```
#include "stdafx.h"
#include "string.h"
#include "malloc.h"
#define MAXSIZE 100

typedef struct lnode
{
    int data;
    struct lnode *next;
}LNode,*Link;

void Creat1(Link *L)//不带头单链表的创建
{
    Link s;
    int x;

    *L = NULL;

    while(scanf("%d", &x)==1)
    {
        s = new LNode;
        s->next = NULL;
        s->data = x;

        s->next = *L;
        *L = s;
    }
```

```
}

void Print1(Link L)//不带头单链表的输出
{
    Link p=L;
    while(p)
    {
        printf("%d ",p->data);
        p = p->next;
    }
    printf("\n\n");
}

int Sort1(Link *L)//不带头单链表的排序
{
    if(!(*L))
        return 0;

    Link L1;
    L1 = (*L)->next;
    (*L)->next = NULL;     //将单链表分裂成两个单链表L和L1
    Link q;
    Link p,pf;

    while(L1)          //外层循环摘取L1中的结点，循环一趟插入一个结点
    {
        q = L1;
        L1 = L1->next;
        q->next = NULL;

        pf = NULL;
        p = *L;

        while(p)   //内层循环遍历表L，查找插入位置
        {
            if(p->data > q->data)  //进行插入
            {
               if(!pf)     //插入到第一个位置的情况
               {
                 q->next = p;
                 *L = q;
                break;
                }
               else
                {
                pf->next = q;
                q->next = p;
                break;
    }
            }
            pf = p;
            p = p->next;
        }

        if(!p)//插入到表尾的情况
        {
            pf->next = q;
        }
    }
}
void Creat(Link *L)//带头单链表的创建
{
    *L=new LNode;
    (*L)->next = NULL;

    Link s;
    int x;
    while(scanf("%d",&x)==1)
    {
        s = new LNode;
        s->data = x;
        s->next = NULL;

        s->next = (*L)->next;
        (*L)->next = s;
    }
}

void Print(Link L)//带头单链表的输出
{
    Link p = L->next;
    while(p)
    {
        printf("%d ",p->data);
        p = p->next;
    }
}

void Sort(Link L)//带头单链表的排序
{
    Link L1;
    Link p,q;

    L1 = L->next;
    L->next = NULL; //分裂成两个单链表

    while(L1) //依次从L1摘取结点，并插入到L中
    {
        q = L1;  //以下三行为摘取一个结点
        L1 = L1->next;
        q->next = NULL;

        p = L; //内循环，遍历L查找插入位置，进行有序插入
        while(p && p->next)
        {
            if(q->data < p->next->data)
            {
                q->next = p->next;
                p->next = q;
                break;
            }
            p = p->next;
        }
        p->next = q;

    }
}
int _tmain(int argc, _TCHAR* argv[])
{
    Link L;
    Creat1(&L);
    Print1(L);
    printf("\n\n");

    Sort1(&L);
    Print1(L);
    int x;
    scanf("%d",&x);
    return 0;
}
```