大连海事大学 2002 年硕士研究生招生考试试题

考试科目:数据结构与程序设计

适用专业:计算机应用技术

一、判断下列叙述是否正确：（共 20 分，每小题 1 分）

1、 二叉树是度为二的树的专业表述。

2、 先序遍历序列与中序遍历序列相同的二叉树只能是单枝树。

3、 一个森林与一棵二叉树的转换是唯一的。

4、 算法的优劣与所用计算机无关，但与算法描述语言有关。

5、 输入非法数据不会使健壮的算法出现莫名其妙的状态。

6、 在完成外排序的过程中，每个记录的 I/O 次数必定相等。

7、 任何广义表都可以有树结构表示。

8、 对森林的中序遍历等价于对应二叉树上的中序遍历。

9、 每一个网（带权网）的最小生成树的唯一的。

10、 在 AOE 网中，只要缩短关键路径上某个活动的时间，就可以缩短整个工程的时间。

11、 同一图的深度遍历序列与广度遍历序列不同。

12、 平衡二叉树中所有结点的平衡因子都不超过 1。

13、 当内存足够时，可以用哈希表实现比折半查找还快的查找算法。

14、 对任何序列进行排序，简单插入排序都比快速排序慢。

15、 哈希函数的时间复杂度应尽量小。

16、 简单插入排序和归并排序都是稳定的排序。

17、 在外排序中，即使工作区长度 w 远远小于 n，也有可能通过一次“置换—选择”排序完成 n 个初始序列记录的全部排序。

18、 进行外排序的速度取决于所选用的内排序算法的速度。

19、 有序表与线性表的区别在于是否按关键字排序。

20、 存在环路的有向图不能完全拓扑排序。

二、请选择准确的字或词填入空缺位置，构成正确完整的描述。（8 分）

1、 队列结构可以实现 “_____进_____出”的缓冲功能。

2、 由一棵树转换成的二叉树的根结点一定没有_____分枝。

3、 快速排序是简单排序算法中_____排序算法的改进。

4、 提高外排序速度的核心工作是减少记录的_____次数。

三、给定进栈元素顺序 a、b、c、d、e，请给出 b 为第二个出栈元素且 e 为第五个出栈元素的所有出栈序列？（8 分）

四、已知广义表 S=(((),a),(b,(c,(d)),e),(f)),gethead()表示取头元素操作，gettail（）表示取尾表操作，试分别求下列操作运算的结果：（8 分）

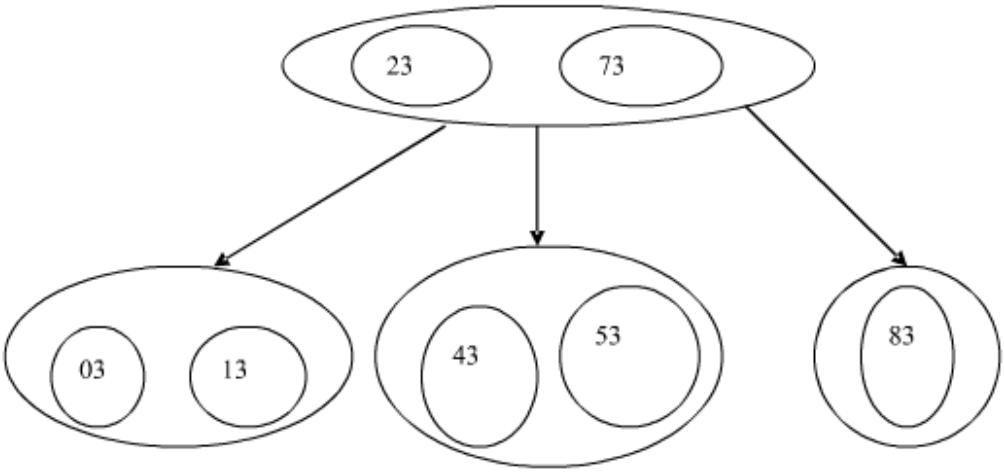
1、gethead（gethead（s））。

2、gettail（gettail（s））。

3、gethead（gettail（s））。

4、gettail （gethead（s））。

五、请在下面的 2-3 B 树上先依次插入关键字 11，31，再依次删除关键字 03，73，写出每次操作后 2-3 B 树的结构。（8 分）



六、选取哈希函数 H(key)=key mod 11，用线性再散列开放定址法解决冲突。试在 0~~10 的散列地址空间内对关键字序列{19，11，31，23，17，27，41，13，91，61}构造哈希表，并计算在等概率下成功查找的平均查找长度。（8 分）

七、试用归纳法证明：高度为 h 的二叉树的节点总数不超过 2^h-1。（10 分）

八、试读下列算法，请统计要完成 catch（9）所需要的比较次数、加法次数、除法次数和乘法次数。（10 分）

```
Void catch (int n) {
    Int k=0;
    Int m=1;
    While ((m+k)<n/2){
        M*=2; k++; n++;
    }
}
```

九、试读下列栈和队列操作算法，请给出调用并执行 testit（3）后的所有输出结果。（10 分）

```
Void testit (int m) {
    Stack s;
    Queue q;
    Int na, nb, nm;
    na=12; nb=21; nm=m;
    while (na<nb) {
```

```
            push (s,na); enqueue(q,nb);
            na++; nb-=2; nm++;
          }
       Printf ("nm=%d,na=%d,nb=%d\n",nm., na, nb) ;
       While (nm>0){
          Nm-=2;
          Na=pop(s);
          Nb=dequeue(q)+na;
          Printf("out:%d\n",nb) ;
       }
    }
```

十、以指向左侧二叉树根的指针 root 做初始值，执行右侧的递归算法 shifttree(root),请给出执行后的二叉树结构图。（10 分）

```
Shifttree ( T ) {
 If (T==NULL)
   Return (0) ;
 T1=T→lchild;
 Shifttree ( T1 );
 Tr=T→rchild;
 If ((T1<>null)&&(Tr<>null)&&(T1→data>Tr→data)) {
   T→Lchild=Tr;
   T→Rchild=T1;
 }
 Shifttree (Tr) ;
}
```