样题 1、已知整数 a、b，假设函数 succ(x)=x+1、pred(x)=x-1，不许直接用“＋”、“－”运算符号，也不许用循环语句，只能利用函数 succ( )和 pred( )，试编写计算 a+b，a-b 的递归函数 add(a,b)，sub(a,b)，并在主程序中验证函娄的正确性。

```
#include "stdio.h"
#include "conio.h"
int succ(int x)
 {return x+1;}
int pred(int x)
 {return x-1;}
int add(int a,int b)
 {if(b==0) return a;
  if(b>0) return succ(add(a,pred(b)));
  else return pred(add(a,succ(b)));}
int sub(int a,int b)
 {if(b==0) return a;
  if(b>0) return pred(sub(a,pred(b)));
  else return succ(sub(a,succ(b)));}
void main()
 {int k,a,b;
  clrscr();
  printf("\n Please input a  b: ");
  scanf("%d%d",&a,&b);
  printf("\n a+b=%d",a+b);
  printf("\n a-b=%d",a-b);
  printf("\n add(a,b)=%d",add(a,b));
  printf("\n sub(a,b)=%d",sub(a,b));
  if((a+b==add(a,b))&&(a-b==sub(a,b)))
printf("\n It's right! ");
  else printf("\n It's wrong! \n\n");
  }
```

样题 2 试编写一个求解 Josephus 问题的函数。用整娄序列 1，2，3，……，n 表示顺序围坐在圆桌周围的人，并采用数组表示作为求解过程中使用的数据结构，然后使用 n=9，s=1，m=5，以及 n=9，s=1，m=0，或者 n=9，s=1，m=10 作为输入数据，验证的正确性。

Josephus 问题:设有 n 个人围坐在一个圆桌周围，现从第 s 个人开始报数，数到第 m 的人出列，然后从出列的下一个人重新开始报数，数到第 m 的人又出列，…，如此反复直到所有的人全部出列为止。Josephus 问题是：对于任意给定的 n,s 和 m，求出按出列次序得到的 n 个人员的序列。

```
#include "conio.h"
#include "iostream.h"
#define  N  100
void Josephus(int n,int s,int m)
{int i,j,k,l;
 int A[N];
 if(n<1 || s<1 || m<1) return;
 for(i=0;i<n;i++) A[i]=i+1;
 l=s-2;k=0;
 while(k<n)
 {
   for(j=0;j<m;)
   {
      l=(l+1)%n;
      if(A[l]!=-1)
    j++;
   }
   cout<<A[l]<<",";
   A[l]=-1;
   k=k+1;
 }
}
void main()
{
  int n,s,m;
  clrscr();
  cout<<"\nJosephus question Please check The Result:\n Please input:n s m= ";
  cin>>n>>s>>m;
  cout<<"\nJosephus("<<n<<"          "<<s<<"  "<<m<<")\n";
  Josephus(n,s,m);
  cout<<"\n\n\nPress any key to Exit.";
 getch();
}
```

样题 3 依次输入 10 个整数，分别用顺序表与单链表存储，并实现其就地逆置。

```
#include "iostream.h"
#include "conio.h"
#define ListSize 10
typedef struct {
    int data[ListSize];
    int lenght;
}SeqList;
typedef struct node {
    int data;
    struct node *next ;
}ListNode;
typedef ListNode *LinkList;
```

```
void main()
{
    int i,j,k;
    int a[ListSize],temp;
    SeqList SL,*L;
    ListNode *s,*r,*p;
    LinkList head;
    clrscr();
    cout<<"Input  Number:\n";
    for(i=0;i<ListSize;i++)  cin>>a[i];
    L = &SL;
    L->lenght = 0;
    for(i=0;i<ListSize;i++)  {
        L->data[i] = a[i];
        L->lenght ++;
    }
    cout<<"SeqList Order:\n";
    for(i=0;i<ListSize;i++)         cout<<"
"<<L->data[i];
    // invert SeqList
    j=0;k=L->lenght -1;
    while (j<k) {
    temp = L->data[j];
    L->data[j] = L->data[k];
        L->data[k]= temp;
        j++;k--;
    }
    cout<<"\nInvert SeqList Order:\n";
    for(i=0;i<ListSize;i++)         cout<<"
"<<L->data[i];
    head = NULL;r = NULL;
    i=0;
    while(i<ListSize)  {
        s=new ListNode[sizeof(ListNode)];
        s->data = a[i];
        if (head == NULL)
            head = s;
        else
            r->next =s;
        r = s;
        i++;
    }
    if (r) r->next=NULL ;
    cout<<"\nListNode Order:\n";
    s = head;
    while(s) {
        cout<<"  "<<s->data ;
    s = s->next ;
    }
    // invert ListNode
    s = head->next ; r = head;
    while(s) {
        p = s->next ;
        s->next = r;
        r = s;
        s = p;
    }
    head->next =NULL;
    head = r;
    cout<<"\nInvert ListNode Order:\n";
    s = head;
    while(s) {
        cout<<"  "<<s->data ;
    s = s->next ;
    }
    cout<<"\n\n\n";
}
```

