7.判断以二叉链表存储的二叉树是否为完全二叉树的算法描述如下：  
#define MAX 100  
void checkt(BTnode \*t)  
 { Btnode \*s[MAX];  
   int i,n=0;  
   for(i=0;i<MAX;i++)  
    s[i]=MAX;  
   if(t==NULL)  
    return 0;  
   s[0]=t;  
   while(i<=n)  
    { if(!s[i])  
       return 0;  
      if(s[i]->lchild)  
       { n=2\*i+1;  
         s[n]=s[i]->lchild;  
       }  
      if(s[i]->rchild)  
        { n=2\*i+2;  
          s[n]=s[i]->rchild;  
        }  
       i++;  
    }  
   return 1;  
 }

1．按中序遍历二叉排序树可得到一个按键值从小到大排列的有序表，利用这个特点来判别二叉树是否为二叉排序树，算法如下：  
#define max 99  
#define min 0  
int judge(BTnode \*bt)  
  { BTnode \*s[max],\*p=bt;  
    int top=0,preval=min;  
    do {   
        while(p)  
          {s[top++]=p;  
           p=p->lchild;  
          }  
        if(top>0)  
         {p=s[--top];  
          if(p->data<preval)  
           return 0;  
          preval=p->data;  
          p=p->rchild;  
         }  
      }while(p||top>0)  
     return 1;  
  }

快速排序非递归算法描述如下（其中s为顺序栈）：  
void Quicksort1(RecType t[],int n)  
 {int top,low=0,high=n,i,j,\*s;  
  RecType x;  
  S=(int \*)malloc(sizeof(int)\*2\*n);  
  Top=0;  
  Do{  
while(low<high)  
{i=low;j=high;x=r[i];  
   do{  
     while((r[i].key>=x.key)&&(i<j))  
      j--;  
    if(i<j){  
     r[j]=r[i];j--;}  
    }while(i<j);  
   r[i]=x;  
   if((i+1)<high)  
   {s[++top]=i+1;  
    s[++top]=high;}  
  high=i-1;  
if(top>0)  
  {low=s[top--];  
   high=s[top--];}  
}while(top>0||low<high);  
}