安徽大学20_20_—20_21_学年第_2_学期

《　　》（B卷）考试试题参考答案及评分标准

一、算法分析题（每小题5分，共20分）

1. 执行次数为$\sum_{i=1}^{n}\frac{i(i+1)}{2}=\frac{n(n+1)(n+2)}{6}$； 正确写出计算公式和简化结果给5分。

说明：正确写出计算式子，没有简化的，或者没有计算公式只有简化结果且正确的，均给3分。

2. 答案与评分标准如下：

（1）该算法功能是求元素递增有序排列的两个表L1和L2的交集，并将结果仍按递增有序另存入表C，同时保持表L1和L2不变；（3分）

（2）得到的表C=(4，5，19)。（2分）

3.（1）算法功能为：将串s中首次与串t匹配的子串逆置。　　（3分）

（2）算法执行完成后，s=〝helwolord〞。　　（2分）

4. 答案与评分标准如下：

（1）p==NULL （或者 !p ）　（1分）

（2）p->rchild!=NULL　　（2分）

（3）p　　（2分）

二、简答题（第5题7分，第6题8分，共15分）

5. 答案与评分标准如下：

（1）每个元素占32个二进位，主存字长16位，故每个元素占2个字长。数组A共有[9-(-1)+1]*[11-1+1]=11*11=121个元素。所以数组A共占有121*2=242个单元；　（3分）

（2）元素A[7，4]的起始地址为：S+{[7-(-1)]*(11-1+1)+(4-1)}*2=S+182　　（4分）

6. 答案见表1，填错一个扣0.5分：

表1

广义表	表长	表深	表头	表尾
A=( )	0	1	不存在	不存在
B=(a,( ))	2	2	a	(( ))
C=((a,b),c,((d)))	3	3	(a,b)	(c,((d)))
D=(a,D)=(a,(a,(a,…)))	2	∞	∞	(D)

三、应用题（第7题5分，其余每题10分，共35分）

7.（1） 哈夫曼树如下图1所示。　　（3分）

（2）以图1为例，左分支编“0”，右分支编“1”，编码为 ：　　（2分）

a: 011　　b: 10　　c: 00　　d: 010　　e: 11

说明：只要画出的树是符合哈夫曼树构造思想的，编码只要在此树基础上是前缀编码，均不扣分。

图1 第7题构造的哈夫曼树

8.答案与评分标准如下：

（1）排序结束条件为某一趟中没有发生元素交换。　　（2分）

（2）一共进行了8趟，每一趟结果如下：　　（8分）

初　始：　18　7　10　12　5　4　11　20　13

第一趟：　7　18　10　12　4　5　11　20　13

第二趟：　7　10　18　4　12　5　11　13　20

第三趟：　7　10　4　18　5　12　11　13　20

第四趟：　7　4　10　5　18　11　12　13　20

第五趟：　4　7　5　10　11　18　12　13　20

第六趟：　4　5　7　10　11　12　18　13　20

第七趟：　4　5　7　10　11　12　13　18　20

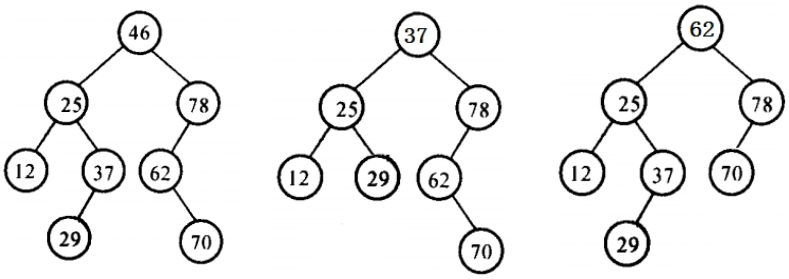
第八趟：　4　5　7　10　11　12　13　18　20

在第八趟中没有发生交换，说明所有元素已经有序排列，所以排序结束。

第（2）问的评分说明：完全正确得8分，否则按照每正确写出一趟给1分。

9. 答案及评分标准如下：

（1）二叉排序树如图2（a）所示：　（5分）



(a)　(b)　(c)

图2　第9题的各二叉排序树

（2）查找成功的平均查找长度为　（3分）

ASL=(1+2*2+3*3+2*4)/8=22/8=11/4=2.75

（3）删除46，有两种做法，写出其中任何一种即可：第一种以左子树上的最大关键字37和46交换，然后删除原37位置处结点；第二种以右子树上最小关键字62和46交换，然后删除原62位置处结点。分别如图（2）中(b)和(c)所示。　（2分）

