length -1;

    for(++p; p <= q; ++p)
    {
        *(p-1) = *p;
    }

    --l.length;

    return OK;
} //ListDelete_Sq;

Status GetElem(SqList& la, int i, int& ai)
{
    if(i<1 || i>la.length)
    {
        return ERROR;
    }

    ai = la.base[i - 1];

    return OK;
} //GetElem;

//对int表进行冒泡排序
Status ListSort(SqList& l)
{
    if(l.base == NULL)
    {
        printf("There is no this list!\n");
        return ERROR;
    }
    if(l.length == 0)
    {
        printf("There is no elem in this list!\n");
        return OK;
    }

    ElemType temp;
    for(int i = 0; i < l.length - 1; ++i)
    {
        for(int j = 0; j < (l.length - 1) - i; ++j)
        {
                if (l.base[j] > l.base[j+1])
                {
                    temp = l.base[j];
                    l.base[j] = l.base[j+1];
                    l.base[j+1] = temp;
                }
            }
    }
    return OK;
} //ListSort;

//合并线性表la与lb（递增顺序,无重复元素）
void MergeList(SqList& la, SqList& lb, SqList& lc)
{
    printf("la & lb will be sorted first.waiting…\n");
    ListSort(la);
    ListSort(lb);

    int i = 1;
    int j = 1;
    ElemType ai;
    ElemType bj;

    if(lc.base)
    {
        free(lc.base);
    }
    InitList_Sq(lc);

    while((i <= la.length) && (j <= lb.length))
    {
        GetElem(la, i, ai);
        GetElem(lb, j, bj);

        if(ai <= bj)
        {
            ListInsert_Sq(lc, lc.length + 1, ai);

            if(ai == bj)
            {
                ++j;
            }

            ++i;
        }
        else
        {
            ListInsert_Sq(lc, lc.length + 1, bj);

```

```
                ++j;
            }
        }

        while(i <= la.length)
        {
            GetElem(la, i++, ai);
            ListInsert_Sq(lc, lc.length + 1, ai);
        }
        while(j <= lb.length)
        {
            GetElem(lb, j++, bj);
            ListInsert_Sq(lc, lc.length + 1, bj);
        }

        PrintStar();
        printf("la: ");
        PrintList(la);
        printf("lb: ");
        PrintList(lb);
        printf("lc: ");
        PrintList(lc);
        PrintStar();
    }   //MergeList;

    Status PrintList(SqList& l)
    {
        if(l.base == NULL)
        {
            printf("The list haven't exist yet!\n");
            return ERROR;
        }
        if(l.length == 0)
        {
            printf("There is no elem in this list!\n");
            return OK;
        }

        for (ElemType *p = l.base; p <= &l.base[l.length - 1]; ++p)
        {
            printf("%d ", *p);
        }

        putchar(10);

        //printf("PrintList has completed.\n");
        return OK;
    } //PrintList;
```

```
void PrintStar(void)
{
    printf("******************************************************\n");
} //PrintStar
```