# 复旦大学计算机科学技术学院

## 《数据结构》期中考试试卷

## 共？页

**课程代码：COMP130004.01-03　考试形式：□开卷　□闭卷**　　2012 年 11 月

（本试卷答卷时间为 120 分钟，答案必须写在试卷上，做在草稿纸上无效）

专业_____学号_____姓名_____成绩_____

| 题号 | 一 | 二 | 三 | 总分 |
|---|---|---|---|---|
| 得分 | | | | |

（装订线内不要答题）

### 一、填空题（20%）

1、 假设字符串下标从 1 开始，模式串 P=”abaabcac”的 next 函数值（未优化）序列为_____。[答：-1、0、0、1、1、2、0、1]。

2、 使用一个 100 个元素的数组存储循环队列，如果采取少用一个元素空间的方法来区别循环队列的队空和队满，约定队头指针 front 等于队尾指针 rear 时表示队空。若为 front=8，rear=7，则队列中的元素个数为_____。[答：99]。

3、 高已知广义表 A=((a,b),(c,d))，假设求表头操作为 Head，求表尾操作为 Tail，则 Tail(Head(Tail(A)))=_____。[答：(d) ]。

4、 深度为 5 的二叉树至多有_____个结点。

5、 线索化二叉树的结点 t，如果 t->ltag==0，则前驱或者为_____，或者为当前结点左子树中序下的最后一个结点。 [答：NULL ]。

6、 设森林中有 4 棵树，树中结点的个数依次为 n1、n2、n3、n4，则把森林转换成二叉树后，其根结点的右子树上有_____个结点。[答：n2+n3+n4 ]

7、 具有 10 个叶子结点的二叉树中，度为 2 的结点有 _____个。[答：9]

8、 已知序列 25, 13, 10, 12, 9 是最大堆，在序列尾部插入新元素 18，将其再调整为最大堆，调整过程中元素之间进行的比较次数是。［答：　2 ］

9、 程序段{i=1;　　while (i<=n)　i=i*3;}的时间复杂度是：_____　。［答：　$O(\log_3 n)$ ］

10、设串的长度为 n，则它的子串个数为_____　　。［答：　n(n+1)/2 ］

### 二、问答题（30%）

1、 在表达式中，有的运算符要求从右到左计算，如 A^B^C 的计算次序应为(A^(B^C))，这在由中缀生成后缀的算法中是怎样实现的？以表达式 A^B^C 为例说明转换的过程。（6 分）

答：采用常规的中缀表达式转为后缀表达式的规则，不同之处是对运算符^优先级的规定。在算术运算中，先乘除后加减，先括号内后括号外，相同级别的运算符按从左到右的规则运算。而对^运算符，其优先级同常规理解，即高于加减乘除而小于左括号。为了适应本题中“从右到左计算”的要求，规定栈顶运算符^的级别小于正从表达式中读出的运算符^，即刚读出的运算符^级别高于栈顶运算符^，因此也入栈。

下面以 A**B**C 为例说明实现过程：读入 A，不是操作符，直接写入结果表达式。再读入^，与运算符栈顶比较（运算符栈顶初始化后，首先压入‘#’作为开始标志），其级别高于‘#’，入栈。再读入 B，直接进入结果表达式。接着读入^，与栈顶比较，均为^，由于后读入的^级别高于栈顶的^，因此^入栈。接着读入 C，直接到结果表达式。现在的结果（后缀）表达式是 ABC。最后读入‘#’，表示输入表达式结束，这时运算符栈中从栈顶到栈底有两个^和一个‘#’。两个运算符^退栈至结果表达式，结果表达式变为 ABC****。运算符栈中只剩‘#’，退栈，运算结束。

（装订线内不要答题）

2、 算法填空：下面给出了建立二叉树的算法，请阅读此算法并把缺失的语句补上（10 分）

```
typedef struct BinTreeNode {
      DataType data;
      struct BinTreeNode * leftChild, * rightChild;
   }

template <class Type>  class Stack {
    …
    public:
       Stack(){ … }       //构造函数
       ~Stack(){ … }
       int push(Type x){ … }
       Type*  pop(){ … }
       Type*  getTop(){ … }
       int makeEmpty(){ … }
}

void CreatBinTree(BinTreeNode *&BT,char ls){
  Stack<BinTreeNode*> s;
  s.makeEmpty();
  BT=NULL; //置空二叉树
  BinTreeNode *p;
  int k; istream ins(ls);   //把串 ls 定义为输入字符串流对象 ins ;
```

```
char ch; ins>>ch; //从 ins 顺序读入一个字符
while (ch != '#'){ //逐个字符处理，直到遇到'#'为止，假设输入的字符串流格式正确
  switch(ch){
    case '(': (1) ; k=1; break;
    case ')': s.pop(); break;
    case ',' : (2) ; break;
    default :
      p=new BinTreeNode;
      (3) ;
      p->leftChild=NULL;
      p->rightChild=NULL;
      if(BT==NULL) (4) ;
      else if (k==1) (s.getTop())->leftChild=p;
          else (s.getTop())->rightChild=p;
  }
  (5) ;
}
}
```

