数据结构1800例题与答案

第一章　绪　论

一、选择题（每小题2分）

1．算法的计算量的大小称为计算的（　　B　）。　【北京邮电大学 2000 二、3 （20/8分）】

A．效率　　B．复杂性　　C．现实性　　D．难度

2．算法的时间复杂度取决于（C）。　　【中科院计算所 1998 二、1 （2分）】

A．问题的规模　　B．待处理数据的初态　　C．A和B　　D．都不是

3．计算机算法指的是（① C　），它必须具备（②　B　） 这三个特性。

① A．计算方法　　B．排序方法

C．解决问题的步骤序列　　D．调度方法

② A．可执行性、可移植性、可扩充性　　B．可执行性、确定性、有穷性

C．确定性、有穷性、稳定性　　D．易读性、稳定性、安全性

【南京理工大学 1999 一、1（2分） 【武汉交通科技大学 1996 一、1（ 4分）】

4．一个算法应该是（ B　　）。【中山大学 1998 二、1（2分）】

A．程序　　B．问题求解步骤的描述

C．要满足五个基本特性　　D．A和C.

5．下面关于算法说法错误的是（　　D　）【南京理工大学 2000 一、1（1.5分）】

A．算法最终必须由计算机程序实现

B.为解决某问题的算法同为该问题编写的程序含义是相同的

C. 算法的可行性是指指令不能有二义性　　D. 以上几个都是错误的

6. 下面说法错误的是（ C　　）【南京理工大学 2000 一、2 （1.5分）】

(1）算法原地工作的含义是指不需要任何额外的辅助空间

（2）在相同的规模n下，复杂度O(n)的算法在时间上总是优于复杂度$O(2^n)$的算法

（3）所谓时间复杂度是指最坏情况下，估算算法执行时间的一个上界

（4）同一个算法，实现语言的级别越高，执行效率就越低

A．(1)　　B.(1),(2)　　C.(1),(4)　　D.(3)

7．从逻辑上可以把数据结构分为（　C　）两大类。【武汉交通科技大学 1996 一 、4（2分）】

A．动态结构、静态结构　　B．顺序结构、链式结构

C．线性结构、非线性结构　　D．初等结构、构造型结构

8．以下与数据的存储结构无关的术语是（　D　　）。【北方交通大学 2000 二、1（2分）】

A．循环队列　　B. 链表　　C. 哈希表　　D.　栈

9．以下数据结构中，哪一个是线性结构（　D　）？【北方交通大学 2001 一、1（2分）】

A．广义表　　B. 二叉树　　C. 稀疏矩阵　　D.　串

10．以下那一个术语与数据的存储结构无关？（　A　）【北方交通大学 2001 一、2（2分）】

A．栈　　B. 哈希表　　C. 线索树　　D.　双向链表

11．在下面的程序段中，对x的赋值语句的频度为（　C　）【北京工商大学 2001 一、10（3分）】

```
FOR i:=1　TO　n　DO
```

FOR j:=1 TO n DO
x:=x+1;

A． O(2n) B．O(n) C．$O(n^2)$ D．$O(\log_2^n)$

12．程序段 FOR i:=n-1 DOWNTO 1 DO
FOR j:=1 TO i DO
IF A[j]>A[j+1]
THEN A[j]与 A[j+1]对换；
其中 n 为正整数，则最后一行的语句频度在最坏情况下是（ D ）
A. O（n） B. O(nlogn) C. $O(n^3)$ D. $O(n^2)$ 【南京理工大学 1998 一、1(2 分)】

13．以下哪个数据结构不是多型数据类型（ D ）【中山大学 1999 一、3（1 分）】
A．栈 B．广义表 C．有向图 D．字符串

14．以下数据结构中，（ A ）是非线性数据结构【中山大学 1999 一、4】
A．树 B．字符串 C．队 D．栈

15. 下列数据中，（ C ）是非线性数据结构。【北京理工大学 2001 六、1（2 分）】
A．栈 B. 队列 C. 完全二叉树 D. 堆

16．连续存储设计时，存储单元的地址（ A ）。【中山大学 1999 一、1（1 分）】
A．一定连续 B．一定不连续 C．不一定连续 D．部分连续，部分不连续

17．以下属于逻辑结构的是（ C ）。【西安电子科技大学应用 2001 一、1】
A．顺序表 B. 哈希表 C.有序表 D. 单链表

二、判断题

1. 数据元素是数据的最小单位。(2)
【北京邮电大学 1998 一、1（2 分）】【青岛大学 2000 一、1 （1 分）】
【上海交通大学 1998 一、1】 【山东师范大学 2001 一、1 （2 分）】

2. 记录是数据处理的最小单位。 (2) 【上海海运学院 1998 一、5（1 分）】

3. 数据的逻辑结构是指数据的各数据项之间的逻辑关系；(2)【北京邮电大学 2002 一、1（1 分）】

4．算法的优劣与算法描述语言无关，但与所用计算机有关。(2)
【大连海事大学 2001 一、10（1 分）】

5．健壮的算法不会因非法的输入数据而出现莫名其妙的状态。(1)
【大连海事大学 2001 一、11（1 分）】

6．算法可以用不同的语言描述，如果用 C 语言或 PASCAL 语言等高级语言来描述，则算法实际上就是程序了。(2)【西安交通大学 1996 二、7（3 分）】

7．程序一定是算法。(2)【燕山大学 1998 二、2（2 分）并改错】

8．数据的物理结构是指数据在计算机内的实际存储形式。(1)【山东师范大学 2001 一、2（2 分）】

9. 数据结构的抽象操作的定义与具体实现有关。(2)【华南理工大学 2002 一、1（1 分）】

10. 在顺序存储结构中，有时也存储数据结构中元素之间的关系。(2)
【华南理工大学 2002 一、2 （1 分）】

11. 顺序存储方式的优点是存储密度大，且插入、删除运算效率高。(2)
【上海海运学院 1999 一、1（1 分）】

12. 数据结构的基本操作的设置的最重要的准则是，实现应用程序与存储结构的独立。(1)
【华南理工大学 2002 一、5（1 分）】

13. 数据的逻辑结构说明数据元素之间的顺序关系,它依赖于计算机的储存结构.(2)
【上海海运学院 1998 一、1（1 分）】

三、填空

1．数据的物理结构包括 数据元素 的表示和 数据元素间关系 的表示。【燕山大学 1998 一、1（2 分）】

2. 对于给定的 n 个元素,可以构造出的逻辑结构有 集合 ， 线性结构 ， 树形结构 ， 图状结构或网状结构 四种。
【中科院计算所 1999 二、1（4 分）】

3．数据的逻辑结构是指 数据的组织形式，即数据元素之间逻辑关系的总体。而逻辑关系是指数据元素之间的关联方式或程“邻接关系”【北京邮电大学 2001 二、1（2 分）】

4.一个数据结构在计算机中 表示(又称映像)称为存储结构。【华中理工大学 2000 一、1（1 分）】

5．抽象数据类型的定义仅取决于它的一组 逻辑特性 ，而与 在计算机内部如何表示和实现 无关，即不论其内部结构如何变化，只要它的 数学特性 不变，都不影响其外部使用。【山东大学 2001 三、3（2 分）】

6．数据结构中评价算法的两个重要指标是 算法的时间复杂度和空间复杂度 【北京理工大学 2001 七、1（2 分）】

7. 数据结构是研讨数据的 逻辑结构 和 物理结构 ，以及它们之间的相互关系，并对与这种结构定义相应的 操作(运算) ,设计出相应的算法 。【西安电子科技大学 1998 二、2（3 分）】

8. 一个算法具有 5 个特性: 有穷性 、 确定性、 可行性 ，有零个或多个输入、有一个或多个输出。
【华中理工大学 2000 一、2（5 分）】 【燕山大学 1998 一、2（5 分）】

9．已知如下程序段

```
FOR i:= n  DOWNTO  1  DO          {语句 1}
   BEGIN
      x:=x+1;                     {语句 2}
      FOR j:=n  DOWNTO  i  DO     {语句 3}
        y:=y+1;                   {语句 4}
   END;
```

语句 1 执行的频度为 n+1 ；语句 2 执行的频度为 n ；语句 3 执行的频度为 n(n+3)/2 ；语句 4 执行的频度为 n(n+1)/2 。【北方交通大学 1999 二、4（5 分）】

10．在下面的程序段中，对 x 的赋值语句的频度为 _1+(1+2++(1+2+3)+…+(1+2+3+…+n)=n(n+1)(n+2)/6 0（n 的立方）____（表示为 n 的函数）

```
FOR    i：＝1  TO  n  DO
 FOR    j：＝1   TO  i  DO
   FOR  k：＝1   TO  j  DO
        x：＝x＋delta;
```

【北京工业大学 1999 一、6（2 分）】

11.下面程序段中带下划线的语句的执行次数的数量级是：_____【合肥工业大学 1999 三、1（2 分）】

```
   i：=1;  WHILE i<n DO i：=i*2;
```

12. 下面程序段中带下划线的语句的执行次数的数量级是( )。【合肥工业大学 2000 三、1（2 分）】

```
i:=1;
WHILE i<n BEGIN  FOR j:=1 TO n DO x:=x+1;i:=i*2  END;
```

13. 下面程序段中带有下划线的语句的执行次数的数量级是(　　)【合肥工业大学 2001 三、1（2 分）】

```
i：=n*n    WHILE i<>1  DO i:=i div 2;
```

