一、选择题

CBBDB　CBDAC　CDCB

二、填空题

1. ABCDEFGH
2. 2^k-1
3. 68
4. $n_2+2n_3+3n_4+1$
5. （n+1）/2
6. 2^{i-1}；2^k-1；2^{k-1}
7. O(n)；O(log n)

三、应用题

1. 解：设二叉树中叶子结点、度为 1、度为 2 的结点数目分别 n_0、 n_1、n_2，

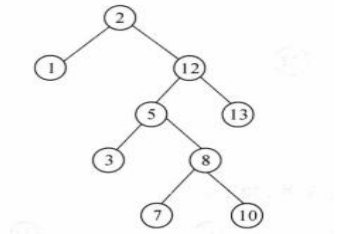
则有： $n_1 = 0$ 或 $n_1 = 1$

$n_0 + n_1 + n_2 = 9999$

$n_0 = n_2 + 1$

解方程组得：$n_0 = 5000$, $n_1 = 0$, $n_2 = 4999$

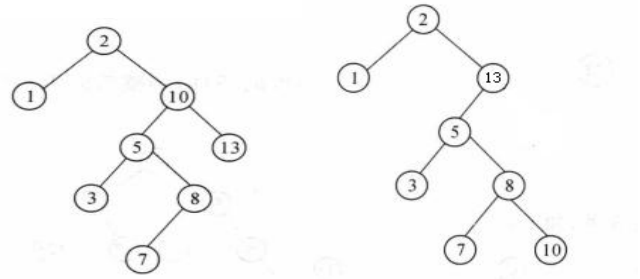
2. 答：（1）构造的二叉排序树如下：



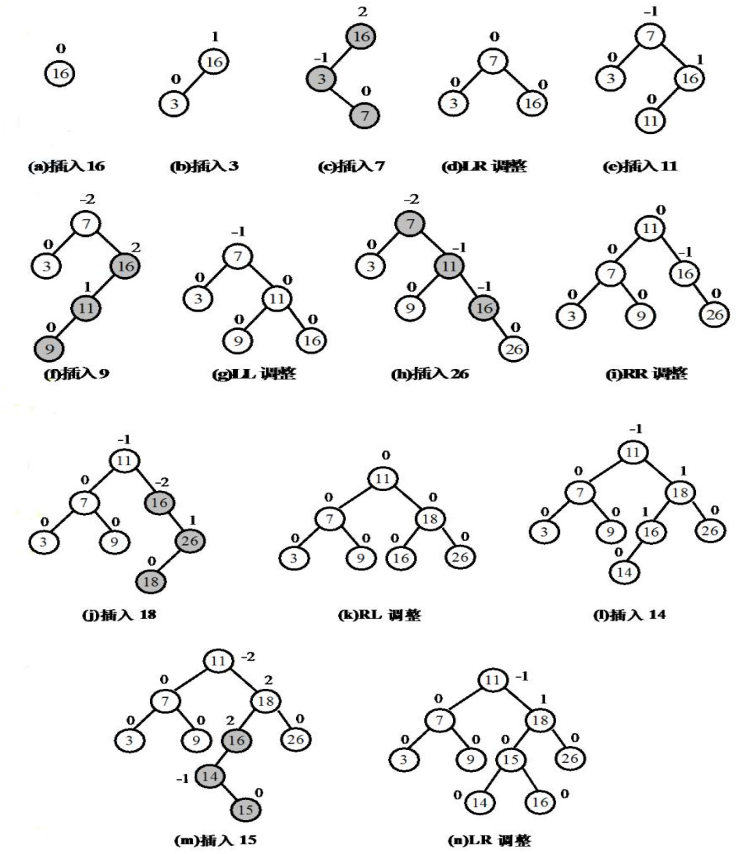
（2）中序遍历为：1，2，3，5，7，8，10，12，13

后序遍历为：1，3，7，10，8，5，13，12，2

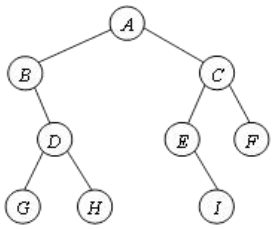
（3）删除“12”后的二叉排序树如下：



3.



