30240184，2010 年 5 月 21 日 姓名：

2	0	0							

题号	1	2	3	4.1	4.2	4.3	4.4	∑
得分								

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Fib(n)	0	1	1	2	3	5	8	13	21	34	55	89	144	233	377	610	987	1597	2584
2^n	1	2	4	8	16	32	64	128	256	512	1024	2048	4096	8192					

第1题 正误判断 2 ×8

1. T F 在对二进制串做匹配时，采用 next[]表比采用 BC[]表通常效率更高。

2. T F 所有叶节点深度一致的有根二叉树，必为满树。

3. T F 完全二叉树的子树，也一定是完全二叉树。

4. T F 由合法的先序遍历序列和中序遍历序列，可以唯一确定一棵二叉树。

5. T F 在 Huffman 算法过程中，权重小的内部节点必然早于权重大的内部节点被创建。

6. T F 由同一组互异关键码，按不同次序逐个插入而生成的 BST 必互异。

7. T F BST 中新插入的节点，必是叶节点。

8. T F 在 AVL 树中删除节点之后若树高降低，则必然做过旋转调整。

第2题 多重选择 4 ×5

1. 【 】在（ ）中，越深的节点必然越多。

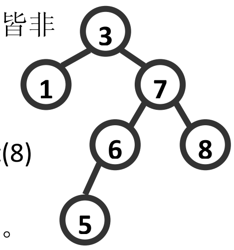
   A. 二叉树 B. AVL 树 C. 满二叉树 D. 完全二叉树 E. 以上皆非

2. 【 】在包含 2010 个节点的 AVL 树中，最高与最低叶节点之间的深度差最大可达（ ）。

   A. 8 B. 9 C. 10 D. 11 E. 以上皆非

3. 【 】由 6 个节点组成的二叉树，若中序遍历序列为 ABCDEF，则不可能的后序遍历序列是（ ）。

   A. CBEADF B. ADFECB C. ABDECF D. BDACFE E. 以上皆非



4. 【 】右图有可能是一棵刚做过 BST 的（ ）操作，但尚未旋转调整的 AVL 树。

   A. delete(2) B. insert(3) C. detele(4) D. insert(5) E. insert(8)

5. 【 】设 x 为某伸展树中的最大关键码，则在 find(x)过程中不可能实施（ ）调整。

   A. zig-zig（CW+CW） B. zig-zag C. zag-zig D. zag-zag E. 以上皆非

2	0	0							

第3题　填空　4 ×6

1. 由 2010 个节点组成的完全二叉树，共有（　　　）个叶节点。

2. 由 5 个互异节点组成、先序遍历序列与层次遍历序列相同的 BST，共有（　　　）棵。

3. 在由 2010 个节点组成的二叉树中，若单分支节点不超过 10 个，则对其做迭代式中序遍历时辅助栈的容量为（　　　）即足够。

4. 由 2010 节点组成的 AVL 树，最大高度可达（　　　）。

5. 在高度为 2010 的 AVL 树中删除一节点，至多可能造成（　　　）个节点失衡，至多需做（　　　）次旋转调整。

6. 高度为 3 的 5 阶 B-树，至多可存放（　　　）个关键码，至少需存放（　　　）个。

第4题　计算、理解与分析　10 ×4

1. 分别计算以下模式串的 next[ ]表、改进的 next[ ]表以及 BC[ ]表

j	0	1	2	3	4	5	6
Pattern[ ]	B	A	R	B	A	R	A
next[ ]	-1	0	0	0	1	2	3
改进的 next[ ]	-1	0	0	-1	0	0	3
BC[ ]				B		R	A

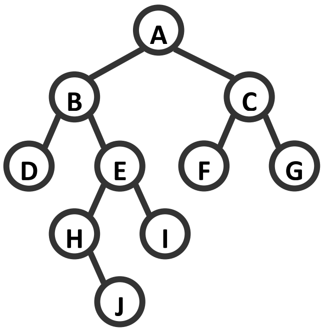
j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Pattern[ ]	T	A	R	S	O	M	E	_	T	A	T	A	R	S	U	S
next[ ]	-1	0	0	0	0	0	0	0	0	1	2	1	2	3	4	0
改进的 next[ ]	-1	0	0	0	0	0	0	0	-1	0	2	0	0	0	4	0
BC[ ]					O	M	E	_			T	A	R		U	S

2. 某二叉树有 A~G 共 7 个节点，其先序遍历、后序遍历序列的部分内容如下，试将其补全

先序遍历	E	C		B	D		F
后序遍历	B	A				G	

2	0	0							

3. 按如下算法遍历二叉树 T，试给出每次执行 PrintStack(S)的输出结果




```
StatusTraversal(Bintree T, Status (*Visit)(TElemType e)) {
  Stack* S = StackInit(-1);
  while (true) {
    GoAlongLeftBranch(S, T);  if (StackEmpty(S)) break;
    PrintStack(S);//输出栈S中的内容
    T = (Bintree) Pop(S); Visit(T->data); T = RChild(T);
  }
  StackDestroy(S);   return OK;
}
```

#	栈底 <--------------- PrintStack()输出的栈 S 内容 ---------------> 栈顶									
1										
2										
3										
4										
5										
6										
7										
8										
9										
10										
11										

4. 节点 x 的父节点和祖父节点分别记作 p 和 g。试在下图中补充尽可能少的节点以构造一棵 AVL 树，使得：1) 在摘除 x 之后，p 失衡；2) 经局部双旋调整之后，g 因失衡需再次实施双旋调整。
请同时在右侧画出最终恢复平衡的树形。