14. 计算机执行下面的语句时，语句 s 的执行次数为 _____ 。【南京理工大学 2000 二、1（1.5 分）】

```
FOR(i=1; i<n-1; i++)
 FOR(j=n;j>=i;j--)
 s;
```

15. 下面程序段的时间复杂度为_____。(n>1)

```
sum=1;
for (i=0;sum<n;i++) sum+=1;
```

【南京理工大学 2001 二、1（2 分）】

16. 设 m.n 均为自然数，m 可表示为一些不超过 n 的自然数之和，f(m,n)为这种表示方式的数目。例 f(5,3)=5，有 5 种表示方式：3+2，3+1+1，2+2+1，2+1+1+1，1+1+1+1+1。

①以下是该函数的程序段，请将未完成的部分填入，使之完整

```
int f(m,n)
   int m,n;
{ if(m==1)
        return  (1)  ;
if(n==1){
        return  (2)  ;}
if(m<n)
     {return f(m,m);}
if (m==n)
     {return 1+ (3)  ;}
return f(m.n-1)+f(m-n, (4)  );
}
```

②执行程序，f(6,4)=_____。【中科院软件所 1997 二、1 （9 分）】

17. 在有 n 个选手参加的单循环赛中，总共将进行_____场比赛。【合肥工业大学 1999 三、8（2 分）】

四、应用题

1. 数据结构是一门研究什么内容的学科？【燕山大学 1999 二、1 （4 分）】

2. 数据元素之间的关系在计算机中有几种表示方法？各有什么特点？【燕山大学 1999 二、2（4 分）】

3. 数据类型和抽象数据类型是如何定义的。二者有何相同和不同之处，抽象数据类型的主要特点是什么？使用抽象数据类型的主要好处是什么？【北京邮电大学 1994 一（8 分）】

4. 回答问题（每题 2 分）【山东工业大学 1997 一 （8 分）】

（1）在数据结构课程中，数据的逻辑结构，数据的存储结构及数据的运算之间存在着怎样的关系？

（2）若逻辑结构相同但存储结构不同，则为不同的数据结构。这样的说法对吗？举例说明之。

（3）在给定的逻辑结构及其存储表示上可以定义不同的运算集合，从而得到不同的数据结构。这样说法对吗？举例说明之。

（4）评价各种不同数据结构的标准是什么？

5．评价一个好的算法，您是从哪几方面来考虑的？

【大连海事大学 1996 二、3 （2 分)】【中山大学 1998 三、1 （5 分)】

6．解释和比较以下各组概念【华南师范大学 2000 一（10 分）】

（1）抽象数据类型及数据类型 （2）数据结构、逻辑结构、存储结构

（3）抽象数据类型【哈尔滨工业大学 2000 一、1（3 分)】

（4）算法的时间复杂性 【河海大学 1998 一、2（3 分)】

（5）算法【吉林工业大学 1999 一、1（2 分)】

（6）频度【吉林工业大学 1999 一、2（2 分)】

7. 根据数据元素之间的逻辑关系，一般有哪几类基本的数据结构？

【北京科技大学 1998 一、1】【同济大学 1998】

8．对于一个数据结构，一般包括哪三个方面的讨论？【北京科技大学 1999 一、1（2 分)】

9. 当你为解决某一问题而选择数据结构时，应从哪些方面考虑？【西安电子北京科技大学 2000】

10. 若将数据结构定义为一个二元组（D，R）,说明符号 D，R 应分别表示什么？

【北京科技大学 2001 一、1（2 分)】

11．数据结构与数据类型有什么区别？【哈尔滨工业大学 2001 三、1（3 分)】

12．数据的存储结构由哪四种基本的存储方法实现？【山东科技大学 2001 一、1（4 分)】

13．若有 100 个学生，每个学生有学号，姓名，平均成绩，采用什么样的数据结构最方便，写出这些结构？

【山东师范大学 1996 二、2（2 分)】

14. 运算是数据结构的一个重要方面。试举一例，说明两个数据结构的逻辑结构和存储方式完全相同，只是对于运算的定义不同。因而两个结构具有显著不同的特性，是两个不同的结构。

【北京大学 1998 一、1（5 分)】

15. 在编制管理通讯录的程序时，什么样的数据结构合适？为什么？【 长沙铁道学院 1998 四、3(6 分)】

16. 试举一例，说明对相同的逻辑结构,同一种运算在不同的存储方式下实现，其运算效率不同。

【北京理工大学 2000 三、1（4.5 分)】

17. 有实现同一功能的两个算法 A1 和 A2，其中 A1 的时间复杂度为 T1=O(2^n)，A2 的时间复杂度为 T2=O(n^2)，仅就时间复杂度而言，请具体分析这两个算法哪一个好。【北京航空航天大学 2000 二（10 分)】

18．设计一数据结构，用来表示某一银行储户的基本信息： 账号、姓名、开户年月日、储蓄类型、存入累加数、利息、帐面总数。【浙江大学 1994 一 、3（5 分）】

19. 写出下面算法中带标号语句的频度。

```
TYPE  ar=ARRAY[1..n] OF datatype;
PROCEDURE    perm  ( a: ar; k, n: integer);
  VAR   x: datatype;     i:integer;
  BEGIN
```

```
(1) IF k=n
    THEN BEGIN
         (2) FOR  i:=1  TO  n DO
         (3) write (a[i]);
             writeln;
         END
     ELSE BEGIN
         (4)  FOR   i:=k  TO  n  DO
         (5) a[i]:=a[i]+i*i;
         (6)  perm (a, k+1, n);
          END;
END;
```

设 k 的初值等于 1。

【北京邮电大学 1997 二（10 分）】

20. 分析下面程序段中循环语句的执行次数。

```
i:=0;s:=0;n:=100;
REPEAT
   i:=i+1;
   s:=s+10*i;
UNTIL  NOT((i<n) AND (s<n));
```

【北京邮电大学 1998 四、1（5 分）】

21. 下列算法对一 n 位二进制数加 1，假如无溢出，该算法的最坏时间复杂性是什么？并分析它的平均时间复杂性。

```
TYPE  num=ARRAY [1..n] of [0..1];
PROCEDURE  Inc  (VAR a：num);
VAR  i：integer;
BEGIN  i：=n;
   WHILE  A[i]=1  DO
      BEGIN  A[i]：=0;   i：=i-1; END;
END;
A[i]：=1;
END Inc;
```

【东南大学 1998 三 (8 分) 1994 二（15 分)】

22. 阅读下列算法，指出算法 A 的功能和时间复杂性

```
PROCEDURE  A (h,g:pointer);
(h,g 分别为单循环链表（single linked  circular list）中两个结点指针)
PROCEDURE  B(s,q:pointer);
    VAR p:pointer;
    BEGIN
        p:=s;
        WHILE p^.next<>q DO p:=p^.next;
        p^.next:=s;
    END;(of B)
BEGIN
```

```
        B(h,g);  B(g,h);
    END;(of A)
```

【东南大学 1999 二（10 分）】

23. 调用下列 C 函数 f(n)或 PASACAL 函数 f(n)　回答下列问题 ：

（1） 试指出 f(n)值的大小，并写出 f(n) 值的推导过程;

（2） 假定 n= 5，试指出 f(5)值的大小和执行 f(5)时的输出结果 。

C 函数：

```
int f(int  n)
{ int i,j, k,sum= 0;
 for(i=l; i<n+1;i++)
  {for(j=n;j>i-1; j--)
    for(k=1;k<j+1;k++ )
       sum++;
    printf("sum=%d\n",sum);
   }
 return (sum);
}
```

【华中理工大学 2000 六（10 分）】

24. 设 n 是偶数，试计算运行下列程序段后 m 的值并给出该程序段的时间复杂度。

```
m:=0;
FOR  i:=1  TO  n  DO
  FOR  j:=2*i  TO  n  DO
      m:=m+1;
```

【南京邮电大学 2000 一、1】

25. 有下列运行时间函数：

（1）$T_1(n)=1000$;　（2）$T_2(n)=n^2+1000n$;　（3）$T_3(n)=3n^3+100n^2+n+1$;

分别写出相应的大 0 表示的运算时间。

【吉林工业大学 1999 二（12 分）】

26. 试给出下面两个算法的运算时间。

（1）

```
for  i←1  to  n  do
  x ← x+1
 END
```

（2）

```
for i← 1  to  n  do
  for  j←1  to  n  do
     x← x+1
   end
end
```

【中科院自动化研究所 1995 二、2 （6 分）】

27. 斐波那契数列 F_n 定义如下

$F_0=0$，　$F_1=1$，　$F_n=F_{n-1}+F_{n-2}$，　n=2,3...

请就此斐波那契数列，回答下列问题。

(1) (7 分) 在递归计算 F_n 的时候，需要对较小的 F_{n-1}，F_{n-2}，…, F_1, F_0 精确计算多少次?

(2) (5 分) 如果用大 O 表示法,试给出递归计算 F_n 时递归函数的时间复杂度录多少?