样题 4　设有环队列类型如下：

```
        typedef struct {
            dnintype data[maxsize];
             int front,rear;
        }cyequeue;
        cycqueue  sq;
```

编写入队操作 int EnQueue(cycqueue sq,datarype x)的代码，并调试通过。（提示：将 x 入队列 sq，成功返回 1，否则，返回 0）

```
#include "stdio.h"
#include "stdlib.h"
#include "conio.h"
#include "string.h"
#include "malloc.h"
#define maxsize 10
#define datatype char
typedef struct {
    datatype data[maxsize];
    int front,rear;
}cycqueue;
cycqueue *sq;

void InitQueue(cycqueue *sq)
{
    sq->front = sq->rear =0;
}
int EnQueue(cycqueue *sq,datatype x)
```

```
{
    if(((sq->rear+1) % maxsize)==(sq->front % 
maxsize)) return 0;
    sq->data[sq->rear] = x;
    sq->rear = (sq->rear+1) % maxsize ;
    return 1;
}
void main()
{
    int i,j,k;
    char s[100],x;
    j=0;
    clrscr();
    sq = (cycqueue*)malloc(sizeof(cycqueue));
    InitQueue(sq);
    printf("\nInput string: ");
    gets(s);
    j=strlen(s);
    i = 0;
    while(j!=0) {
        x = s[i++];
        k = EnQueue(sq,x);
        if(k==1) printf("'%c' EnQueue 
OK!\n",x);
            else {
            printf("'%c' EnQueue 
Error,overflow!\n",x);
            break;
            }
         j--;
    }
}
```

样题 5 设有两个整数类型的顺序表 A（有 5 个元素）和 B（有 4 个元素），其元素均以从小到大的升序排列。请编写一个函数，将这两个顺序表合并成一个顺序表 C（要求 C 的元素也以从小到大的升序排列），并编写主程序验证该函数的正确性。

```
#include"iostream.h"
#include"conio.h"
#define LA 5
#define LB 4
#define LC LB+LA
void MergeSort(int A[],int B[],int C[])
{
    int i,j,k;
    i = 0; j = 0; k = 0;
    while (i<LA && j<LB )  {
        if (A[i] > B[j] ) C[k++] = B[j++];
        else C[k++] = A[i++];
    }
   while (i<LA) C[k++] = A[i++];
   while (j<LB) C[k++] = B[j++];
}
void main()
{
    int i;
    int A[LA],B[LB],C[LC];
    clrscr();
    cout<<"\nInput       SeqList       A(5 
integers):\n";
    for(i=0;i<LA;i++)  cin>>A[i];
    cout<<"\nInput       SeqList       B(4 
integers):\n";
    for(i=0;i<LB;i++)  cin>>B[i];
    cout<<"\nSeqList A :";
    for(i=0;i<LA;i++)  cout<<"  "<<A[i];
    cout<<"\nSeqList B :";
    for(i=0;i<LB;i++)  cout<<"  "<<B[i];
    MergeSort(A,B,C);
    cout<<"\nSeqList C :";
    for(i=0;i<LC;i++)  cout<<"  "<<C[i];
    cout<<"\n\n\n";
}
```

样题 6 编写一个算法程序 frequency,统计在一个输入字体串中各个不同字符出现的频度。算法返回两个数组：A[ ]记录字体串中有多少种不同的字符，C[ ]记录每一种字符的出现次数。此外，还要返回不同字符数。

```
#include "conio.h"
#include "stdio.h"
#include "iostream.h"
#include "string.h"
#define MaxSize 500
//统计字符串S有多少种字符,是些什么字符,和这些
字符出现的频度
int frequency(char A[],int C[],char S[],int max)
{
    int i,j,k;
    i=0;j=0;
    while (j<max)  {
        for (k=0;k<j;k++)  {
            if (A[k] != S[j]) continue ;
            else  {
                C[k]++;
                break;
            }
        }
        if (k==j) {
            A[i] = S[j];
            C[i] = 1;
            i++;
        }
        j++;
    }
    A[i] = '\0';
    return i;
}

void main()
{
    int  count,i,j;
    char A[MaxSize],*S;
    int  C[MaxSize];
    clrscr();
    cout<<"\nPlease input string for
counting:\n";
    j=strlen(gets(S));
    count=frequency(A,C,S,j);
    cout<<"\nThere is "<<count<<" kinds char in
String\n";
    for(i=0;i<count;i++){
        cout<<"Char["<<A[i]<<"]:"<<C[i]<<"
times.\n";

    }
    cout<<"\n\n\nPress any key to Exit.";
    getch();
}
```

