厦门大学《数据结构与算法》期中小测

☛ 请将答案按序写在学校统一印制的专用答题卷上，写在本卷或自备纸上者一律不得分。

一．**简答题** (含 6 个小题，每小题 5 分，计 30 分)

1．存储结构由哪两种基本的存储方法实现？

答案：

（1）顺序存储结构

顺序存储结构是借助元素在存储器中的相对位置来表示数据元素之间的逻辑关系，通常借助程序设计语言的数组类型来描述。

（2）链式存储结构

顺序存储结构要求所有的元素依次存放在一片连续的存储空间中，而链式存储结构，无需占用一整块存储空间。但为了表示结点之间的关系，需要给每个结点附加指针字段，用于存放后继元素的存储地址。所以链式存储结构通常借助于程序设计语言的指针类型来描述。

2．试举一个数据结构的例子，叙述其逻辑结构和存储结构两方面的含义和相互关系。

答案：

例如有一张学生基本信息表，包括学生的学号、姓名、性别、籍贯、专业等。每个学生基本信息记录对应一个数据元素，学生记录按顺序号排列，形成了学生基本信息记录的线性序列。对于整个表来说，只有一个开始结点(它的前面无记录)和一个终端结点(它的后面无记录)，其他的结点则各有一个也只有一个直接前趋和直接后继。学生记录之间的这种关系就确定了学生表的逻辑结构，即线性结构。

这些学生记录在计算机中的存储表示就是存储结构。如果用连续的存储单元(如用数组表示)来存放这些记录，则称为顺序存储结构；如果存储单元不连续，而是随机存放各个记录，然后用指针进行链接，则称为链式存储结构。

即相同的逻辑结构，可以对应不同的存储结构

3．在如下数组 A 中链接存储了一个线性表，表头指针为 A [0].next，试写出该线性表。

A	0	1	2	3	4	5	6	7
data		60	50	78	90	34		40
next	3	5	7	2	0	4		1

线性表为：（78，50，40，60，34，90）

4. 证明：如果二叉树 T 的叶子结点数为 n_0，度为 2 的结点数为 n_2，则 $n_0=n_2+1$。

证明：设二叉树 T 共有 n 个结点，叶子结点数为 n0，度=1 的结点数为 n1，度=2 的结点数为 n2，T 的分支数为 m。则

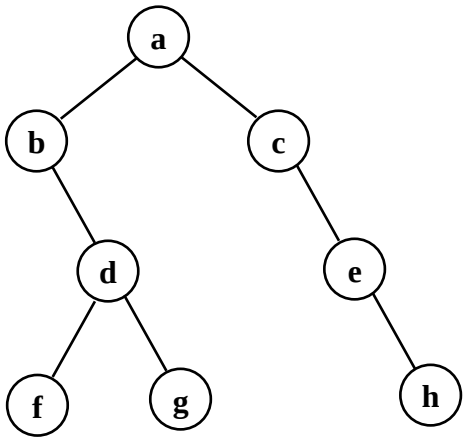
(1)由于二叉树中所有结点的度≤2，则有 n＝n0+n1+n2；

(2)除根结点外，其余结点都有一个分支对应(唯一前驱)，则有 n-1＝m；

(3)由于度= i (i = 0, 1, 2)的结点具有 i 个分支，则有 m＝0 + 1×n1+ 2×n2。

联合上面三个式子可推得 n0=n2+1。
证毕。

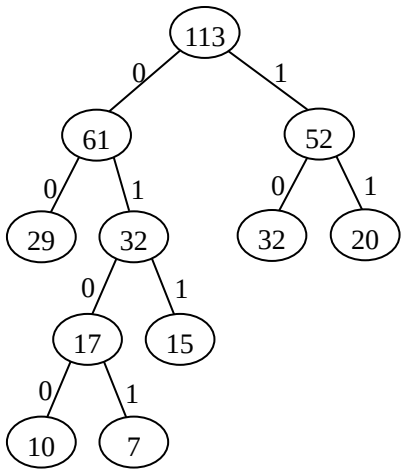
5. 已知二叉树 T 的先序遍历序列和中序遍历序列分别为 abdfgceh 和 bfdgaceh，试画出 T，并给出 T 的后序遍历序列。



T 的后序遍历序列为 fgdbheca。

6. 已知字符集 = { A, B, C, D, E, F }，它们的权值 = {10，32，15，29，20，7}，试求构造哈夫曼树和哈夫曼编码。

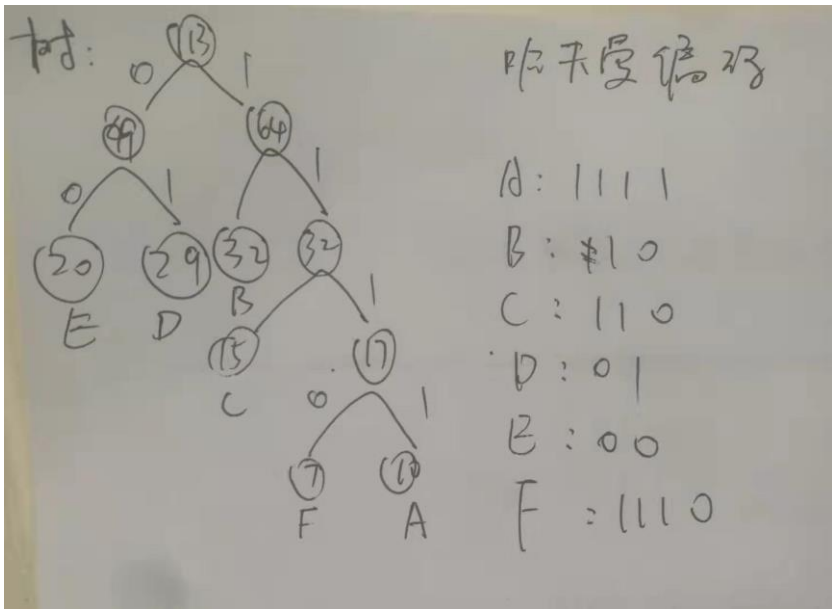
哈夫曼树和哈夫曼编码如下：



哈夫曼编码：
A：0100
B：10
C：011
D：00
E：11
F：0101

或者(两个 32 对换位置)
哈夫曼编码:
A：1000
B：01
C：101
D：00
E：11
F：1001

（或者按照从小到大的顺序）



二．**算法题**（含 2 个小题，每小题 10 分，计 20 分）

1．统计出单链表 HL 中结点的值等于给定值 X 的结点数。

int CountX(LNode* HL,ElemType x)

```
int CountX(LNode* HL,ElemType x)
    {   int i=0; LNode* p=HL;//i 为计数器
        while(p!=NULL)
          { if (P->data==x) i++;
            p=p->next;
           }//while, 出循环时 i 中的值即为 x 结点个数
        return i;
    }//CountX
```

2．设计二叉树的存储结构以及判断两个二叉树是否相同的算法。

[题目分析]先判断当前节点是否相等(需要处理为空、是否都为空、是否相等)，如果当前节点不相等，直接返回两棵树不相等;如果当前节点相等，那么就递归的判断他们的左右孩子是否相等。

[参考答案]

```
        typedef struct Node
        {  datatype data;
        struct Node *lchild,*rchild;
        } bitree;

        int judgebitree(bitree *bt1,bitree *bt2)
        {  if (bt1==0 && bt2==0) return(1);
        if (bt1==0 || bt2==0 ||bt1->data!=bt2->data) return(0);
        return(judgebitree(bt1->lchild,bt2->lchild)*judgebitree(bt1->rchild,bt2->rchild));
}
```