【清华大学 2000 二（12 分）】

28. 将下列函数，按它们在 $n\to\infty$ 时的无穷大阶数，从小到大排序。

n, $n-n^3+7n^5$, nlogn, $2^{n/2}$, n^3, logn, $n^{1/2}+\log n$, $(3/2)^n$, ,n!, $n^2+\log n$

【中科院计算所 1995 】

第一章　绪论（答案）

三．填空题

1．数据元素　数据元素间关系　　2．集合　线性结构　树形结构　图状结构或网状结构。

3．数据的组织形式，即数据元素之间逻辑关系的总体。而逻辑关系是指数据元素之间的关联方式或称“邻接关系”。

4．表示（又称映像）。　5．(1) 逻辑特性　(2) 在计算机内部如何表示和实现　(3) 数学特性。

6．算法的时间复杂度和空间复杂度。7．(1) 逻辑结构 (2) 物理结构 (3) 操作（运算）(4) 算法。

8．(1) 有穷性　(2) 确定性　(3) 可行性。

9．(1) n+1　(2) n　(3) n(n+3)/2　(4) n(n+1)/2。

10．1+（1+2++（1+2+3）+…+（1+2+…+n）=n(n+1)(n+2)/6　$O(n^3)$

11．$\log_2 n$　　12．$n\log_2 n$　　13．$\log_2 n^2$　　14．(n+3)(n-2)/2　　15．O(n)

16．① (1)1　(2)1　(3)f(m,n-1)　(4)n　② 9　　17．n(n-1)/2

四．应用题

1．数据结构是一门研究在非数值计算的程序设计问题中，计算机的操作对象及对象间的关系和施加于对象的操作等的学科。

2．四种表示方法

（1）顺序存储方式。数据元素顺序存放，每个存储结点只含一个元素。存储位置反映数据元素间的逻辑关系。存储密度大，但有些操作（如插入、删除）效率较差。

（2）链式存储方式。每个存储结点除包含数据元素信息外还包含一组（至少一个）指针。指针反映数据元素间的逻辑关系。这种方式不要求存储空间连续，便于动态操作（如插入、删除等），但存储空间开销大（用于指针），另外不能折半查找等。

（3）索引存储方式。除数据元素存储在一地址连续的内存空间外，尚需建立一个索引表，索引表中索引指示存储结点的存储位置（下标）或存储区间端点（下标），兼有静态和动态特性。

（4）散列存储方式。通过散列函数和解决冲突的方法，将关键字散列在连续的有限的地址空间内，并将散列函数的值解释成关键字所在元素的存储地址，这种存储方式称为散列存储。其特点是存取速度快，只能按关键字随机存取，不能顺序存取，也不能折半存取。

3．数据类型是程序设计语言中的一个概念，它是一个值的集合和操作的集合。如 C 语言中的整型、实型、字符型等。整型值的范围（对具体机器都应有整数范围），其操作有加、减、乘、除、求余等。实际上数据类型是厂家提供给用户的已实现了的数据结构。“抽象数据类型（ADT）”指一个数学模型及定义在该模型上的一组操作。“抽象”的意义在于数据类型的数学抽象特性。抽象数据类型的定义仅取决于它的逻辑特性，而与其在计算机内部如何表示和实现无关。无论其内部结构如何变化，只要它的数学特性不变就不影响它的外部使用。抽象数据类型和数据类型实质上是一个概念。此外，抽象数据类型的范围更广，它已不

再局限于机器已定义和实现的数据类型，还包括用户在设计软件系统时自行定义的数据类型。使用抽象数据类型定义的软件模块含定义、表示和实现三部分，封装在一起，对用户透明（提供接口），而不必了解实现细节。抽象数据类型的出现使程序设计不再是“艺术”，而是向“科学”迈进了一步。

4.（1）数据的逻辑结构反映数据元素之间的逻辑关系（即数据元素之间的关联方式或“邻接关系”），数据的存储结构是数据结构在计算机中的表示，包括数据元素的表示及其关系的表示。数据的运算是对数据定义的一组操作，运算是定义在逻辑结构上的，和存储结构无关，而运算的实现则是依赖于存储结构。

（2）逻辑结构相同但存储不同，可以是不同的数据结构。例如，线性表的逻辑结构属于线性结构，采用顺序存储结构为顺序表，而采用链式存储结构称为线性链表。

（3）栈和队列的逻辑结构相同，其存储表示也可相同（顺序存储和链式存储），但由于其运算集合不同而成为不同的数据结构。

（4）数据结构的评价非常复杂，可以考虑两个方面，一是所选数据结构是否准确、完整的刻划了问题的基本特征；二是是否容易实现（如对数据分解是否恰当；逻辑结构的选择是否适合于运算的功能，是否有利于运算的实现；基本运算的选择是否恰当。）

5. 评价好的算法有四个方面。一是算法的正确性；二是算法的易读性；三是算法的健壮性；四是算法的时空效率（运行）。

6.（1）见上面题 3　（2）见上面题 4　（3）见上面题 3

（4）算法的时间复杂性是算法输入规模的函数。算法的输入规模或问题的规模是作为该算法输入的数据所含数据元素的数目，或与此数目有关的其它参数。有时考虑算法在最坏情况下的时间复杂度或平均时间复杂度。

（5）算法是对特定问题求解步骤的描述，是指令的有限序列，其中每一条指令表示一个或多个操作。算法具有五个重要特性：有穷性、确定性、可行性、输入和输出。

（6）频度。在分析算法时间复杂度时，有时需要估算基本操作的原操作，它是执行次数最多的一个操作，该操作重复执行的次数称为频度。

7. 集合、线性结构、树形结构、图形或网状结构。　　8. 逻辑结构、存储结构、操作（运算）。

9. 通常考虑算法所需要的存储空间量和算法所需要的时间量。后者又涉及到四方面：程序运行时所需输入的数据总量，对源程序进行编译所需时间，计算机执行每条指令所需时间和程序中指令重复执行的次数。

10. D 是数据元素的有限集合，S 是 D 上数据元素之间关系的有限集合。

11.“数据结构”这一术语有两种含义，一是作为一门课程的名称；二是作为一个科学的概念。作为科学概念，目前尚无公认定义，一般认为，讨论数据结构要包括三个方面，一是数据的逻辑结构，二是数据的存储结构，三是对数据进行的操作（运算）。而数据类型是值的集合和操作的集合，可以看作是已实现了的数据结构，后者是前者的一种简化情况。

12. 见上面题 2。

13. 将学号、姓名、平均成绩看成一个记录（元素，含三个数据项），将 100 个这样的记录存于数组中。因一般无增删操作，故宜采用顺序存储。

```
typedef  struct
  {int num;//学号
   char name[8];//姓名
   float score;/平均成绩
  }node;
 node student[100];
```

14. 见上面题 4（3）。

15. 应从两方面进行讨论：如通讯录较少变动（如城市私人电话号码），主要用于查询，以顺序存储较方便，既能顺序查找也可随机查找；若通讯录经常有增删操作，用链式存储结构较为合适，将每个人的情况作为一个元素（即一个结点存放一个人），设姓名作关键字，链表安排成有序表，这样可提高查询速度。

16. 线性表中的插入、删除操作，在顺序存储方式下平均移动近一半的元素，时间复杂度为 O（n）；而在链式存储方式下，插入和删除时间复杂度都是 O（1）。

17. 对算法 A1 和 A2 的时间复杂度 T1 和 T2 取对数，得 $n\log^2$ 和 $2\log^n$。显然，算法 A2 好于 A1。

18.
```
struct node
{int year,month,day; };
typedef struct
{int num;//帐号
 char name[8];//姓名
 struct node date;//开户年月日
 int tag;//储蓄类型，如：0- 零存，1- 一年定期……
 float put;//存入累加数;
 float interest;//利息
 float total;//帐面总数
}count;
```

19. (1)n　(2)n+1　(3)n　(4)(n+4)(n-1)/2　(5)(n+2)(n-1)/2　(6)n-1

这是一个递归调用，因 k 的初值为 1，由语句（6）知，每次调用 k 增 1，故第(1)语句执行 n 次。（2）是 FOR 循环语句，在满足(1)的条件下执行，该语句进入循环体(3)n 次，加上最后一次判断出界，故执行了 n+1 次。(4)也是循环语句，当 k=1 时判断 n+1 次（进入循环体(5)n 次），k=2 时判断 n 次，最后一次 k=n-1 时判断 3 次，故执行次数是（n+1）+n+…+3=(n+4)(n-1)/2 次。语句(5)是(4)的循环体，每次比(4)少一次判断，故执行次数是 n+(n-1)+…+2=(n+2)(n-1)/2 次。注意分析时，不要把(2)分析成 n 次，更不是 1 次。

20. 4 （这时 i=4， s=100） REPEAT 语句先执行循环体，后判断条件，直到条件为真时退出循环。

21. 算法在最好情况下，即二进制数的最后一位为零时，只作一次判断，未执行循环体，赋值语句 A[i]执行了一次；最坏情况出现在二进制数各位均为 1（最高位为零，因题目假设无溢出），这时循环体执行了 n-1 次，时间复杂度是 O(n)，循环体平均执行 n/2 次，时间复杂度仍是 O(n)。

22. 该算法功能是将原单循环链表分解成两个单循环链表：其一包括结点 h 到结点 g 的前驱结点；另一个包括结点 g 到结点 h 的前驱结点。时间复杂度是 O(n)。

23. 第一层 FOR 循环判断 n+1 次，往下执行 n 次，第二层 FOR 执行次数为(n+(n-1)+(n-2)+…+1)，第三层循环体受第一层循环和第二层循环的控制，其执行次数如下表：

i=	1	2	3	…	n
j=n	n	n	n	…	n
j=n-1	n-1	n-1	n-1	…	
…	…	…	…		
j=3	3	3			
j=2	2	2			
j=1	1				

