题目：

输入一个单向链表，输出该链表中倒数第k个结点。链表的倒数第0个结点为链表的尾指针。（考虑算法的效率）

方法一：

假设整个链表有n个结点，那么倒数第k个结点是从头结点开始的第n-k-1个结点（从0开始计数）。如果我们能够得到链表中结点的个数n，那我们只要从头结点开始往后走n-k-1步就可以了。如何得到结点数n？这个不难，只需要从头开始遍历链表，每经过一个结点，计数器加一就行了。

#include <stdio.h>

#include <stdlib.h>

**typedef** struct node

**{**

int value**;**

struct node **\***next**;**

**}**LinkList**;**

void CreateList**(**LinkList **\***L**);** //建立一个链表

int GetLength**(**LinkList **\***L**);** //获取链表长度

int Output**(**LinkList **\***L**,**int len**,**int k**);** //输出长度为len的链表的倒数第k个结点的值

int main**()**

**{**

int n**,**k**;**

LinkList **\***L**=(**LinkList**\*)**malloc**(sizeof(**LinkList**));**

L**->**next**=NULL;**

CreateList**(**L**);**

n**=**GetLength**(**L**);**

**while** **(**printf**(**"Please input K: "**),**scanf**(**"%d"**,&**k**)!=**EOF**)** //输入倒数第几个结点k

**{**

**if** **(**k**>**n**-**1**)**

**{**

printf**(**"Sorry,no such Node!\n\n"**);**

**continue;**

**}**

printf**(**"%d\n\n"**,**Output**(**L**,**n**,**k**));**

**}**

**return** 0**;**

**}**

void CreateList**(**LinkList **\***L**)**

**{**

int data**;**

LinkList **\***s**=NULL,\***tail**=**L**;**

**while** **(**printf**(**"Input value: "**),**scanf**(**"%d"**,&**data**)!=**EOF**)**

**{**

s**=(**LinkList**\*)**malloc**(sizeof(**LinkList**));**

s**->**value**=**data**;**

s**->**next**=NULL;**

tail**->**next**=**s**;**

tail**=**s**;**

**}**

**}**

int GetLength**(**LinkList **\***L**)**

**{**

LinkList **\***s**=**L**->**next**;**

int count**=**0**;**

**while** **(**s**!=NULL)**

**{**

count**++;**

s**=**s**->**next**;**

**}**

**return** count**;**

**}**

int Output**(**LinkList**\***L**,**int len**,**int k**)**

**{**

LinkList **\***s**=**L**;**

int i**;**

**for** **(**i**=**0**;**i**<**len**-**k**;**i**++)**

**{**

s**=**s**->**next**;**

**}**

**return** s**->**value**;**

**}**

这种思路的时间复杂度是O(n)，但需要遍历链表两次。第一次得到链表中结点个数n，第二次得到从头结点开始的第n­-k-1个结点即倒数第k个结点。如果链表的结点数不多，这是一种很好的方法。但如果输入的链表的结点个数很多，有可能不能一次性把整个链表都从 硬盘读入物理内存，那么遍历两遍意味着一个结点需要两次从硬盘读入到物理内存。我们知道把数据从硬盘读入到内存是非常耗时间的操作。我们能不能把链表遍历 的次数减少到1？如果可以，将能有效地提高代码执行的时间效率。

方法二：

如果我们在遍历时维持两个指针，第一个指针从链表的头指针开始遍历，在第k-1步之前，第二个指针保持不动；在第k-1步开始，第二个指针也开始从链表的头指针开始遍历。由于两个指针的距离保持在k-1，当第一个（走在前面的）指针到达链表的尾结点时，第二个指针（走在后面的）指针正好是倒数第k个结点。

这种思路只需要遍历链表一次。对于很长的链表，只需要把每个结点从硬盘导入到内存一次。因此这一方法的时间效率前面的方法要高。

#include <stdio.h>

#include <stdlib.h>

**typedef** struct node

**{**

int value**;**

struct node **\***next**;**

**}**LinkList**;**

void CreateList**(**LinkList **\***L**);** //建立一个链表

void Output**(**LinkList **\***L**,**int k**);** //输出倒数第k个结点的值

int main**()**

**{**

int k**;**

LinkList **\***L**=(**LinkList**\*)**malloc**(sizeof(**LinkList**));**

L**->**next**=NULL;**

CreateList**(**L**);**

**while** **(**printf**(**"Please input K: "**),**scanf**(**"%d"**,&**k**)!=**EOF**)** //输入倒数第几个结点k

**{**

Output**(**L**,**k**);**

**}**

**return** 0**;**

**}**

void CreateList**(**LinkList **\***L**)**

**{**

int data**;**

LinkList **\***s**=NULL,\***tail**=**L**;**

**while** **(**printf**(**"Input value: "**),**scanf**(**"%d"**,&**data**)!=**EOF**)**

**{**

s**=(**LinkList**\*)**malloc**(sizeof(**LinkList**));**

s**->**value**=**data**;**

s**->**next**=NULL;**

tail**->**next**=**s**;**

tail**=**s**;**

**}**

**}**

void Output**(**LinkList**\***L**,**int k**)**

**{**

LinkList **\***tail**=**L**,\***head**=**L**;**

int i**;**

**if** **(**k**<=**0**)** //如果k小于等于0，退出

**{**

printf**(**"Input error!\n"**);**

**return;**

**}**

**for** **(**i**=**0**;**i**<**k**;**i**++)** //tail指针先走

**{**

tail**=**tail**->**next**;**

**if** **(**tail**==NULL)** //如果k输入过大，退出

**{**

printf**(**"No such node !\n"**);**

**return;**

**}**

**}**

**while(**tail**!=NULL&&**head**!=NULL)** //tail和head指针一起走

**{**

tail**=**tail**->**next**;**

head**=**head**->**next**;**

**}**

printf**(**"%d\n"**,**head**->**value**);**

**}**