```
typedef int Position;
typedef struct LNode *List;
struct LNode {
    ElementType Data[MAXSIZE];
    Position Last;
};

/* 初始化 */
List MakeEmpty()
{
    List L;

    L = (List)malloc(sizeof(struct LNode));
    L->Last = -1;

    return L;
}

/* 查找 */
#define ERROR -1

Position Find( List L, ElementType X )
{
    Position i = 0;

    while( i <= L->Last && L->Data[i]!= X )
        i++;
    if ( i > L->Last )  return ERROR; /* 如果没找到，返回错误信息 */
    else  return i;  /* 找到后返回的是存储位置 */
}

/* 插入 */
/*注意:在插入位置参数P上与课程视频有所不同，课程视频中i是序列位序（从1开始），这里P是存储下标位置（从0开始），两者差1*/
bool Insert( List L, ElementType X, Position P )
{ /* 在L的指定位置P前插入一个新元素X */
    Position i;

    if ( L->Last == MAXSIZE-1) {
        /* 表空间已满，不能插入 */
        printf("表满");
        return false;
    }
    if ( P<0 || P>L->Last+1 ) { /* 检查插入位置的合法性 */
        printf("位置不合法");
        return false;
    }
    for( i=L->Last; i>=P; i-- )
        L->Data[i+1] = L->Data[i]; /* 将位置P及以后的元素顺序向后移动 */
    L->Data[P] = X;  /* 新元素插入 */
    L->Last++;       /* Last仍指向最后元素 */
    return true;
}

/* 删除 */
/*注意:在删除位置参数P上与课程视频有所不同，课程视频中i是序列位序（从1开始），这里P是存储下标位置（从0开始），两者差1*/
bool Delete( List L, Position P )
{ /* 从L中删除指定位置P的元素 */
    Position i;

    if( P<0 || P>L->Last ) { /* 检查空表及删除位置的合法性 */
        printf("位置%d不存在元素", P );
        return false;
    }
    for( i=P+1; i<=L->Last; i++ )
        L->Data[i-1] = L->Data[i]; /* 将位置P+1及以后的元素顺序向前移动 */
    L->Last--; /* Last仍指向最后元素 */
    return true;
}
```