样题 7　设在一个带表头结点的单链表中所有元素结点的数据值按递增顺序排列，请编写一个函数，删除表中所有大于 min，小于 max 的元素(若存在)，并编写主程序验证该函数的正确性。

```
//J7.CPP
#include "stdio.h"
#include "conio.h"
#include "malloc.h"
typedef struct node {
    char data;
    struct node *next ;
}ListNode;
typedef ListNode *LinkList;
LinkList CreateListR(void)//With HeadNode
{
    char ch;
    LinkList head =
(LinkList)malloc(sizeof(ListNode));
    ListNode *s,*r;
    r = head;
    while((ch=getchar())!='\n') {
        s = (ListNode
*)malloc(sizeof(ListNode));
        s->data = ch;
        r->next =s;
        r =s;
    }
    r->next = NULL;
    return head;
}
void DeleteNode(LinkList head, char min,char
max)
{
    ListNode *p,*s;
    p = head; s =NULL;
    while(p->next) {
        if (p->next->data > min) break;
        p = p->next ;
    }
    if (!p->next)  return;
    s = p;
    while(p->next) {
        if (p->next->data >= max) break;
        p = p->next ;
    }
    if (!p->next)  s->next = NULL;
    else s->next = p->next;
}
void main()
{
    char *hold;
    char min,max;
    LinkList head,s;
    clrscr();
    printf("Input char to build
ListNode(Min->Max):\n");
    head = CreateListR();
    printf("\nListNode element and element data
value :\n");
    s = head ;
    while(s->next) {
        s = s->next ;
        printf("%2c",s->data );
    }
    s = head ;
    printf("\n\nInput char min & max
(separator\" \"):");
    scanf("%c %c",&min,&max);
    DeleteNode(head,min,max);
    printf("\nListNode (%c<Deleted<%c)
element :\n",min,max);
    s = head ;
    while(s->next) {
            s = s->next ;
        printf("%2c",s->data );
    }
    printf("\n\n\n");
    printf("Enter to Exit.");
    gets(hold);
    gets(hold);
}
```

样题 8 所谓回文，是指从前向后顺读和从后向前倒读都一样的不含空白字符的串，例如 did，madamimadam,pop 即是回文。请编写一个递归程序，以差数一个串是否是回文。算法的声明为 int pal(char A[ ],int s,int c)。pal 判断从 s 到 e 中的字符串是否回文，通过函数返回是（1）或不是（0）。

```
#include "stdio.h"
#include "conio.h"
#define MaxSize 100
int pal(char A[],int s,int e)
{
    int i;
    if (s >= e) return 1;
    else return (A[s] == A[e]) &&
pal(A,++s,--e);
}
void main()
{
    char *hold;
    char A[MaxSize],B[MaxSize];
    int s,e,t,i,j;
    clrscr();
    printf("\nPlease Input palindrome test
String A: ");
    gets(A);
    printf("\nPlease Input start
place(integer): ");
    scanf("%d",&s);
    printf("\nPlease Input end place(integer):
");
    scanf("%d",&e);
    if(s>e){j=s;s=e;e=j;}
    t= pal(A,s-1,e-1);
    i=0;
    for(j=s-1;j<e;j++){B[i]=A[j];i++;}B[i]='\
0';
    if (t) printf("\nThis is a
palindrome:%s",B);
    else printf("\nThis is not a
palindrome:%s",B);
    printf("\n\n\n");
    printf("Enter to Exit.");
    gets(hold);
    gets(hold);
}
```

样题 9 已知二叉树中的结点类型用 BinTreeNode 表示，被定义为：strect BinTreeNode{char data;BintreeNode leftChild, rightChild; }; 其中 data 为结点值域，leftChild 和 right 和分别为指向左、右子女结点的指针域，根据下面函数声明编写出来一棵二叉树高度的函数代码，并编写主程序验证，该高度由函数返回。假定极结点的层次为 0，参数 BT 初始指向这棵二叉树的极结点。

Int BtreeHeight(BintreeNode* BT);

```
// 创建二叉树->装入数据->求树高->遍历显示->销毁   递归实现
#include "conio.h"
#include "stdio.h"
#include "iostream.h"
typedef char DataType;
typedef struct BinTreeNode //树的数据结构
{
    DataType        data;
    struct BinTreeNode *LeftChild,
*RightChild;
}BinTreeNode;
//初始化为空树
BinTreeNode * Initiate()
{
    BinTreeNode   *root = 0;
    return   root ;
}
//建节点
BinTreeNode * Creat(  DataType data  )
{
    BinTreeNode * Temp = new BinTreeNode ;
    Temp -> data = data ;
    Temp -> LeftChild = 0 ;
    Temp -> RightChild = 0;
    return Temp ;
}
//输入数据为BST(二叉排序树)的先根序列
void  Insert( BinTreeNode *&root , DataType data ) //在c下不能这样 Node *&root
{
    BinTreeNode *p = Creat( data );
    if( !root  ){
        root = p;
    }
    else if( p->data < root->data ){
        Insert ( root->LeftChild , p->data );
    }else{
        Insert ( root->RightChild , p->data );
    } //相等的就将数据装到右孩子
}
//测试BINTREE的高度
int BTreeHeight(struct BinTreeNode *BT)
{
    int h1,h2;
    if (!BT) return 0;
    else  {
        h1 = BTreeHeight(BT->LeftChild);
        h2 = BTreeHeight(BT->RightChild);
        if(h1>h2) return h1+1;
        else return h2+1;
    }
}
//递归中序遍历,显示从小到大
 void PrintTree(BinTreeNode * root)
{
    if( !root )  return ;
    PrintTree(root->LeftChild);
    cout<< root->data <<" ";
    PrintTree( root->RightChild );
    return ;
}
//释放BINTREE占用的内存空间
void  FreeTree(BinTreeNode * root)
{
    if( !root ) return;
    FreeTree(root -> LeftChild);
    FreeTree(root -> RightChild);
    delete root;//跟节点要最后删!
}
void main()
{
    char a;
    int height;
    BinTreeNode *Root = Initiate() ;
    clrscr();
    cout<<"\n\n***********DATA STRUCTURE EXAMPLE[8]***********\n";
    cout<<endl<<"Input data to create BinTree (Data format is Root in first BST,\nExample:5 3 2 4 7 8#, \"#\" to end):"<<endl;
    cin>>a;
    while( (a != '#')&&cin.good() ){//遇到非法输入同样退出循环
        Insert( Root ,  a );
        cin>>a ;
    }
    if(!cin.good()){//输出错误信息
        cout<<" the type is error ! "<<endl;
    }
    height = BTreeHeight(Root);
    cout<<endl<<"BinTree Height is: "<<height<<endl;
    cout<<"Root in middle BST list is: ";
    PrintTree(Root);
    cout<<endl;
    FreeTree(Root);//销毁树 防止内存泄漏
    cout<<"\n\n\nPress any key to Exit.";
    getch();
}
/*数据结构书本(P175)

          (5)
         /   \
       (3)    (7)
      /   \     \
    (2)   (4)   (8)  这棵BST的先根遍历序列为:5 3 2 4 7 8 (可作测试用例数据序列输入创建此二叉树)



                 (2)
          \
          (3)
            \
            (7)
            / \
          (5) (8)
         /
       (4)          这棵BST的先根遍历序列为:2 3 7 5 4 8 (可作测试用例数据序列输入创建此二叉树)
*/
```