10. 答案与评分标准如下：（1）表2分值3分；（2）表3分值4分；

表2　顶点事件的发生时间

顶点i	最早	最迟
1	0	0
2	3	4
3	2	2
4	6	6
5	6	7
6	8	8

表3　工序的开始时间

工序	最早	最迟
A	0	1
B	0	0
C	3	4
D	3	4
E	2	2
F	2	5
G	6	6
H	6	7

（3）关键路径BEG，该工程所需最短时间8个时间单位。　（3分）

四、算法设计题（每小题10分，共30分）

11. 要求尽量少移动元素，可以按照以下思路：

（1）从表头向后逐个检查数据元素，找到第一个满足删除条件的数据元素L.elem[i]；

（2）继续向后求出满足删除条件的数据元素个数j；

（3）所有被删除元素之后的元素一次移动到位。

说明: 1）只写出思路，没有算法描述的，给4分；

2）算法思路是每找到一个满足删除条件的，就删除（多次移动后续元素），描述正确的,也给满分;

3）其它情况,酌情给分.

```
void sq_dele(SqList &L,ElemType k1,ElemType k2)
 { // 在有序的顺序表L中删除其值在给定值k1与k2之间的所有元素
   i=0;
   while(i<L.length)  //找到第一个满足删除条件的数据元素
     if((L.elem[i]>=k1)&&(L.elem[i]<=k2))
           break;
     else i++;
   if(i<L.length)
    {   j=1; // j为计数器，条件满足条件的元素个数
      While((i+j<L.length)&&(L.elem[i+j]<=k2))
          j++;
      for(k=i+j;k<L.length;k++)  //移动元素
```

```
            L.elem[k-j]=L.elem[k];
        L.length=L.length-j;  //修改表长
      }
   } //sq_dele
```

12、（1）在入栈出栈的过程中，需要满足在入栈出栈序列的任一位置，入栈次数即I的个数必须大于或等于出栈次数即O的个数，而且由于栈的初态和终态均为空，故整个序列中I和O的个数应该相等。选项中满足这些条件的序列才是合法序列。经分析，A和D是合法序列，B和C是非法序列。　　（4分）

（2） 该题答案不唯一，算法描述能够正确判定序列合法或非法的，得6分；正确给出算法思想的给3分；不完全正确的，可酌情给分。

```
bool Judge(char A[ ])
{
      i=0;   // i为下标
      j=k=0;  //j,k分别为统计I和O个数的计数器
      while( A[i] !='\0')
      {
      switch(A[i])
      {
        case 'I' :j++ ;break ;  //入栈次数增1
        case 'O' :k++ ;  //出栈次数增1
        if(k>j) return false ;  //如果出栈次数多于入栈次数，则为非法
       }
       i++ ;  //i后移一位指向序列中下一个操作符
    }
    if(k !=j)return false ;  //操作序列结束，如果出栈次数和入栈次数不相同，则为非法
    else return true ;
}
```

13. 本题共10分,分值分配说明: (1)正确实现层次遍历得6分; (2)在(1)的基础上,输出结点的同时有表明其所在层次的,加2分；正确输出每层结点数目,也加2分.

参考答案如下(答案不唯一,其它解答请参考上面分值分配酌情给分):

算法思想:

用队列Q作为辅助结构，保存遍历过程中的各层结点。每次输出队头结点，访问之，若其左、右孩子不空，则入队。如此反复操作，直至队空，遍历结束。

另设整形变量K，保存遍历过程中某层的最后一个结点在队中的位置，level指示层次数。当队列的头指针front=K时，说明此时队头结点刚好是这层最后一个结点，则level=level+1，而此时尾指针rear正好是指示了下一层的最后一个结点。

```
        #define maxsize 50    //预定义队列的最大空间
       #define maxlevel 20   //预定义二叉树的最深层次数
       typedef struct bnode{
          ElemType data;
          Struct bnode *lchild,*rchild;
        }BNode,*BTree;  // 定义二叉树类型BTree
        Typedef struct{
          BTree elem[maxsize];
          int front,rear;
        }squeue;   // 定义顺序队列类型
      void  level_traver(BTree t)
      {
        squeue Q;    //定义辅助队列
        int C[maxlevel+1];  //定义辅助数组，C[i]统计第i层结点数，根为第一层
        if(t!=NULL){
          Q.rear=Q.front=0;  //初始化队列
          for(i=1;i<=maxlevel;i++)C[i]=0;  //初始化数组C
          Q.elem[Q.rear++]=t;  //根指针入队
          K=1; level=1;  //K记录当前层最后一个结点在队列中的位置
```

```
    while(Q.rear>Q.front)
     {   p=Q.elem[Q.front++];  //队头结点指针出队
        printf(p->data,level);  // 输出结点及其所在层次
        C[level]=C[level]+1;  //累计当前层次的结点数
        if(p->lchild!=NULL)
            Q.elem[Q.rear++]=p->lchild;  //当前结点左孩子入队
        if(p->rchild!=NULL)
            Q.elem[Q.rear++]=p->rchild;  //当前结点右孩子入队
        if(Q.front==K) //若条件满足，说明队头结点为当前层的最后一个结点
         {  level=level+1;
           K=rear;
         } //进入下一层，保留队尾指针于K中
      }
      for(i=1;i<level;i++)
         printf(I,C[i]);
    }
}
```