执行次数为(1+2+…+n)+(2+3+…+n)+…+n=n*n(n+1)/2-n(n^2-1)/6。在 n=5 时，f(5)=55，执行过程中，输出结果为：sum=15,sum=29,sum=41,sum=50,sum=55（每个 sum= 占一行,为节省篇幅，这里省去换行）。

24．O(n^2),m 的值等于赋值语句 m:=m+1 的运行次数，其计算式为

25．(1)O(1)　(2)O(n^2)　(3)O(n^3)　　26．(1)O(n)　(2)O(n^2)

27．（1）由斐波那契数列的定义可得：

$$F_n=F_{n-1}+F_{n-2}$$
$$=2F_{n-2}+F_{n-3}$$
$$=3F_{n-3}+2F_{n-4}$$
$$=5F_{n-4}+3F_{n-5}$$
$$=8F_{n-5}+5F_{n-6}$$
$$\cdots\cdots$$
$$=pF_1+qF_0$$

设 F_m的执行次数为 B_m(m=0、1、2、…、n-1)，由以上等式可知，F_{n-1}被执行一次，即 B_{n-1}=1；F_{n-2}被执行两次，即 B_{n-2}=2；直至 F_1被执行 p 次、F_0被执行 q 次，即 B_1=p，B_0=q。B_m的执行次数为前两等式第一因式系数之和，即 $B_m=B_{m-1}+B_{m-2}$，再有 B_{n-1}=1 和 B_{n-2}=2，这也是一个斐波那契数列。可以解得：

$B_m=$　[(　　)$^{n-m+2}$-(　　)$^{n-m+2}$]　　(m=0,1,2,…,n-1)

（2）时间复杂度为 O(n)

28．从小到大排列为：logn, $n^{1/2}$+logn, n, nlogn, n^2+logn，n^3,　$n-n^3+7n^5$, $2^{n/2}$, $(3/2)^n$, n!,

第 2 章　　线性表

一　选择题

1．下述哪一条是顺序存储结构的优点？（　　）【北方交通大学 2001 一、4（2 分）】

A．存储密度大　B．插入运算方便　C．删除运算方便　D．可方便地用于各种逻辑结构的存储表示

2．下面关于线性表的叙述中，错误的是哪一个？（　　）【北方交通大学 2001 一、14（2 分）】

A．线性表采用顺序存储，必须占用一片连续的存储单元。

B．线性表采用顺序存储，便于进行插入和删除操作。

C．线性表采用链接存储，不必占用一片连续的存储单元。

D．线性表采用链接存储，便于插入和删除操作。

3．线性表是具有 n 个（　　）的有限序列（n>0）。【清华大学 1998 一、4（2 分）】

A．表元素　　B．字符　　C．数据元素　　D．数据项　　E．信息项

4.若某线性表最常用的操作是存取任一指定序号的元素和在最后进行插入和删除运算，则利用（　　）存储方式最节省时间。【哈尔滨工业大学 2001 二、1（2 分）】

A．顺序表　　B．双链表　　C．带头结点的双循环链表　　D．单循环链表

5．某线性表中最常用的操作是在最后一个元素之后插入一个元素和删除第一个元素，则采用（　　）存储方式最节省运算时间。【南开大学 2000 一、3】

A．单链表　　B．仅有头指针的单循环链表　　C．双链表　　D．仅有尾指针的单循环链表

6．设一个链表最常用的操作是在末尾插入结点和删除尾结点，则选用(　　)最节省时间。

A．单链表　B.单循环链表　C．带尾指针的单循环链表　D.带头结点的双循环链表

【合肥工业大学 2000 一、1（2 分）】

7．若某表最常用的操作是在最后一个结点之后插入一个结点或删除最后一个结点。则采用（　　）存储方式最节省运算时间。【北京理工大学 2000 一、1（2 分）】

A．单链表　　B．双链表　　C．单循环链表　　D．带头结点的双循环链表

8．静态链表中指针表示的是（　　）．【北京理工大学 2001 六、2（2 分）】

A．　内存地址　　B．数组下标　　C．下一元素地址　　D．左、右孩子地址

9．链表不具有的特点是（　　）【福州大学 1998 一、8（2 分）】

A．插入、删除不需要移动元素　B．可随机访问任一元素

C．不必事先估计存储空间　D．所需空间与线性长度成正比

10．下面的叙述不正确的是（　　）【南京理工大学 1996 一、10（2 分）】

A．线性表在链式存储时，查找第 i 个元素的时间同 i 的值成正比

B．线性表在链式存储时，查找第 i 个元素的时间同 i 的值无关

C．线性表在顺序存储时，查找第 i 个元素的时间同 i 的值成正比

D．线性表在顺序存储时，查找第 i 个元素的时间同 i 的值无关

12.(1) 静态链表既有顺序存储的优点，又有动态链表的优点。所以，它存取表中第 i 个元素的时间与 i 无关。

(2) 静态链表中能容纳的元素个数的最大数在表定义时就确定了，以后不能增加。

(3) 静态链表与动态链表在元素的插入、删除上类似，不需做元素的移动。

以上错误的是（　　）【南京理工大学 2000 一、3（1.5 分）】

A．(1)，(2)　　B．(1)　　C．(1)，(2)，(3)　　D．(2)

13. 若长度为 n 的线性表采用顺序存储结构，在其第 i 个位置插入一个新元素的算法的时间复杂度为（　　）(1<=i<=n+1)。【北京航空航天大学 1999 一、1（2 分）】

A. O(0)　　B. O(1)　　C. O(n)　　D. $O(n^2)$

14. 对于顺序存储的线性表，访问结点和增加、删除结点的时间复杂度为（　　）。

A. O(n)　O(n)　　B. O(n)　O(1)　　C. O(1)　O(n)　　D. O(1) O(1)

【青岛大学 2000 五、1（2 分）】

15. 线性表（ a1,a2,…,an）以链接方式存储时，访问第 i 位置元素的时间复杂性为（　　）

A. O（i）　　B. O（1）　　C. O（n）　　D. O（i-1）【中山大学 1999 一、2】

16. 非空的循环单链表 head 的尾结点 p↑满足（　　）。【武汉大学 2000 二、10】

A. p↑.link=head　　B. p↑.link=NIL　　C. p=NIL　　D. p= head

17. 循环链表 H 的尾结点 P 的特点是（　　）。【中山大学 1998 二、2（2 分）】

A. P^.NEXT:=H　　B. P^.NEXT:= H^.NEXT　　C. P:=H　　D. P:=H^.NEXT

18. 在一个以 h 为头的单循环链中，p 指针指向链尾的条件是（）【南京理工大学 1998 一、15（2 分）】

A. p^.next=h　　B. p^.next=NIL　C. p^.next.^next=h　　D. p^.data=-1

19. 完成在双循环链表结点 p 之后插入 s 的操作是（　　）;【北方交通大学 1999 一、4（3 分）】

A.　p^.next:=s ; s^.priou:=p; p^.next^.priou:=s ; s^.next:=p^.next;

B.　p^.next^.priou:=s; p^.next:=s; s^.priou:=p; s^.next:=p^.next;

C.　s^.priou:=p; s^.next:=p^.next; p^.next:=s; p^.next^.priou:=s ;

D.　s^.priou:=p; s^.next:=p^.next; p^.next^.priou:=s ; p^.next:=s;

20.在双向循环链表中,在 p 指针所指向的结点前插入一个指针 q 所指向的新结点,其修改指针的操作是(　　)。【北京邮电大学 1998 二、2（2 分）】

注:双向链表的结点结构为(llink,data,rlink)。 供选择的答案:

A.　p↑.llink:=q;　q↑.rlink:=p;　　p↑.llink↑.rlink:=q;　q↑.llink:=q;

B. p↑.llink:=q; p↑.llink↑.rlink:=q ; q↑.rlink:=　p;　　q↑.llink:=p↑.llink;

C.　q↑.rlink:=p;　q↑.llink:=p↑.llink;　p↑.llink↑.rlink:=q;　p↑.llink:=q;

D.　q↑.llink:=p↑.llink; q↑.rlink:=p;　p↑.llink:=q; p↑.llink:=q;（编者按：原题如此）

21. 在非空双向循环链表中 q 所指的结点前插入一个由 p 所指的链结点的过程依次为:

rlink(p) ← q;　llink(p) ← llink(q);　llink(q) ← p;　(　　)

A. rlink(q) ← p　B. rlink(llink(q)) ← p　C. rlink(llink(p)) ← p　D. rlink(rlink(p)) ← p

【北京航空航天大学 2000 一、1（2 分）】

22.　双向链表中有两个指针域，llink 和 rlink，分别指回前驱及后继，设 p 指向链表中的一个结点，q 指向一待插入结点，现要求在 p 前插入 q，则正确的插入为（　）【南京理工大学 1996 一、1（2 分）】

A.　p ^ .llink:=q;　q ^ .rlink:=p;　p ^ .llink ^ .rlink:=q; q^.llink:=p^.llink;

B.　q ^ .llink:=p^.llink;　p ^ .llink^.rlink:=q;　q ^ .rlink:=p;

p^.llink:=q^.rlink;

