第一题：

（code）：

HW6.1

```c
#include <stdio.h>
void reverse(int a[], int l, int r){
    int temp;
    while (r > l)
    {
        temp = a[l];
        a[l] = a[r];
        a[r] = temp;
        r--;
        l++;
    }
}

void move(int a[], int m, int n){ // m为右移的次数，n为数组元素的个数
    if (n % m == 0)
    {
        return;
    }
    else
    {
        m = m % n;
    }
    reverse(a, 0, n-m-1); // 将数组的第0位到第n-m-1位进行转置
    reverse(a, n-m, n-1); // 将数组的第n-m位到最后一位进行转置
    reverse(a, 0, n-1);   // 将这个数组进行转置
}

int main(){
    int array[5] = {1,2,3,4,5};
    int m; // 需要右移的次数
    printf("右移前的数组为");
    for(int i = 0; i < 5; i++){
        printf("%d",array[i]);
    }
    printf("\n");
    printf("请输入需要向右移动多少位\n");
    scanf("%d",&m);
    move(array, m, 5);
    printf("右移后的数组为\n");
        for(int i = 0; i < 5; i++){
        printf("%d",array[i]);
    }
    return 0;
}
```

核心思想：

1. 输入右移的次数。
2. 对右移的次数进行取模（假如共有 5 个元素，向右 6 移动 6 次就等于向右移动 1 次）。
3. 设右移的次数为 m，则先将前 n - m 个元素转置。假如我共有五个元素（1，2，3，4，5），右移两位，则该步骤将转换为（3，2，1，4，5）。
4. 再将最后 m 位进行转置。（3，2，1，5，4）
5. 最后将整个数组进行转置。（4，5，1，2，3）

第二题：

（Code）：

HW6.2

```c
// 完成用十字链表存储的稀疏矩阵的加法运算。
typedef struct OLNode
{
    int i;
    int j;
    int e;
    struct OLNode *right;
    struct OLNode *down;
} OLNode, *OLink;

typedef struct
{
    OLink *rhead;
    OLink *chead;
    int mu; // 行数
    int nu; // 列数
    int tu;
} CrossList;

void addArray(CrossList &a, CrossList b)
{
    int cnt = a.tu + b.tu; // 统计一共有多少个非零数
    int i = 1;
    // int j1, j2;
    while (cnt > 0) // 当还有数没加时继续循环
    {
        OLink p = a.rhead[i];
        OLink q = b.rhead[i];

        if (!p && q) // p当前行无元素，q有元素，则将q赋给p
            p = q;
        if (!p && !q) // p、q当前行均无元素，则跳转下一行
            continue;
        if (p && !q) // p有元素，q无元素，则跳转下一行
            continue;
        if (p && q) // p、q均有元素，则进行加法
        {
            do
            {
                if (p->j < q->j)
                {
                    cnt--;        // cnt减一
                    if (p->right) // p的右元素不为空
                    {
                        p = p->right; // 工作指针p向右移
                    }
                    else
                    {
                        p->right = q; // q当前行的所有元素接到p的后面
                    }
                }
                else if (p->j == q->j)
                {
                    p->e = p->e + q->e;
                    p = p->right;
                    q = q->right;  // p、q同时向右移动
                    cnt = cnt - 2; // cnt减二
                }
                else
                {
                    OLink tmp1 = p;
                    p = q;                  // a.rhead[i]指向q的节点
                    OLink tmp2 = q->right; // 将q的右节点的指针保存
                    q->right = tmp1;       // q的右指针指向p
                    cnt--;
                    if (tmp2) // 若q的右节点不为空
                    {
                        q = tmp2; // 工作指针q向右移
                    }
                }
            } while (!p->right); // 当工作指针为当前行的最后一个元素时退出循环
        }

        i++; // 转移到下一行
    }
}
```

核心思想：

1. 获取两个稀疏矩阵总有多少个非零元素，记作 cnt。

2. 当 cnt 不为零时一直循环，每循环一次 i++，也就是行循环，每循环一次就转移至下一行。

3. 先从第一行开始循环，使得两个工作指针 p、q 分别指向两个稀疏矩阵的第一行第一个非零元。对当前行 p、q 有无元素个数进行判断：A（p 当前行无元素并且 q 有元素，则将赋值给 p，也就是 q 当前行的元素全部都接在了 p 之后）；B（p、q 当前行均无元素，则跳出当前循环，令 i+1，进入下一行也就是下一个循环）;C（p 当前行有元素、q 无元素，则跳出当前循环，转移至下行）；D（p、q 当前行均有元素则进行下述循环）。

```
do
{
    if (p->j < q->j)
    {
        cnt--;          // cnt减一
        if (p->right) // p的右元素不为空
        {
            p = p->right; // 工作指针p向右移
        }
        else
        {
            p->right = q; // q当前行的所有元素接到p的后面
        }
    }
    else if (p->j == q->j)
    {
        p->e = p->e + q->e;
        p = p->right;
        q = q->right;  // p、q同时向右移动
        cnt = cnt - 2; // cnt减二
    }
    else
    {
        OLink tmp1 = p;
        p = q;                  // a.rhead[i]指向q的节点
        OLink tmp2 = q->right; // 将q的右节点的指针保存
        q->right = tmp1;        // q的右指针指向p
        cnt--;
        if (tmp2) // 若q的右节点不为空
        {
            q = tmp2; // 工作指针q向右移
        }
    }
} while (!p->right); // 当工作指针为当前行的最后一个元素时退出循环
```