样题 10 假定元素类型为长整型的一维数组 R(n) 中保存着按关键码升序排列的 n 个元素。关键码域 key 的类型为 long，请按照下面的函数声明编写一个非递归算法，从一维数组及 R[n]中用折半搜索法找出关键码等于 k 的元素，若搜索成功则返回元素位置（即元素下标），否则返回－1。

```
    Int BinSearch(long R[ ],int n,long k);
#include "stdio.h"
#include "conio.h"
#define N  100
//折半查找函数
int BinSearch(long R[],int n,long k)
{
    int low =1,high =n,mid;
    while(low<=high) {
        mid = (low + high)/2;
        if(R[mid]==k) return mid;
        if(R[mid] > k)
            high = mid -1;
        else
            low = mid +1;
    }
    return -1;
}
void main()
{
    char *hold;
    int i,n,t;
    long R[N],k;
    clrscr();
    printf("\n\nInput Long Integer(Min->Max), While '-9999' to end:\n");
    i=1;
    while(1) {
        scanf("%ld",&R[i]);
        if(R[i]==-9999) break;
        else i++;
    }
    n = i;
    printf("\n\nLong Integer Array: ");
    for(i=1;i<n;i++) printf("%-5d",R[i]);
    printf("\n\nInput you want to find Keyword: ");
    scanf("%ld",&k);
    t = BinSearch(R,n,k);
    if (t == -1) printf("\n\nKeyword %d not found.",k);
    else {printf("\n\nKeyword %d ",k);printf("Position is %d.",t);}
    printf("\n\n\n");
    printf("\n\nEnter to Exit.");
    gets(hold);
    gets(hold);
}
```

样题 11 有一种简单的排序算法，叫做计数排序(count Sorting)。这种排序算法对一个待排序的表（用数组表示）进行排序，并将排序结果存放到另一个新的表中，必须注意的是，表中所有待排序的关键码互不相同。计数排序算法针对表中的每个记录，扫描待排序的表一趟，统计表中有多少个记录的关键码比该记录的关键码小。假设针对某一个记录，统计出的计数值为 C，哪 么，这个记录在新的有序表中的合适的存放位置即为 c。请编写实现计数排序的算法;

```
#include "stdio.h"
#include "conio.h"
#define N  30
//计数法排序函数,关键字不可重复!
void CountSorting(int A[],int count,int B[])
{
    int i,j,k;
    for(i=0;i<count;i++) {
        k = 0;
        for(j=0;j<count;j++) if(A[j]<A[i]) k++;
        B[k]=A[i];
    }
}
void main()
{
    char *hold;
    int i;
    int count;
    int A[N],B[N];
    clrscr();
    printf("\n\nInput different 
Integers(quantity<=100) to do counting sort test, 
\n\"-9999\" to end:\n");
    i=0;
    while(1) {
        scanf("%d",&A[i]);
        if(A[i]==-9999 ||i>29) break;
        else i++;
    }
    count = i;
    printf("\n\nThe Input Array Order:\n");
    for(i=0;i<count;i++) printf("%-5d",A[i]);
    CountSorting(A,count,B);
    printf("\n\nCounting sort Order:\n");
    for(i=0;i<count;i++) printf("%-5d",B[i]);
    printf("\n\n\n");
    printf("\n\nEnter to Exit.");
    gets(hold);
    gets(hold);
}
```