C. q^.rlink:=p; p^.rlink:=q; p^.llink^.rlink:=q; q^.rlink:=p;

D. p^.llink^.rlink:=q; q^.rlink:=p; q^.llink:=p^.llink; p^.llink:=q;

23. 在双向链表指针 p 的结点前插入一个指针 q 的结点操作是(　　)。【青岛大学 2000 五、2（2 分）】

A. p->Llink=q;q->Rlink=p;p->Llink->Rlink=q;q->Llink=q;

B. p->Llink=q;p->Llink->Rlink=q;q->Rlink=p;q->Llink=p->Llink;

C. q->Rlink=p;q->Llink=p->Llink;p->Llink->Rlink=q;p->Llink=q;

D. q->Llink=p->Llink;q->Rlink=q;p->Llink=q;p->Llink=q;

24. 在单链表指针为 p 的结点之后插入指针为 s 的结点，正确的操作是：(　　)。

A. p->next=s;s->next=p->next; B. s->next=p->next;p->next=s;

C. p->next=s;p->next=s->next; D. p->next=s->next;p->next=s;

【青岛大学 2001 五、3（2 分）】

25. 对于一个头指针为 head 的带头结点的单链表，判定该表为空表的条件是（　　）

A. head==NULL B. head→next==NULL C. head→next==head D. head!=NULL

【北京工商大学 2001 一、5（3 分）】

26. 在双向链表存储结构中，删除 p 所指的结点时须修改指针（　　）。

A. (p^.llink)^.rlink:=p^.rlink (p^.rlink)^.llink:=p^.llink;

B. p^.llink:=(p^.llink)^.llink (p^.llink)^.rlink:=p;

C. (p^.rlink)^.llink:=p p^.rlink:=(p^.rlink)^.rlink

D. p^.rlink:=(p^.llink)^.llink p^.llink:=(p^.rlink)^.rlink;

【西安电子科技大学 1998 一、1（2 分）】

27. 双向链表中有两个指针域，llink 和 rlink 分别指向前趋及后继，设 p 指向链表中的一个结点，现要求删去 p 所指结点，则正确的删除是（　　）(链中结点数大于 2，p 不是第一个结点)

A. p^.llink^.rlink:=p^.llink; p^.llink^.rlink:=p^.rlink; dispose(p);

B. dispose(p); p^.llink^.rlink:=p^.llink; p^.llink^,rlink:=p^.rlink;

C. p^.llink^.rlink:=p^.llink; dispose(p); p^.llink^.rlink:=p^.rlink;

D. 以上 A，B，C 都不对。 【南京理工大学 1997 一、1（2 分）】

二、判断

1. 链表中的头结点仅起到标识的作用。(　　)【南京航空航天大学 1997 一、1（1 分）】

2. 顺序存储结构的主要缺点是不利于插入或删除操作。(　)【南京航空航天大学 1997 一、2（1 分）】

3. 线性表采用链表存储时，结点和结点内部的存储空间可以是不连续的。(　　)

【北京邮电大学 1998 一、2（2 分）】

4. 顺序存储方式插入和删除时效率太低，因此它不如链式存储方式好。(　　)

【北京邮电大学 2002 一、2（1 分）】

5. 对任何数据结构链式存储结构一定优于顺序存储结构。(　)【南京航空航天大学 1997 一、3（1 分）】

6. 顺序存储方式只能用于存储线性结构。(　　)

【中科院软件所 1999 六、1-2（2 分）】【上海海运学院 1997 一、1（1 分）】

7. 集合与线性表的区别在于是否按关键字排序。(　　)【大连海事大学 2001 一、5（1 分）】

8. 所谓静态链表就是一直不发生变化的链表。（　　）【合肥工业大学 2000 二、1（1 分）】
9. 线性表的特点是每个元素都有一个前驱和一个后继。（　　）【合肥工业大学 2001 二、1（1 分）】
10. 取线性表的第 i 个元素的时间同 i 的大小有关.（　　）【南京理工大学 1997 二、9（2 分）】
11. 循环链表不是线性表.（　　）【南京理工大学 1998 二、1（2 分）】
12. 线性表只能用顺序存储结构实现。（　　）【青岛大学 2001 四、2（1 分）】
13. 线性表就是顺序存储的表。（　　）【青岛大学 2002 一、1（1 分）】
14. 为了很方便的插入和删除数据，可以使用双向链表存放数据。（　　）
【上海海运学院 1995 一、1（1 分）】　【上海海运学院 1997 一、2（1 分）】
15. 顺序存储方式的优点是存储密度大，且插入、删除运算效率高。（　　）
【上海海运学院 1996 一、1（1 分）】　【上海海运学院 1999 一、1（1 分）】
16. 链表是采用链式存储结构的线性表,进行插入、删除操作时，在链表中比在顺序存储结构中效率高。（　　）　【上海海运学院 1998 一、2（1 分）】

三、填空

1. 当线性表的元素总数基本稳定，且很少进行插入和删除操作，但要求以最快的速度存取线性表中的元素时，应采用_____存储结构。【北方交通大学 2001 二、4】
2. 线性表 L=（a1,a2,…,an）用数组表示，假定删除表中任一元素的概率相同，则删除一个元素平均需要移动元素的个数是_____。【北方交通大学 2001 二、9】
3. 设单链表的结点结构为(data,next)，next 为指针域，已知指针 px 指向单链表中 data 为 x 的结点，指针 py 指向 data 为 y 的新结点 ，若将结点 y 插入结点 x 之后，则需要执行以下语句:_____；　_____;【华中理工大学 2000 一、4（2 分）】
4. 在一个长度为 n 的顺序表中第 i 个元素（1<=i<=n）之前插入一个元素时，需向后移动_____个元素。
【北京工商大学 2001 二、4（4 分）】
5. 在单链表中设置头结点的作用是_____。【哈尔滨工业大学 2000 二、1（1 分）】
6. 对于一个具有 n 个结点的单链表，在已知的结点*p 后插入一个新结点的时间复杂度为_____,在给定值为 x 的结点后插入一个新结点的时间复杂度为_____。【哈尔滨工业大学 2001 一、1（2 分）】
7. 根据线性表的链式存储结构中每一个结点包含的指针个数，将线性链表分成_____和_____；而又根据指针的连接方式，链表又可分成_____和_____。【西安电子科技大学 1998 二、4（3 分）】
8. 在双向循环链表中,向 p 所指的结点之后插入指针 f 所指的结点，其操作是_____、_____、_____、_____。【中国矿业大学 2000 一、1（3 分）】
9. 在双向链表结构中，若要求在 p 指针所指的结点之前插入指针为 s 所指的结点，则需执行下列语句：
s^.next:=p; s^.prior:= _____; p^.prior:=s; _____:=s;
【福州大学 1998 二、7 (2 分)】
10. 链接存储的特点是利用_____来表示数据元素之间的逻辑关系。【中山大学 1998 一、1 （1 分）】
11. 顺序存储结构是通过_____表示元素之间的关系的;链式存储结构是通过_____表示元素之间的关系的。【北京理工大学 2001 七、2 （2 分）】
12. 对于双向链表,在两个结点之间插入一个新结点需修改的指针共 _____个，单链表为

_____个。

【南京理工大学 2000 二、2 （3 分)】

13. 循环单链表的最大优点是：_____。【福州大学 1998 二、3 (2 分)】

14. 已知指针 p 指向单链表 L 中的某结点，则删除其后继结点的语句是：_____

【合肥工业大学 1999 三、2 （2 分)】

15. 带头结点的双循环链表 L 中只有一个元素结点的条件是：_____

【合肥工业大学 1999 三、3　2000 三、2（2 分)】

16. 在单链表 L 中，指针 p 所指结点有后继结点的条件是：____【合肥工业大学 2001 三、3 （2 分)】

17. 带头结点的双循环链表 L 为空表的条件是：_____。

【北京理工大学 2000 二、1 （2 分)】【青岛大学 2002 三、1 （2 分)】

18. 在单链表 p 结点之后插入 s 结点的操作是：_____。【青岛大学 2002 三、2（2 分)】

19. 请在下列算法的横线上填入适当的语句。【清华大学 1994 五 （15 分)】

```
FUNCTION  inclusion(ha,hb:linklisttp):boolean;
{以 ha 和 hb 为头指针的单链表分别表示有序表 A 和 B，本算法判别表 A 是否包含在表 B 内，若是，则返回“true”，否则返回“false”}
BEGIN
  pa:=ha^.next; pb:=hb^.next; (1)      ;
  WHILE (2)      DO
     IF pa^.data=pb^.data  THEN (3)      ELSE (4)      ;
  (5)
  END;
```

20. 完善算法：已知单链表结点类型为：

```
TYPE  ptr=^node;
     node=RECORD
           data：integer;   next：ptr
          END;
```

过程 create 建立以 head 为头指针的单链表。

```
PROCEDURE  create ( (1) );
VAR p,q：ptr;    k：integer;
BEGIN
   new (head);   q:=head; read (k);
    WHILE k>0 DO
       BEGIN
          (2); (3); (4); (5);
          read (k)
       END;
    q^.next：=NIL;
END;
```

【北京师范大学 1999 三】

21. 已给如下关于单链表的类型说明：

```
TYPE
  list=^node ;
  node=RECORD
       data:  integer;   next:  list;
```

```
        END;
```

以下程序采用链表合并的方法，将两个已排序的单链表合并成一个链表而不改变其排序性（升序），这里两链表的头指针分别为 p 和 q.

```
PROCEDURE mergelink(VAR p,q:list):
  VAR h,r: list;
  BEGIN
    (1)_____
    h^.next:= NIL; r:=h;
    WHILE((p<>NIL) AND (q<>NIL))   DO
      IF (p^.data<=q^.data)
        THEN  BEGIN   (2)___;  r:=p; p:=p^.next;   END
        ELSE  BEGIN  (3)____;  r:=q; q:=q^.next;   END;
    IF (p=NIL)  THEN  r^.next:=q;
      (4)__;
     p:=h^.next; dispose(h);
 END;【厦门大学 2000 三、2 （8 分）】
```

