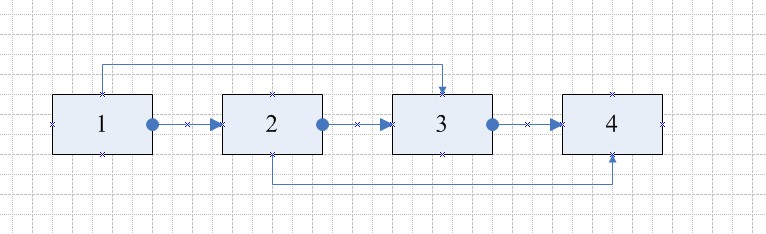
题目：有一个复杂链表，其结点除了有一个m\_pNext指针指向下一个结点外，  
还有一个m\_pSibling指向链表中的任一结点或者NULL。其结点的C++定义如下：  
 struct ComplexNode  
{  
    int m\_nValue;  
    ComplexNode\* m\_pNext;  
    ComplexNode\* m\_pSibling;  
};

下图是一个含有5个结点的该类型复杂链表。  
图中实线箭头表示m\_pNext指针，虚线箭头表示m\_pSibling指针。为简单起见，  
指向NULL的指针没有画出。



第一反应就是根据m\_pNext直接先构造链表A，然后取源链表B中的第n个节点，取m\_pSibling，然后开始从头寻找节点x，x=m\_pSibling，记录走的次数y，对于链表A，对于第n个节点u，同样A的头结点也走y步到节点m，然后将u->m\_pSibling=m;这样的话，时间复杂度是O(n2)  
后来，经过提示，想到另一种方法。

首先在原链表的每个节点后都创建一个对应的节点，初始化的时候m\_pSibling=NULL

void CloneNodes(ComplexNode\* pHead)

{

    ComplexNode\* pNode = pHead;

    while(pNode != NULL)

    {

        ComplexNode\* pCloned = new ComplexNode();

        pCloned->m\_nValue = pNode->m\_nValue;

        pCloned->m\_pNext = pNode->m\_pNext;

        pCloned->m\_pSibling = NULL;

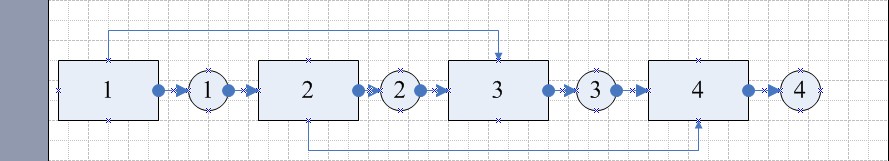
        pNode->m\_pNext = pCloned;

        pNode = pCloned->m\_pNext;

    }

}

 结果如下：

  
开始设置新插入的节点的m\_pSibling的值，A中新的节点1的m\_pSibling 等于原节点中1的m\_pSibling的m\_pNext

void ConnectSiblingNodes(ComplexNode\* pHead)

{

    ComplexNode\* pNode = pHead;

    while(pNode != NULL)

    {

        ComplexNode\* pCloned = pNode->m\_pNext;

        if(pNode->m\_pSibling != NULL)

        {

            pCloned->m\_pSibling = pNode->m\_pSibling->m\_pNext;

        }

        pNode = pCloned->m\_pNext;

    }

}

构造完新链表之后，提取链表

ComplexNode\* ReconnectNodes(ComplexNode\* pHead)

{

    ComplexNode\* pNode = pHead;

    ComplexNode\* pClonedHead = NULL;

    ComplexNode\* pClonedNode = NULL;

    if(pNode != NULL)

    {

        pClonedHead = pClonedNode = pNode->m\_pNext;

        pNode->m\_pNext = pClonedNode->m\_pNext;

        pNode = pNode->m\_pNext;

    }

    while(pNode != NULL)

    {

        pClonedNode->m\_pNext = pNode->m\_pNext;

        pClonedNode = pClonedNode->m\_pNext;

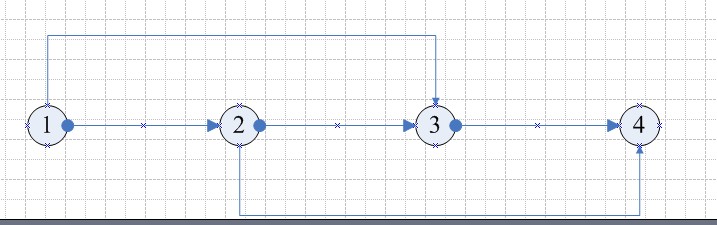
        pNode->m\_pNext = pClonedNode->m\_pNext;

        pNode = pNode->m\_pNext;

    }

    return pClonedHead;

}

  
将上面的三步连起来就可以完成 复杂链表的复制