样题 12 有 10 个值在 1 到 10000 的整数（存放于数组 A[0]—A[9]），编写一个利用散列方法的算法程序，以最少的数据比较次数和移动次数对它们进行排序。

```
#include "conio.h"
#include "iostream.h"
#define  KeySize 5
#define  Radix 10
typedef struct queuenode{
    int data;
    struct queuenode *next;
}QueueNode;
typedef struct {
    QueueNode *front;
    QueueNode *rear;
}LinkQueue;
void InitQueue(LinkQueue *Q)
{
    Q->front=Q->rear=NULL;
}
int QueueEmpty(LinkQueue *Q)
{
    return Q->front==NULL&&Q->rear==NULL;
}
void EnQueue(LinkQueue *Q,int x)
{
    QueueNode *p=new QueueNode;
    p->data=x;p->next=NULL;
    if(QueueEmpty(Q)) Q->front=Q->rear=p;
    else{
        Q->rear->next=p;
        Q->rear=p;
    }
}
int DeQueue(LinkQueue *Q)
{
    int x;
    QueueNode *p;
    p=Q->front;
    x=p->data;
    Q->front=p->next;
    if(Q->rear==p) Q->rear=NULL;
    delete p;
    return x;
}
void Distribute(int R[],LinkQueue B[],int j)
{
    int i,k,t;
    j=KeySize-j;
    for(i=0;i<Radix;i++){
        k=R[i];
        for(t=1;t<j;t++)k=k/10; //此处用到
类似散列函数,所以可以称为非比较的散列排序
        k=k%10;
        EnQueue(&B[k],R[i]);
    }
}
void Collect(int R[],LinkQueue B[])
{
    int i=0,j,m;
    for(j=0;j<Radix;j++)
        while(!QueueEmpty(&B[j])){
            R[i++]=DeQueue(&B[j]);
        }
}
void RadixSort(int R[])
{
    LinkQueue B[Radix];
    int i,j,k=1;
    for(i=0;i<Radix;i++) InitQueue(&B[i]);
    for(i=KeySize-1;i>=0;i--){
        Distribute(R,B,i);
        Collect(R,B);
        cout<<"\nCollect "<<k<<":";
        for(j=0;j<Radix;j++)        {cout<<"
"<<R[j];}
        k++;
    }
}
void main()
{
    int i=0;
    int A[Radix];
    clrscr();
    cout<<"\nInput    "<<Radix<<"    integer
(<=10000) to the SeqList A :\n";
    while(i<Radix)
{cin>>A[i];if(int(A[i])>10000)A[i]=10000;i+
+;}
    RadixSort(A);
    i=0;
    cout<<"\n\nHashSort :";
    while(i<Radix) {cout<<" "<<A[i];i++;}
    cout<<"\n\n\nPress any key to Exit.";
    getch();
}
```

样 题 13 设两个初始归并段为：(10,15,31,9,20),(22,34,37,6,15,42),请编写程序 merge，实现两路归并算法。

```
#include "stdio.h"
#include "conio.h"
#define N  100
//将两个有序子串归并成一个有序串,数据结构书本
(P157)
void Merge(int R[],int low,int m,int high)
{
    int i =low,j=m+1,p=0;
    int R1[N];
    while(i<=m && j<=high) R1[p++] =
(R[i]<=R[j]?R[i++]:R[j++]);
    while(i<=m ) R1[p++] = R[i++];
    while(j<=high ) R1[p++] = R[j++];
    for(p=0,i=low;i<=high;p++,i++)
R[i]=R1[p];
}
//用分治法进行二路归并排序,数据结构书本(P159)
void MergeSortDC(int R[],int low,int high)
{
    int mid;
    if(low<high) {
        mid = (low + high)/2;
        MergeSortDC(R,low,mid);
        MergeSortDC(R,mid+1,high);
        Merge(R,low,mid,high);
    }
}
void main()
{
    char *hold;
    int i;
    int low,high;
    int R[]={10,15,31,9,20,22,34,37,6,15,42};
    low =0;high =10;
    clrscr();
    printf("\n\nUnSort Array Order:\n");
    for(i=low;i<=high;i++)
printf("%-5d",R[i]);
    MergeSortDC(R,low,high);
    printf("\n\nMergeSort Array Order:\n");
    for(i=low;i<=high;i++)
printf("%-5d",R[i]);
    printf("\n\n\nEnter to Exit.");
    gets(hold);
}
```

