《数据结构与算法 B》随堂摸底测试（三）参考答案

学号_____ 姓名_____ 教师/教室_____

一、 选择题（单选或多选）

1 下列关于二叉树性质的说法正确的有（ ）。

A. 非空满二叉树（满二叉树每个结点有 0 个或 2 个孩子）的结点个数一定为奇数。
B. 当一棵完全二叉树是满二叉树时，叶子结点一定集中在最下面一层。
C. 完全二叉树最多只有最下面的一层结点度数可以小于 2。
D. 满二叉树的所有结点的度均为 2。

2 如果结点 A 有 3 个兄弟(不包括 A 本身)，B 是 A 的父结点，则 B 的度是（ ）。

A. 3
B. 4
C. 5
D. 1

3 已知某二叉树的后序遍历序列是 dabec，中序遍历序列是 debac，它的前序遍历序列是（ ）。

A. acbed
B. decab
C. deabc
D. cedba

4 下列关于二叉搜索/排序树（Binary Search Tree）的说法错误的有（ ）。

A. 二叉搜索树按照中序遍历将各结点打印出来，将得到按照由小到大的排列。
B. 二叉搜索树一定是满二叉树。
C. 当根结点没有左儿子时，根结点一定是二叉搜索树中值最小的结点。
D. 如果结点 x 的左儿子有右子树，则说明存在某个结点的值介于结点 x 的值和 x 左儿子的值之间，并且这个结点在 x 的左子树之中。

5 下列关于 Huffman 树和 Huffman 编码的说法错误有（ ）。

A. Huffman 树一定是扩充的二叉树。
B. 对于具有同样的一组权值的内容可以得到不同的 Huffman 编码方案。
C. Huffman 树一定是完全二叉树。
D. Huffman 编码中所有编码都是不等长的。

二、 **填空题**

1 在一棵深度（独根树深度为 0）为 K 的完全二叉树中，至少存在_____个结点。

2 使用 3 个结点 A,B,C，可以构造出_____ 个不同的二叉树（不是树型，是树）。

3 在一棵非空二叉树中，若度为 0 的结点的个数 n，度为 2 的结点个数为 m，则有 n=_____。

三、 算法填空题

（如果填空的思路跟自己不一致，可以自己写思路、完整的带注释算法框架）。

在下面算法中填写合适的语句（每个空缺可以填写 0 条或多条语句），使得算法功能完整。

一棵二叉树中所有叶结点的最大枝长和最小枝长分别定义如下：

1 所谓最大枝长就是二叉树层数（根结点为第 0 层）；

2 所谓最小枝长就是离根结点距离最近的叶结点到根结点的路径上的边数；

3 空树的最大最小枝长都定义为 -1；

4 结点数目 n =1（只有树根）的二叉树最大最小枝长都为 0，n=2 的二叉树最大最小枝长都为 1。

下面的这个算法，同时求出一棵二叉树所有叶结点的最大和最小枝长。

```
void FarNearLeaf(BinaryTreeNode *root, int &Far, int &Near)
{
      // 按照后序周游方式求树的最大/小枝长
       int rightFar, leftFar, rightNear, leftNear;
       if ( root == NULL )
       {
            Far=-1; Near=-1; return ;   // 空树
       }
       if ( (root->leftPtr==NULL) && (root->rightPtr==NULL) )
       {
            Far=0;   Near=0;   return;   // 独根/叶结点
       }
    FarNearLeaf (root->leftPtr, leftFar, leftNear);          // 递归求左子树最大/小枝长
    FarNearLeaf (root->rightPtr, rightFar, rightNear);     // 递归求右子树最大小枝长
    // 下面填空是求最大枝长，子根 root 的最大枝长是左右子树较大的最大枝长+1；
    if (leftFar > rightFar)
                    Far = leftFar + 1;
    else
       ____________________________________
    if (!root->leftPtr || !root->rightPtr)
    {   // 单支树，最小枝长为非空子树最小枝长+1
       if (root->leftPtr)
          ____________________________________
       else
          ____________________________________
    }
    else
    {   // 左右子树均存在，最小枝长为左右子树较小的最小枝长+1
      if (leftNear < rightNear)
         ____________________________________
      else
         ____________________________________
     }
}
```