22. 假设链表 p 和链表 q 中的结点值都是整数,且按结点值的递增次序链接起来的带表头结点的环形链表。各链表的表头结点的值为 max,且链表中其他结点的值都小于 max，在程序中取 max 为 9999。在各个链表中，每个结点的值各不相同，但链表 p 和链表 q 可能有值相同的结点（表头结点除外）。下面的程序将链表 q 合并到链表 p 中，使得合并后的链表是按结点值递增次序链接起来的带表头结点的环形链表，且链表中各个结点的值各不相同。请在划线处填上适当内容，每个框只填一个语句或一个表达式，链表的结点类型如下

```
TYPE  nodeptr=^nodetype;
      nodetype=RECORD
                  data：integer;  link：nodeptr;
                END;
CONST max=9999;
PROCEDURE  merge(VAR p:nodeptr;q:nodeptr);
VAR r,s: nodeptr;
BEGIN
  r:=p;
  WHILE  (A)___  DO
     BEGIN
       WHILE  r^.link^.data<q^.link^.data  DO   (B)___;
       IF  r^.link^.data>q^.link^.data
          THEN BEGIN s:=(C)_; (D)_:=s^.link; s^.link:=(E)_;  (F)__:=s; (G)_;
END
          ELSE BEGIN  (H)__; s:=q^.link;  (I)__; dispose(s)  END
       END;
       dispose(q)
  END;【复旦大学 1997 五（18 分）】
```

23. PROC ins__linklist(la:linkisttp; i:integer; b:elemtp);

{la 为指向带头结点的单链表的头指针，本算法在表中第 i 个元素之前插入元素 b}

p:=(1)_____; j:=(2)_____;{指针初始化，j 为计数器}

```
WHILE (p<>NIL)  AND  ((3)    ) DO  [p:=(4)        ;  j:=j+1; ]
{寻找第 i-1 个结点}
IF (p=NIL)  OR  ((5)      )
  THEN error ( 'No this position' )
  ELSE [new(s) ;  s↑.data:=b; s↑.next:=p↑.next; p↑.next:=s;]
ENDP;{ins-linklist}
```
【燕山大学 1998 四、1（15 分）】

24. 已知双链表中结点的类型定义为：

```
TYPE dpointer=^list;
     list=RECORD
            data:integer; left,right:dpointer;
          END;
```

如下过程将在双链表第 i 个结点（i>=0）之后插入一个元素为 x 的结点，请在答案栏给出题目中_____处应填入的语句或表达式，使之可以实现上述功能。

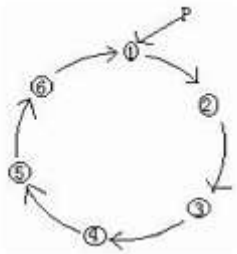
```
PROCEDURE  insert(VAR head:dpointer;i,x:integer);
VAR s,p:dpointer;   j: integer;
BEGIN
  new(s); s^.data:=x;
  IF(i=0)THEN BEGIN s^.right:=head; (1)    head:=s END{如果 i=0,则将 s 结点插入到表头后返回}
        ELSE BEGIN p:=head; (2)      ; {在双链表中查找第 i 个结点,由 p 所指向}
             WHILE ((p<>NIL) AND (j<i)) DO  BEGIN  j:=j+1; (3)   END;
             IF p<>NIL THEN
                IF  (p^.right=NIL)
                   THEN BEGIN p^.right:=s; s^.right:=NIL; (4)   END
                   ELSE BEGIN s^.right:=p^.right; (5)  ;p^.right:=s; (6)  END
             ELSE  writeln( 'can not find node!' )
            END
END;
```
【厦门大学 2002 二 （12 分）】

25. 阅读以下算法，填充空格，使其成为完整的算法。其功能是在一个非递减的顺序存储线性表中，删除所有值相等的多余元素。

```
CONST  maxlen=30
 TYPE sqlisttp=RECORD
                 elem:ARRAY[1..maxlen] OF integer;
                 last:0..maxlen
               END;
 PROC exam21(VAR L:sqlisttp);
   j:=1;  i:=2;
   WHILE (1)      DO
      [ IF  L.elem[i]<>L.elem[j]  THEN [ (2)      ; (3)      ];
        i:=i+1 ]
   (4)      ;
 ENDP;
```
【同济大学 2000 二、1 (10 分)】

26. 在本题的程序中，函数过程 Create_link_list(n)建立一个具有 n 个结点的环形链表；程序过程 josephus(n,i,m)对由 Create_link_list(n)所建立的具有 n 个结点的环形链表按

一定的次序逐个输出并删除链表中的所有结点，参数 n(n>0)指明环形链表的结点个数，参数 i(1<=i<=n)指明起始结点，参数 m (m>0)是步长，指明从起始结点或前次被删除并输出的结点之后的第 m 个结点作为本次被输出并删除的结点。例如，对于下图中具有 6 个结点的环形链



表，在调用 josephus(6,3,2)后，将输出 5,1,3,6,4,2 请在横线处填上适当内容，每空只填一个语句。

```
TYPE nodeptr=^nodetype;
    nodetype=RECORD
                data: intrger; link: nodeptr
             END;
VAR n,i,m: integer;
FUNCTION Create_link_list(n: integer): nodeptr;
 VAR head,p,q: nodeptr; i:integer;
 BEGIN head := NIL;
   IF n>0 THEN
         BEGIN new(head); p: =head;
               FOR i:=1 TO n-1 DO
                  BEGIN  p^.data:=i;  new(q); (A)____; (B)____ END;
               p^.data:=n;  (C)___;
         END;
   Creat_link_list:=head
 END;
PROCEDURE  josephus(n,i,m:integer);
  VAR p,q:nodeptr; j:integer;
  BEGIN p:=Creat_link_list(n);
        WHILE i>1 DO BEGIN p:=p^.link;  i:=i-1  END;
        (D)____ ;
        WHILE  j<n  DO
         BEGIN
           FOR i:=1  TO  m-1  DO  p:=p^.link;
           (E)___; write(q^.data:8);  (F)__ ;
           dispose(q);  j:=j+1
         END
END;【复旦大学 1997 四（12 分）】
```

27. 对于给定的线性链表 head，下面的程序过程实现了按结点值非降次序输出链表中的所有结点，在每次输出一个结点时，就把刚输出的结点从链表中删去。请在划线处填上适当的内容，使之成为一个完整的程序过程，每个空框只填一个语句。

```
TYPE   nodeptr =^ nodetype;
       nodetype = RECORD
                    data : integer; link : nodeptr
                  END;
VAR  head : nodeptr;
PROCEDURE  sort_output_delete (head : nodeptr);
VAR p,q,r,s: nodeptr;
BEGIN  WHILE head <> NIL  DO
        BEGIN  p:= NIL ; q:= head; r:= q ; s:=q^.link ;
          WHILE  s <> NIL  DO
            BEGIN  IF  s^.data < q^.data THEN  BEGIN  (1)__; (2)___ END ;
                r:= s ;   (3)___
              END;
           write(q^.data : 5) ;
           IF p=NIL THEN  (4)___  ELSE  (5)___ ;
           dispose (q) ;
         END;
         writeln
END;【复旦大学 1996 七（20 分） 1995 一（12 分）与本题相似】
```

28. 下面函数的功能是在一个按访问频度不增有序的，带头结点的双向链环上检索关键值为 x 的结点，对该结点访问频度计数，并维护该链环有序。若未找到，则插入该结点。所有结点的频度域初值在建表时都为零。请将程序中四处空缺补写完整。

```
TYPE
     link=^node
     node=RECORD
          key:char; freq:integer; pre,next:link;
          END;
VAR   l:link;
FUNCTION  loc(l:link;x:char):link;
  VAR  p,q:link;
  BEGIN
     p:=l^.next; (1) ;
     WHILE  p^.key<>x   DO  p:=p^.next;
     IF  p=l   THEN  [ new(q); q^.key:=x; q^.freq:=0 ]
     ELSE  {找到}
      [ p^.freq:=p^.freq+1; q:=p; (2) ;
       WHILE  q^.freq>p^.pre^.freq  DO  p:=p^.pre;
       IF   p<>q   THEN  [ (3) ]
      ];
     IF (4) THEN [q^.next:=p, q^.pre;=p^.pre; p^.pre^.next:=q;  p^.pre:=q]
     return(q);
  END;【北京工业大学 1999 五 (12 分)】
```

29. 循环链表 a 和 b 的结点值为字母，其中 a 表非递减有序，下面的程序欲构造一个递增有序的循环链表 c，其中结点的值为同时在 a，b 两链表中出现的字母，且 c 中字母不重复，

请补上程序中空缺的部分，并估计算法的时间复杂度。（设 a，b 的结点数分别为 m，n）

```
TYPE
  link=^node;
  node=RECORD
         key:char;  next:link
       END;
PROC  jj(a,b:link; VAR  c:link);
VAR  p,q,r,s:link;
BEGIN
  new(c);c^.next:=c;  q:=a;  p:=a^.next;
  WHILE  p<>a DO
   [ (1)  ;
    WHILE  p^.key=p^.next^.key  DO  [q:=p;  p=p^.next];{跳过相同字母}
    r:=b^.next ;  (2)  ;
    WHILE  r^.key <>p^.key DO r:=r^.next;
    IF  r<>b  THEN
       [ s:=p;  q^.next:=p^.next;  (3)  ;
         s^.next:=c^.next;  c^.next:=s;  c:=s ]
    ELSE [ q:=p;  p:=p^.next ]
  ];  c:=c^.next;
  END;
```