样题 14 在二叉搜索树上删除一个有两个子女的结点时，可以采用以下方案：用左子树 T_L 上具有最大关键码的结点，或者用右子树 T_R 上具有最小关键码的结点顶替，再递归地删除适当的结点。随机选择其中一个方案，并编写程序实现这个删除方法。

```
#include "stdio.h"
#include "conio.h"
#include "malloc.h"
typedef struct node{
    int key;
    struct node *lchild,*rchild;
}BSTNode;
typedef BSTNode *BSTree;
void InsertBST(BSTree *Tptr,int key)
{
    BSTNode *f,*p = *Tptr;
    while(p) {
        if(p->key==key) return;
        f = p;
        p = (key<p->key)?p->lchild:p->rchild;
    }
    p = (BSTNode *)malloc(sizeof(BSTNode));
    p->key =key; p->lchild =p->rchild =NULL;
    if(* Tptr == NULL)
        *Tptr = p;
    else
        if(key<f->key)
            f->lchild = p;
        else f->rchild = p;
}
BSTree CreateBST(void)
{
    BSTree T=NULL;
    int key;
    scanf("%d",&key);
    while(key) {
        InsertBST(&T,key);
        scanf("%d",&key);
    }
    return T;
}
void InorderBST(BSTree T)
{
    if(T) {
        InorderBST(T->lchild);
        printf("%5d",T->key);
        InorderBST(T->rchild);
    }
}
void DelBSTNode(BSTree *Tptr,int key)
{
    BSTNode *parent=NULL,*p=*Tptr,*q,*child;
    while(p) {
        if(p->key==key)break;
        parent = p;
        p=(key<p->key)?p->lchild:p->rchild;
    }
    if(!p) return;
    q=p;
    if(q->lchild && q->rchild)
        for
(parent=q,p=q->rchild;p->lchild;parent=p,p=p-
>lchild);
    child = (p->lchild)?p->lchild:p->rchild;
    if(!parent)
        *Tptr=child;
    else  {
        if (p==parent->lchild) parent->lchild
= child;
        else parent->rchild = child;
        if (p!=q)
            q->key = p->key;
    }
    free(p);
}
void main()
{
    char *hold;
    int key;
    BSTree T;
    clrscr();
    printf("\n\nInput data to build BST ,\"0\"
to end:\n");
    T = CreateBST();
    printf("\nBSTree Order: ");
    InorderBST(T);
    printf("\n\nInput node key to delete: ");
    scanf("%d",&key);
    DelBSTNode(&T,key);
    printf("\nBSTree Order: ");
    InorderBST(T);
    printf("\n\n\nEnter to Exit.");
    gets(hold);
    gets(hold);
}
```

样题 15　编写程序，用栈实现 10 进制正整数到 2 进制整数的转换。

```
#include "stdio.h"
#include "conio.h"
#define StackSize 100
typedef struct {
    int data[StackSize];
    int top;
}SeqStack;
void InitStack(SeqStack *S){S->top = -1;}
int  StackEmpty(SeqStack *S){return S->top ==
-1;}
int  StackFull(SeqStack *S){return S->top ==
StackSize -1;}
void Push(SeqStack *S,int x)
{
    if(StackFull(S)) {
        printf("Stack overflow");
        return;
    }
    S->data[++S->top] =x;
}
```

```
int Pop(SeqStack *S)
{
    if(StackEmpty(S)) {
        printf("Stack underflow");
        return 0;
    }
    return S->data[S->top--];
}
int StackTop(SeqStack *S)
{
    if(StackEmpty(S)) {
        printf("Stack is Empty");
        return 0;
    }
    return S->data[S->top];
}
void MultiBaseOutput(int N,int B)
{
    int i;
    SeqStack S;
    InitStack(&S);
    while(N) {
        Push(&S,N % B);
        N = N / B;
    }
    while(!StackEmpty(&S)) {
        i = Pop(&S);
        if (i>9) printf("%c",(i-10)+65);
        else printf("%d",i);
    }
}
void main()
{
    char *hold;
    int N,B;//N为十进制正整数,B为需要转的目标
进制,例如:2,8,16进制等
    clrscr();
    printf("\n\nPlease input decimal positive
integer N: ");
    scanf("%d",&N);
    printf("\nPlease input data in system
format B(Example:2,8,16,etc.):");
    scanf("%d",&B);if(B<2 ||B>16)B=2;
    printf("\nOutput %d in system data:",B);
    MultiBaseOutput(N,B);
    printf("\n\n\nEnter to Exit.");
    gets(hold);
    gets(hold);
}
```