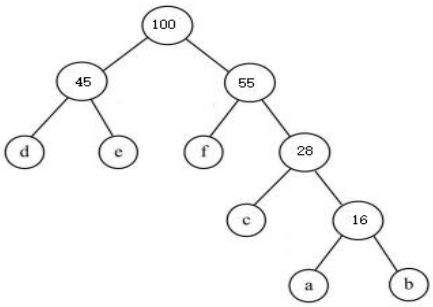
4.答：最后构造的二叉树如图所示。



其层次遍历序列为 ABCDEFGHI。

5.答：（1）　　（2）

6.解：根据权值构造哈夫曼树如图所示：



WPL = 7*4+9*4+12*3+22*2+23*2+27*2=244

令所有左分支取编码为 0，所有右分支取编码为 1，则各字符对应的哈夫曼编码分别为：

a: 1110　　b: 1111　　c: 110　　d: 00　　e: 01　　f: 10

7.解：（1）初始状态如图 1 所示，由于希望先输出最大值，故应调整为最大堆，调整后如图 2 所示。

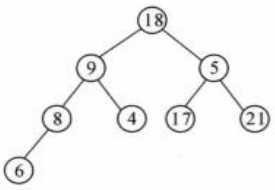


图 1

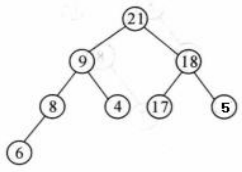


图 2

（2）输出最大值后．用最后一个元素值替换根节点，继续调整为最大堆，则此时根节点为次大值，如图 3 所示。

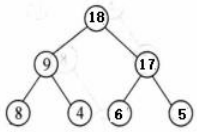
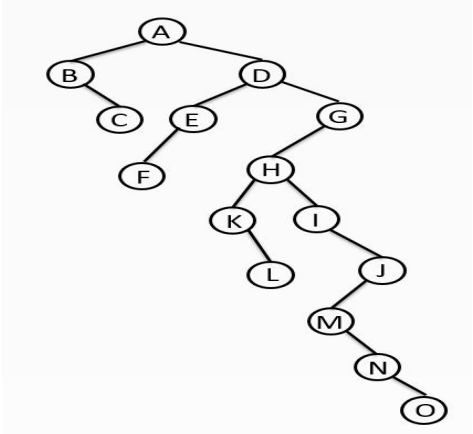


图 3

8.答：转化为的二叉树如图所示：



四、算法设计题

1.
```
int n1=0;
void preorder(bstnode *t)
{
    if(t == NULL)        //若是空树则直接返回
        return;
    else
    {       if((t->lchild == NULL && t->rchild != NULL)||
              (t->rchild == NULL && t->lchild != NULL))
                  n1++;          //如果结点为度为1的结点，则n1+1
            preorder(t->lchild);
            preorder(t->rchild);
    }
}
```

2.
```
BTNode *Findx(BTNode *b,char x)   //在二叉树 b 中查找值为 x 的结点
{  BTNode *p;
   if (b==NULL)
        return NULL;       //空树返回 NULL
   else
   {    if (b->data==x)
            return b;         //结点值等于 x，返回其地址
        p=Findx(b->lchild,x);
        if (p!=NULL)
            return p;
        return Findx(b->rchild,x);
   }
}
```

3.
```
int BTreeCount(BTNode *BT)      //二叉树中结点的总数
{
    BTNode * T;
    int count = 0;
    T = BT;     //从根结点出发

    Stack S = CreateStack();
    while( T || !IsEmpty(S) ){
        while( T ){       // 一直向左将沿途结点压入堆栈
            Push(S, T);
            T = T->lchild;
        }
    T = Pop(S);     // 结点弹出堆栈
    count++;      //结点数+1
    T = T->rchild;     // 转向右子树
    }
    return count;
```

```
    }
4. int level(BTNode *t,BTNode *p) //在二叉排序树 t 中查找结点 p 所在的层次
   {    int count = 0;
        if (t==NULL)     //若是空树则返回 0
            return 0;
        else
        {      count++;
               while(t->data != p->data)
               {
                     if(t->data < p->data)
                        t = t->rchild;     //向右子树中移动，继续查找
                     else
                        t = t->lchild;     //向左子树中移动，继续查找
                     count++;             //更新 t 所在的层次
               }
               return count;
        }
   }
5.  void ChangeLR(BTNode &*T)   //交换二叉树中每个结点的左右孩子
     {  BTNode *temp;
        if (T->lchild == NULL && T->rchild == NULL)
            return ;        //若是叶结点则直接返回
        else       //交换结点的左右孩子
        {    temp = T->lchild;
             T->lchild = T->rchild;
             T->rchild = temp;
        }
        ChangeLR(T->lchild);
        ChangeLR(T->rchild);
     }
6.   ElemType BSTMin(BTNode *T)
     {
     if(T == NULL)       //若是空树则退出程序
        exit(1);
     while(T->lchild != NULL)
        T = T->lchild;        //沿左分支一直向下，直到最左端点
     return T->data;       //返回最小元素值
  }
```