算法时间复杂度为 0(4)_____【北京工业大学 2000 四 (15 分)】

30. **以下程序的功能是实现带附加头结点的单链表数据结点逆序连接，请填空完善之。**

```
void reverse(pointer h)
 /* h 为附加头结点指针；类型 pointer 同算法设计第 3 题*/
{ pointer p,q;
  p=h->next;  h->next=NULL;
  while((1)_____)
  {q=p; p=p->next; q->next=h->next; h->next=(2)_____;  }
}
```

【西南交通大学 2000 一、9】

31. 下面是用 c 语言编写的对不带头结点的单链表进行就地逆置的算法，该算法用 L 返回逆置后的链表的头指针，试在空缺处填入适当的语句。

```
void  reverse (linklist &L) {
      p=null; q=L;
      while (q! =null)
      { (1)_____;   q->next=p; p=q; (2)____ ; }
        (3)_____;
    }
```

【北京理工大学 2001 九、1 （6 分)】

32. 下面程序段是逆转单向循环链表的方法，p_0 是原链表头指针，逆转后链表头指针仍为 p_0。

（可以根据需要增加标识符）

```
p:= p0;   q0:=NIL;
WHILE (1)_____ DO
  BEGIN  (2)_____; (3)_____;(4)_____;(5)_____  END;
```

p^.next:= q_0;　　p_0^.next:=p;　　p_0:=p;【中国人民大学 2000 二、1（4 分）】

33. 一个无头结点的线性链表(不循环)有两个域。数据域 data，指针域 next，链首 head，下面算法用 read(num)读入数据，当 num 小于 0 时，输入结束。建立一个数据以递增序组成的链表。

```
PROC  insert( head, x);
 {在链首为 head 的表中按递增序插入 x}
  new(r);r ^ .data:=x;
  IF head=NIL
    THEN[ head:=(1)_____; r ^ .next:= (2)_____ ]
     ELSE IF (3)___ THEN [r ^  .next:=head;  head:=r]
         ELSE  [p:=head;
          WHILE (4)___  AND  (p ^ .next≠NIL ) DO[q:=p; (5)___   ];
          IF (6)___  THEN [ q ^  .next:=(7)___; r ^ .next:= (8)___; ]
           ELSE [p ^ .next:=(9)____; r ^ .next:= (10)___;  ]
  ]
 ENDP;
 PROC  creat(head);
    head:= (11)_____; read(num);
    WHILE  num>0  DO
       [ insert(head,num); read(num) ]
 ENDP;【南京理工大学 1999 三、4（11 分)】
```

34. 一元稀疏多项式以循环单链表按降幂排列，结点有三个域，系数域 coef ，指数域 exp 和指针域 next；现对链表求一阶导数 ，链表的头指针为 ha，头结点的 exp 域为 －1。

```
derivative(ha)
 { q=ha ;   pa=ha->next;
  while( (1)_____)
    { if ( (2)___) { ( (3)__); free(pa);  pa= ( (4) _); }
      else{ pa->coef ( (5) ___); pa->exp( (6)___); q=( (7) __);}
      pa=( (8)_____);
    }
 }
```

【南京理工大学 2000 三、3（10 分)】

35.下面是删除单链表 L 中最大元素所在结点的类 PASCAL 语言算法，请在横线填上内容，完成其功能。

```
TYPE  pointer =↑node;
      node=RECORD
             data:integer;  next: pointer
           END;
PROCEDURE delmax (L:pointer);
  VAR  p,q,r:pointer; m:integer;
  BEGIN
    r:=L;  p:=L↑.next;
    IF  p<>NIL  THEN
     [ m:=p↑.data;  (1)_____;  p:=p↑.next;
       WHILE p<>NIL DO
```

```
        [ IF (2)_____THEN  [ (3)_____ ;  m:=p↑.data; ]
          (4)_____; p:=p↑.next;
        ]
       q:=r↑.next; (5)_____; dispose(q);
     ]
END;【北京科技大学 1998 二】
```

36. 对单链表中元素按插入方法排序的 C 语言描述算法如下，其中 L 为链表头结点指针。请填充算法中标出的空白处，完成其功能。

```
typedef struct node
      {int data;  struct node *next;
      }linknode,*link;
void Insertsort(link L)
    { link p,q,r,u;
      p=L->next;   (1)_____;
      while((2)_____)
        { r=L;  q=L->next;
          while((3)_____&& q->data<=p->data) {r=q; q=q->next;}
          u=p->next;  (4)_____;  (5)_____;  p=u;
        }
    }【北京科技大学 2001 二 （10 分）】
```

37. 下面是一个求两个集合 A 和 B 之差 C=A-B 的程序，即当且仅当 e 是 A 的一个元素，但不是 B 中的一个元素时，e 才是 C 中的一个元素。集合用有序链表实现，初始时，A，B 集合中的元素按递增排列，C 为空；操作完成后 A，B 保持不变，C 中元素按递增排列。下面的函数 append(last,e)是把值为 e 的新结点链接在由指针 last 指向的结点的后面，并返回新结点的地址；函数 difference(A,B)实现集合运算 A-B，并返回表示结果集合 C 的链表的首结点的地址。在执行 A-B 运算之前，用于表示结果集合的链表首先增加一个附加的表头结点，以便新结点的添加，当 A-B 运算执行完毕，再删除并释放表示结果集合的链表的表头结点。

程序(a)（编者略去这个 PASCAL 程序）

程序（b）

```
 typedef struct node{ int element;  struct node *link;
                    }NODE;
NODE  *A, *B, *C;
NODE  *append (NODE *last,int e)
{  last->link=(NODE*) malloc (sizeof(NODE));
   last->link->element=e;
   return(last->link);
}
NODE *difference(NODE *A,NODE *B)
{NODE *C,*last;
 C=last=(NODE*) malloc (sizeof(NODE));
 while (1)___
    if (A->element<B->element) { last=append(last,A->element);  A=A->link;  }
   else  if (2) ___ { A=A->link;  B=B->link;  } ELSE  (3) ___ ;
   while (4) __
```

```
    { last=append(last,A->element); A=A->link;  }
   (5)    ;  last=C;    C=C->link;  free (last);   return (C);
  }
/*call form:C=difference(A,B);*/
```
【上海大学 2000 一、4 （10 分）】

四　应用题

1．线性表有两种存储结构：一是顺序表，二是链表。试问：

（1）如果有 n 个线性表同时并存，并且在处理过程中各表的长度会动态变化，线性表的总数也会自动地改变。在此情况下，应选用哪种存储结构？ 为什么？

（2）若线性表的总数基本稳定，且很少进行插入和删除，但要求以最快的速度存取线性表中的元素，那么应采用哪种存储结构？为什么？【西安电子科技大学 1999 软件 二、1（5 分）】

2．线性表的顺序存储结构具有三个弱点：其一，在作插入或删除操作时，需移动大量元素；其二，由于难以估计，必须预先分配较大的空间，往往使存储空间不能得到充分利用；其三，表的容量难以扩充。线性表的链式存储结构是否一定都能够克服上述三个弱点，试讨论之。【重庆大学 2000 二、5】

3.若较频繁地对一个线性表进行插入和删除操作，该线性表宜采用何种存储结构？为什么？

【北京航空航天大学 1998 一、2（4 分）】

4．线性结构包括_____、_____、_____和_____。线性表的存储结构分成_____和_____。请用类 PASCAL 语言描述这两种结构。【华北计算机系统工程研究所 1999 一、2（10 分）】

5．线性表（a_1，a_2，…，a_n）用顺序映射表示时，a_i 和 a_{i+1}（1<=i<n）的物理位置相邻吗？链接表示时呢？

【东南大学 1996 一、1 （5 分）】

6. 说明在线性表的链式存储结构中，头指针与头结点之间的根本区别；头结点与首元结点的关系。

【厦门大学 2000 五、1 （14%/3 分）】

7. 试述头结点,首元结点,头指针这三个概念的区别.

【武汉交通科技大学 1996 二、2 (3 分)】【西安电子科技大学 2001 计应用 二、1（5 分）】

8. 已知有如下定义的静态链表：

```
TYPE    component=RECORD
                    data:elemtp;
                    next:0..maxsize
                  END
VAR  stalist:ARRAY[0..maxsize] OF component;
```

以及三个指针:av 指向头结点，p 指向当前结点，pre 指向前驱结点，现要求修改静态链表中 next 域中的内容，使得该静态链表有双向链表的功能，从当前结点 p 既能往后查找，也能往前查找：

（1） 定义 next 域中的内容。(用老的 next 域中的值表示)；

（2） 如何得到当前结点 p 的前驱（pre）的前驱，给出计算式;

（3） 如何得到 p 的后继，给出计算式;【中科院计算所 2000 四（10 分）】

9. 在单链表和双向链表中，能否从当前结点出发访问到任何一个结点?