样题 16 编写程序，要求能读入集合 A 的一串整数（以－9999 为结束标记，整数个数小于 1000）和集合 B 的一串整数（以－9999 为结束标记，整数个娄小于 1000），计算出 A 与 B 的交集，并以由小到大的次序输出 A 与 B 的交集中的所有整数（输入整数时，相邻的两个用安全可靠隔开，为 A 或 B 输入时，同一个数可能出现多次，而 A 与 B 的交集中同一个数不能出现多次）。

```
#include "stdio.h"
#include "math.h"
#include "conio.h"
#include "ctype.h"
#include "stdlib.h"
#define N  1000
#define END -9999
void InsertSort(int R[],int n)
{
    int i,j;
 // int count=0;
    for(i=2;i<=n;i++)
        if (R[i]<R[i-1]) {
            R[0]=R[i];
            j=i-1;
            do {
                R[j+1] = R[j];
                j--;
            }while(R[0]<R[j]);
            R[j+1]=R[0];
        }
}
int Intersection(int A[],int a,int B[],int b,int
R[])
{
    int i,j,k;
    k=0;
    for(i=1;i<=a;i++)
        for(j=1;j<=b;j++)
            if (A[i]==B[j]) {
                R[++k]=A[i];
                break;
            }
    return k;
}
int Simplify(int R[],int n)
{
    int i,k;
    int T[N],t;
    if(n<=1) return n;
    t = 1;
    T[t]=R[1];
    k=R[1];
    for(i=2;i<=n;i++)
        if (R[i]>k) {
            T[++t] = R[i];
            k=R[i];
        }
    for(i=1;i<=t;i++) R[i]=T[i];
    return t;
}
int InputData(int R[])
{
    int i,k;
    char str[N],A[10];
    char *s;
    gets(str);
    s = str;
    k =1;
    while (k<1000) {
        if(*s=='\0')break;
        if(toascii(*s)==32) {
            s++;
            continue;
        }
        i = 0;
        while(toascii(*s)!=32 && *s!='\0') {
            A[i++]= *s;
            s++;
        }
        A[i]='\0';
        if (atoi(A) == END ) break;
        else R[k++]= atoi(A);
    }
    return k-1;
}
void main()
{
    char *hold;
    int i;
    int a,b,r;
    int A[N],B[N],R[N];
    clrscr();
    printf("\n\nInput DataSet A for
intersection(AnB):\n");
    a = InputData(A);
    printf("\n\nInput DataSet B for
intersection(A&B):\n");
    b = InputData(B);
    printf("\n\nDataSet A: ");
    for(i=1;i<=a;i++) printf("%5d",A[i]);
    printf("\n\nDataSet B: ");
    for(i=1;i<=b;i++) printf("%5d",B[i]);
    r = Intersection(A,a,B,b,R);
    InsertSort(R,r);
    r = Simplify(R,r);
    printf("\n\nIntersection (AnB):");
    for(i=1;i<=r;i++) printf("%5d",R[i]);
    printf("\n\n\nEnter to Exit.");
    gets(hold);
}
```

样题 17 编写程序，要求能根据读入的数据构造有向图 G，并输出 G 的 DFS 序列（从 V0 开始），图的输入形式为 n V0 Vi0 V1 Vi1 V2 Vi2… Vi Vin-1-1（-1,-1 为输入结束标记，其余的值都>=0 且<n），它们都是整数，且 100>n>0。

```
#include "stdio.h"
#include "stdlib.h"
#include "malloc.h"
#include "string.h"
#include "conio.h"
#define MaxVertexNum 100
typedef int VertexType;
typedef int EdgeType;
typedef struct node {
    int adjvex;
    struct node *next;
}EdgeNode;
typedef struct vnode{
    VertexType vertex;
    EdgeNode *firstedge;
}VertexNode;
typedef VertexNode AdjList[MaxVertexNum];
typedef struct{
    AdjList adjlist;
    int n,e;
}ALGraph;
int visited[MaxVertexNum];
void split(char str[],int e[])
{
     char ch;
```

```
      int i,j,t;
      int len = strlen(str);
      for(i=0;i<100;i++) e[i]=0;
      for(i=0,j=1,t=0; i<len; i++ )
      {
      ch= str[i];
      if(ch != ’ ’ )
      {
        t =t*10 + ch-48;
      }
      else
      {
        if( j==1 )       e[j++] = t;
        else if(t !=0 )  e[j++] = t;
        t= 0;
      }
     }
     if( t!=0 )  e[j++] = t;
     if( j >1 )  e[0]=j-2;
}
void CreateALGraph(ALGraph *G)
{
    int temp;
    int i,j,k;
    char str[300];
    int  e[100];
    int  es[100][100];
    EdgeNode *s;
    clrscr();
    printf("Input Vertex Number\n");
    printf("\nInput the Node Vk,Vik (Format:
Vk Vk0 Vk1 Vk2....Vki...Vkn-1 and \"-1\" to
end):\n");
    printf("\n");
    scanf("%d",&G->n);
    gets(str); //clear buffer of the scanf()
    gets(str); //get data

    k=0;
    while(strcmp(str,"-1"))
    {
       split(str,e);
       for(i=0;i<2+e[0];i++)
          es[k][i] = e[i];    //save e[] into
es[][]
       k++;
       gets(str);
    }
    //save es[][] data into G
    j=G->n;
    for(i=0;i<G->n;i++)
    {
           G->adjlist[i].vertex =i;
           G->adjlist[i].firstedge=NULL;
    }
    for(j=0; j<k; j++)
    {
       for(i=es[j][0]+1; i>=2; i--) //From
small to big
       {
        s=(EdgeNode
*)malloc(sizeof(EdgeNode));
        temp = es[j][i];
        s->adjvex=temp;
        temp= es[j][1];
        s->next=G->adjlist[temp].firstedge;
        G->adjlist[temp].firstedge=s;
       }
    }
    //end
----------------------------------------//
    for(i=0;i<G->n;i++)
       visited[i]=0;
}
void DFS(ALGraph *G,int i)
{
    int kk;
    EdgeNode *p;
    printf("->%d",G->adjlist[i].vertex);
    visited[i]=1;
    p=G->adjlist[i].firstedge;
    while(p){
        if(!visited[p->adjvex])
DFS(G,p->adjvex);
        p=p->next;
    }
}
void DSFTraverse(ALGraph *G)
{
    int i;
    for(i=0;i<G->n;i++)
        if(!visited[i]) DFS(G,i);
}
void main()
```

```
{
    int i;
    ALGraph *G;
       CreateALGraph(G);
       DSFTraverse(G);
       getch();
}
```

