**链表练习**

**1.从链表中删除素数**

以下程序中结点node定义了一个整数的信息，函数IsPrime的功能是判定一个整数是不是素数，是素数返回1否则返回0；函数create创建一个由10个整数构成的链表；函数del从链表中删除所有数据成员为素数的结点， 函数print输出头结点地址为head的链表信息。请补全程序，完成相应的功能。

#include "stdio.h"

#include "stdlib.h"

typedef struct node

{

int n;

struct node\* next;

} NODE;

int IsPrime(int n) /\*判断整数n是不是素数\*/

{

int i;

if(n<2) return 0;

for(i=2;i<n;i++)

if(n%i==0) return 0;

return 1;

}

NODE\* create() /\*创建由10个整数构成的链表\*/

{

int i;

NODE \*head=NULL,\*p;

for(i=0;i<10;i++)

{

p= (1) ;

scanf("%d",&p->n);

p->next=head;

head=p;

}

return head;

}

NODE\* del(NODE\* head) /\*删除所有素数点\*/

{

NODE \*p,\*q;

if(head==NULL) return NULL;

for(p=head,q=p->next;q!=NULL;q=p->next)

{ /\* 删除从第2个结点开始的整数部分是素数的结点\*/

if(IsPrime(q->n))

{

p->next=q->next;

free(q);

}

else

p= (2) ;

}

if(IsPrime(head->n)) /\*处理头结点整数部分是素数的情况\*/

{

p= (3) ;

free(head);

head=p;

}

return head;

}

void print(NODE\* head)

{

NODE\* p=head;

if(p==NULL) printf(" nothing");

while(p!=NULL)

{

printf("%d ",p->n);

p=p->next;

}

}

int main()

{

NODE\* head;

head=create();

printf("Source list: ");

print(head);

printf("\n");

head=del(head);

printf("After deleted: ");

print(head);

printf("\n");

return 0;

}

【样例输入】（输入10个数从头插入链表）

9 8 7 6 5 4 3 2 1 0

【样例输出】

Source list: 0 1 2 3 4 5 6 7 8 9

After deleted: 0 1 4 6 8 9

**2. 计算链表相邻结点和的最小值。**

以下程序的功能是创建有ct个结点的链表，ct的值从键盘输入，每个结点中包含一个整数信息（同样从键盘输入）。计算该链表中每两个相邻结点的整数和，求其中的最小值对应的结点序号（链表建成后结点序号从1到ct编号）。请补全程序，完成相应的功能。

#include <stdio.h>

#include <stdlib.h>

struct node

{

int n;

struct node \*next;

};

int main( )

{

struct node \*head=NULL, \*tail=NULL, \*p;

int i, min, s=0, index=1, ct;

scanf ( "%d", &ct ); /\*从键盘输入ct的值,设ct大于1\*/

for ( i=0; i<ct; i++ )

{

p = (1) ;/\*申请新结点空间\*/

scanf( "%d", &p->n );

p->next=NULL;

if ( head==NULL ) head = p;

else (2) ;

tail = p;

}

min = head->n + head->next->n;

i = 1;

for ( p=head->next; p!=NULL && p->next!=NULL; p=p->next )

{

s = (3) ;

i++;

if( (4) )

{

min=s;

index=i;

}

}

printf( "%d %d", index, index+1 );

}  
【输入形式】  
【输出形式】  
【样例输入】（输入8个数，依次从尾插入链表）

8

3 4 1 2 6 7 9 10   
【样例输出】（求出哪两个结点的和最小，此例为结点3和结点4的和最小）

3 4

**3. 链表的有序插入**

以下结点node定义了一个学生的信息。函数find查找并返回学号小于num，且下一节点学号不小于num的结点指针。函数insert按学号递增顺序插入新学生。测试主函数从键盘输入5个学号，调用insert函数插入链表，并输出链表内容。请补全程序，完成相应的功能。

#include <stdio.h>

#include <stdlib.h>

struct node

{

int num; /\*学生的学号\*/

struct node \*next; /\*指向下一结点\*/

};

struct node \*find(struct node \*head,int num)

{/\*①如果head为空链表，则返回空指针。②如果第一个结点的学号不小于num，也返回空指针。③如果最后一个结点的学号仍小于num，则返回最后一个结点指针。④如果当前结点的学号小于num且下一结点学号不小于num，则返回当前结点的指针\*/

struct node \*p, \*q;

if(head==NULL || head->num>=num)

return NULL;/\*head为空链表时或第一个结点的学号不小于num时返回空指针\*/

for ( p=head; p!=NULL; p=p->next )

{

q = p->next;

if ( (1) ) /\*条件\*/

return p;

}

}

struct node \*insert(struct node \*head,int num)

{/\*按学号顺序插入新结点，新结点的学号为num\*/

struct node \*p, \*q;

p = (struct node \*) (2) ;/\*分配内存空间\*/

p->num = num;

q = find(head,num);/\*找到插入位置在q的后面\*/

if (q==NULL)

{ /\*插入到链表头\*/

p->next = head;

return p;

}

(3) ;/\*插入到q的后面\*/

q->next = p;

return head;

}

int main()

{

struct node \*head=NULL, \*p;

int i, num;

for ( i=0; i<5; i++ )

{

scanf( "%d", &num );

head=insert(head,num);

}

for ( p=head; p!=NULL; p=p->next )

printf( "%d ", p->num );

printf("\n");

return 0;

}

【样例输入】 30 20 50 10 60

【样例输出】 10 20 30 50 60