答：(1)s.push(p) (2)k=2 (3)p->data=ch (4)BT=p (5) ins>>ch

3、 试找出分别满足下列条件的所有二叉树：

⑴ 前序序列和中序序列相同。

⑵ 中序序列和后序序列相同。

⑶ 前序序列和后序序列相同。

（6 分）

答：

⑴ 空二叉树、只有一个根结点的二叉树和右斜树。

⑵ 空二叉树、只有一个根结点的二叉树和左斜树。

⑶ 空二叉树、只有一个根结点的二叉树

（装订线内不要答题）

4、 已知某完全 k 叉树只有度为 k 的结点及叶结点，设叶结点数为 $n_0$，求树的高度 h。（8 分）

答：显然可以算出树的结点数为 $n=（k*n_0-1）/(k-1)$

由于完全 k 叉树，$(k^{(h-1)}-1)/(k-1)<n<=(k^h-1)/(k-1)$

最后算出来 h 是以 k 为底对$(k*n_0)$取对数的上界，或者是以 k 为底对$(k*n_0)$取对数的下界加 1。

（装订线内不要答题）

## 三、算法设计题（50%）

**（以下题目，首先用简明的文字写出算法基本思路，然后给出数据结构和算法的 C 或者 C++描述，假设已经有现成的栈 Stack 和队列 Queue 的数据结构可以直接使用）**

1、编写算法计算给定二叉树中叶结点的个数。其中树结点定义如下

```
typedef struct BiTNode{
   DataType data;
   Struct BiTNode *LChild, * RChild;
}BiTNode, *BiTree;。
```

要求：（1）描述该算法的基本设计思想；（2）给出程序代码（10 分）

答：

(1)算法的基本思想:先序(或中序或后序)遍历二叉树，在遍历过程中查找叶子结点，并计数。由此，需在遍历算法中增添一个“计数”的参数，并将算法中“访问结点” 的操作改为:若是叶子，则计数器增 1。

(2)

```
void CountLeaf (BiTree T,    int& LeafNum)    {
    if ( T ) {
        if ((!T->lchild)&& (!T->rchild))    LeafNum ++;          // 对叶子结点计数
        else {
            CountLeaf( T->lchild, LeafNum); // 求左子树叶子数
            CountLeaf( T->rchild, LeafNum); //求右子树叶子数
        }
    }
} // CountLeaf
```

（装订线内不要答题）

2、 求递归函数 F(n)的非递归算法：（15 分）

$$F(n)=\begin{cases} n+1 & n=0 \\ n\cdot F(n/2) & n>0 \end{cases}$$

答：#include <iostream.h>

```
#define N    20
int main(){
      int I;       int a[N];       int n;
      cout<<"请输入 n:“;       cin>>n;
      for(i=0;i<n+1;i++){ if(i<1) a[i]=1;      else a[i]=i*a[i/2];       }
      cout<<a[n]<<endl;
      return 0;
}
```

3、求子数组的最大和：输入一个整形数组，数组里有正数也有负数。数组中连续的一个或多个整数组成一个子数组，每个子数组都有一个和。求所有子数组的和的最大值。要求时间复杂度为O(n)。例如输入的数组为 1, -2, 3, 10, -4, 7, 2, -5，和最大的子数组为 3, 10, -4, 7, 2，因此输出为该子数组的和 18。要求：（1）描述该算法的基本设计思想；（2）给出程序代码（15 分）

答：(1)采用贪心策略，从左向右扫描，记录当前的和 sum，当 sum < 0 时，把 sum 重置为 0。

(2)
```
int maxSubarray(int a[], int size) {
    if (size<=0) error(“error array size”);
    int sum = 0;
    int max = - (1 << 31);
    int cur = 0;
    while (cur < size) {
        sum += a[cur++];
        if (sum > max) {
            max = sum;
        } else if (sum < 0) {
            sum = 0;
        }
    } return max;
}
```

（装订线内不要答题）

4、从网络上接收类型为 DataType 的数据构成的串，每个串的长度事先不确定，但每个串最长不超过 max 个数据，接收到的数据放在一个单链表中，由于系统设计的限制，内存中的链表长度不能超过⌈max/2⌉，超过这个长度，则链表第一个结点的数据被丢弃，同时把最新接收的数据作为最后一个结点加入链表中，这些工作都由一个已经开发好的程序 A 完成。单链表结点结构为：

```
typedef struct    LNode{
        DataType data;
        Struct    LNode *link;
}。
```

请写出算法 balance(LNode *list) 判断每个串是否中心对称，例如 xyx, xyyx 都是中心对称，list 是由程序 A 提供的每次开始接收一个串时指向链表第一个结点的指针，限制 balance 的空间复杂度不超过 sizeof(DataType)*⌈max/2⌉，可以用 a1==a2 来判断 DataType 类型的数据 a1 和 a2 是否相等。要求：（1）描述 balance 算法的基本设计思想；（2）写出 balance 的程序代码。（10 分）

答：（1）首先要判断链表的中点，用 2 个指针 p 和 q，开始时都指向第一个结点，以后每次 p 前进 2 步，q 前进 1 步，p 到达链表末尾时，q 恰好在中点。其次是判断对称，可以用一个栈，q 每次经过的节点内容就入栈，p 到达链表末尾时，根据链表是偶数还是奇数长度调整 q 后，q 继续逐个结点向前，同时开始弹栈，每个弹出的结点与 q 经过的结点进行比较。

（2）

```
bool balance(LNode *list) {
        DataType stack[max];
        int top=0;
        LNode *p, *q;
        p = q = list;
        while(p) {
            p=p->link;
            if(p) {
                p=p->link;
                stack[top++] = q->data;
                q=q->link;
            }
            else q=q->link;   //跳过中间点
        }
        if(p==q) return true;     //空表
        if(top==0) return true;       //只有一个结点
        while(top>0) {
            if(stack[--top] == q->data) q=q->link;
            else return false;
        }
        return true;
    }
```

（装订线内不要答题）