样题 18　编写程序,要求能读入一串整数(以-9999 为结束标记）并对它们进行从小到大直接插入排序,同时输出排序时对这些整数进行比较的总次数（输入整数时，相邻的两个用安全可靠隔开，整数个数＜2000)。

```
#include "stdio.h"
#include "conio.h"
#include "stdlib.h"
#include "ctype.h"
#define N  2000
#define S  10000
#define END -9999
int InsertSort(int R[],int n)
{
   int i,j;
   int count=0;
   for(i=2;i<=n;i++)
       if (count++,R[i]<R[i-1]) {
           R[0]=R[i];
           j=i-1;
           do {
               R[j+1] = R[j];
               j--;
           }while(count++,R[0]<R[j]);
           R[j+1]=R[0];
       }
    return count;
}
int InputData(int R[])
{
    int i,k;
    char str[N],A[10];
    char *s;
    gets(str);
    s = str;
    k =1;
    while (k<2000 || k==END) {
        if(*s=='\0')break;
        if(toascii(*s)==32) {
            s++;
            continue;
        }
        i = 0;
        while(toascii(*s)!=32 && *s!='\0') {
            A[i++]= *s;
            s++;
        }
        A[i]='\0';
        if (atoi(A) == END ) break;
        else R[k++]= atoi(A);
    }
    return k-1;
}
void main()
{
    char *hold;
    int i;
    int n,count;
    int R[N];
    clrscr();
    printf("\n\nInput DataSet for 
InsertSort(quantity<2000,\"-9999\" to 
end):\n");
    n = InputData(R);
    printf("\n\nInput  DataSet  :");
    for(i=1;i<=n;i++) printf("%5d",R[i]);
    count = InsertSort(R,n);
    printf("\n\nInsertSort Order:");
    for(i=1;i<=n;i++) printf("%5d",R[i]);
    printf("\n\ncompare times is : %d",count);
    printf("\n\n\nEnter to Exit.");
    gets(hold);
}
```

样题 19　编写程序，要求能读入集合 A 的一串整数（以-9999 为结束标记，整数个数小于 1000）和集合 B 的一串整数（以-9999 为结束标记，整数个数小于 1000)，计算并以从小到大的次序输出 A-B 的所有元素（为 A 或 B 输入时，同一个数可能出现多次，而 A 与 B 的差集中同一个数不能出现多次)。

```
#include "stdio.h"
#include "stdlib.h"
#include "math.h"
#include "conio.h"
#include "ctype.h"
#define N  1000
#define S  10000
#define END -9999
```

```
void InsertSort(int R[],int n)
{
    int i,j;
    for(i=2;i<=n;i++)
       if (R[i]<R[i-1]) {
           R[0]=R[i];
           j=i-1;
           do {
               R[j+1] = R[j];
               j--;
           }while(R[0]<R[j]);
           R[j+1]=R[0];
       }
}
int Subtract(int A[],int a,int B[],int b,int
R[])
{
    int i,j,k;
    k=0;
    for(i=1;i<=a;i++) {
        for(j=1;j<=b;j++)
            if (A[i]!=B[j]) continue;
            else break;
        if (j>b) R[++k]=A[i];
    }
    return k;
}
int Simplify(int R[],int n)
{
    int i,k;
    int T[N],t;
    if(n<=1) return n;
    t = 1;
    T[t]=R[1];
    k=R[1];
    for(i=2;i<=n;i++)
        if (R[i]>k) {
            T[++t] = R[i];
            k=R[i];
        }
    for(i=1;i<=t;i++) R[i]=T[i];
    return t;
}
int InputData(int R[])
{
    int i,k;
    char str[N],A[10];
    char *s;
    gets(str);
    s = str;
    k =1;
    while (k<1000) {
        if(*s==’\0’)break;
        if(toascii(*s)==32) {
            s++;
            continue;
        }
        i = 0;
        while(toascii(*s)!=32 && *s!=’\0’) {
            A[i++]= *s;
            s++;
        }
        A[i]=’\0’;
        if (atoi(A) == END ) break;
        else R[k++]= atoi(A);
    }
    return k-1;
}
void main()
{
    char *hold;
    int i,a,b,r;
    int A[N],B[N],R[N];
    clrscr();
    printf("\n\nInput DataSet A for (A-B): ");
    a = InputData(A);
    printf("\n\nInput DataSet B for (A-B): ");
    b = InputData(B);
    printf("\n\nDataSet A: ");
    for(i=1;i<=a;i++) printf("%6d",A[i]);
    printf("\n\nDataSet B: ");
    for(i=1;i<=b;i++) printf("%6d",B[i]);
    r = Subtract(A,a,B,b,R);
    InsertSort(R,r);
    r = Simplify(R,r);
    printf("\n\nSubtract A-B: ");
    for(i=1;i<=r;i++) printf("%6d",R[i]);
    printf("\n\n\nEnter to Exit.");
    gets(hold);
}
```