【西安电子科技大学 1999 计应用一、1 （5 分）】

10. 如何通过改链的方法，把一个单向链表变成一个与原来链接方向相反的单向链表？

【中国人民大学 2001 二、4 （2 分）】

11. 下面是一算法的核心部分，试说明该算法的功能。

```
pre:=L↑.next;
{L 是一单链表，结点有数据域 data 和指针域 next}
IF  pre<>NIL  THEN
 WHILE  pre↑.next<>NIL DO
 BEGIN p:=pre ↑.next; IF p ↑.data>=pre ↑.data  THEN  pre:=p  ELSE return(false) END;
 return(true);
```

【燕山大学 2000 七、1 （7 分）】

12. 设单链表结点指针域为 next，试写出删除链表中指针 p 所指结点的直接后继的 C 语言语句。

【北京科技大学 2000 一、3】

13. 设单链表中某指针 p 所指结点（即 p 结点）的数据域为 data，链指针域为 next，请写出在 p 结点之前插入 s 结点的操作（PASCAL 语句）。【北京科技大学 1999 一、2 （2 分）】

14. 有线性表($a_1, a_2, \cdots, a_n$),采用单链表存储，头指针为 H，每个结点中存放线性表中一个元素，现查找某个元素值等于 X 的结点。分别写出下面三种情况的查找语句。要求时间尽量少。

（1）线性表中元素无序。（2）线性表中元素按递增有序。 （3）线性表中元素按递减有序。

【北京邮电大学 1994 七 （7 分）】

15. 设 pa，pb 分别指向两个带头结点的有序（从小到大）单链表。仔细阅读如下的程序，并回答问题：

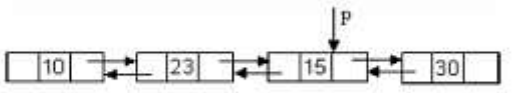
（1） 程序的功能。（2） s1，s2 中值的含义。（3） pa，pb 中值的含义。

```
PROCEDURE  exam(pa,pb)
  BEGIN
    p1:=pa↑.next; p2:=pb↑.next; pa↑.next:=∧; s1:=0; s2:=0;
    WHILE  p1≠∧  AND p2≠∧  DO
    [  CASE  p1  ↑  .data<p2  ↑  .data:  [p:=p1;  p1:=p1  ↑  .next;  s2:=s2+1; dispose(p)  ];
            p1↑.data>p2↑.data: p2:=p2↑.next;
            p1↑.data=p2↑.data: [p:=p1; p1:=p1↑.next; p↑.next:= pa↑.next;
                                pa↑.next:= p; p2:= p2↑.next;s1:=s1+1;  ];
      END
    ];
   WHILE  p1≠∧  DO [ p:=p1; p1:=p1↑.next; dispose(p); s2:=s2+1 ]
  END;
```

【南京航空航天大学 1995 十 （9 分）】

16. 写出下图双链表中对换值为 23 和 15 的两个结点相互位置时修改指针的有关语句。

结点结构为：(llink,data,rlink) 【北京邮电大学 1992 三、4 （25/4 分）】



17. 按照下列题目中的算法功能说明，将算法描述片段中的错误改正过来。

（1） （4 分）下面的算法描述片段用于在双链表中删除指针变量 p 所指的结点：

```
p^.rlink←p^.llink^.rlink;
```

```
p^.llink←p.^rlink^.llink
dispose(p);
```

（2） （6 分）下面的算法描述片段用于在双链表中指针变量 p 所指结点后插入一个新结点：

```
new(q);
q^.llink←p;
p^.rlink←q;
q^.rlink←p^.rlink;
q←p^.rlink^.llink;
```

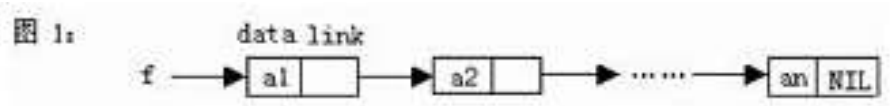
【山东大学 1999 八（10 分）】

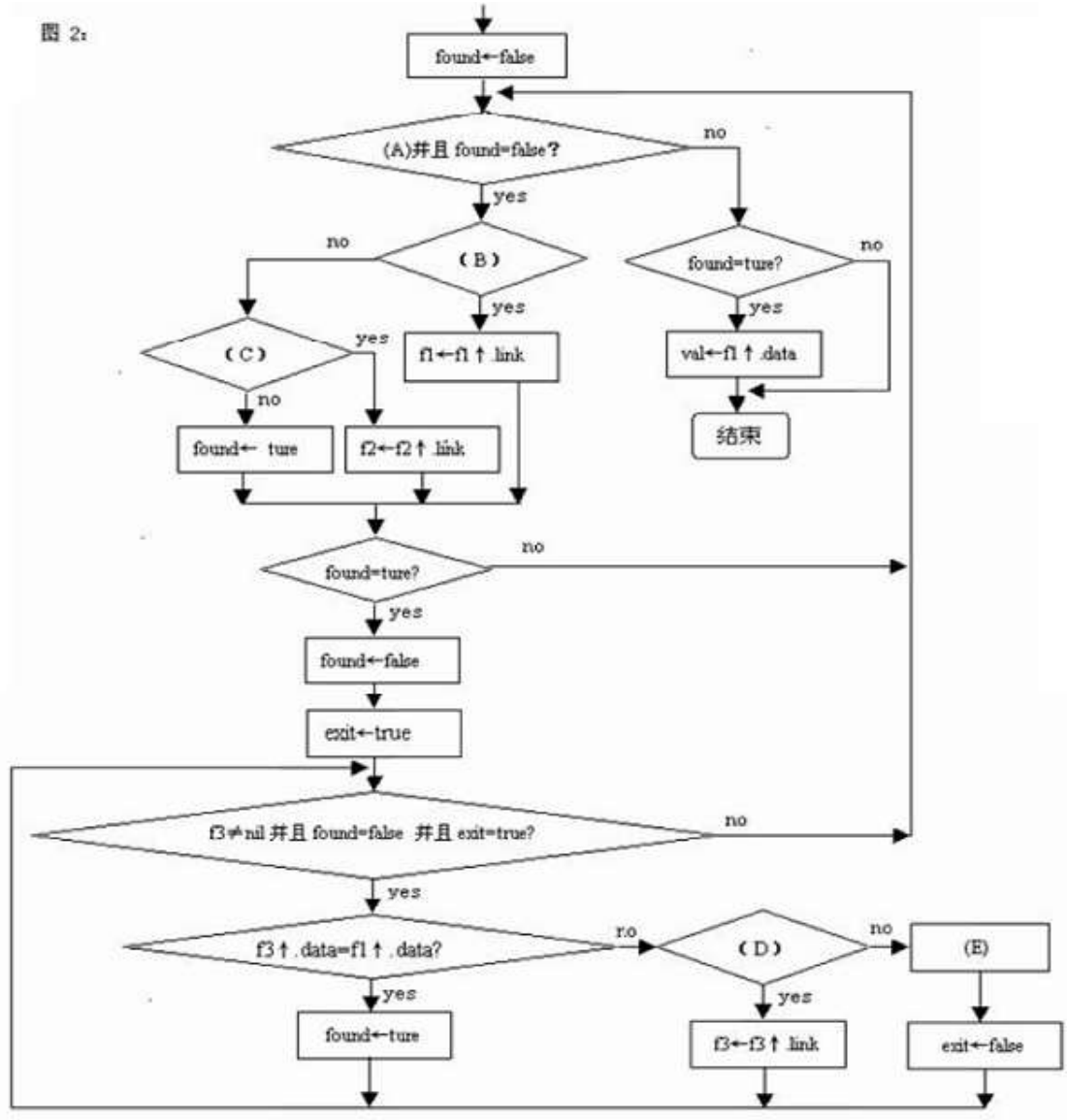
18. 已知 L 是一个数据类型 linkedlist 的单循环链表，pa 和 pb 是指向 L 中结点的指针。简述下列程序段的功能。【山东科技大学 2001 一、2 （5 分）】

```
TYPE linkedlist=↑node;
        node=RECORD
               data:datatype; next:linkedlist
             END;
PROC  Mp(pa,pb:linkedlist);
PROC  subp(s,q: linkedlist);
  p:=s;
  WHILE  p↑.next<>q  DO  p:=p↑.next;
  p↑.next:=s
ENDP;
subp(pa,pb);
subp(pb,pa);
ENDP;
```

19. 设双向循环链表中结点的数据域、前驱和后继指针域分别为 data,pre 和 next,试写出在指针 p 所指结点之前插入一 s 结点的 C 语言描述语句。【北京科技大学 2001 一、3 （2 分）】

20. 本题给出一个子程序的框图，如图 2，试填空完善此算法框图。该子程序用来寻找第一个均出现在三个整数单向链表 f1，f2，f3 中的相同整数。假定在调用该子程序前，这三个整数链表已按从小到大的次序排序，单向链表的形式如下图 1 的例子所示。





注：在图 2 的框图中：found 和 exit 均为布尔型的变量，可取值为 true 和 false。val 是整型变量，用来存放第一个均出现在 f1，f2，f3 中的相同整数。若 f1，f2 和 f3 中无相同的整数，found 的值为 false，否则 found 的值为 true。f1↑.link 表示访问 f1 所指结点的 link 域。

【哈尔滨工业大学 1999 三 （15 分）】

21. 一线性表存储在带头结点的双向循环链表中，L 为头指针。如下算法：

（1）说明该算法的功能。（2）在空缺处填写相应的语句。

```
void unknown  (BNODETP *L)
{    …
  p=L->next; q=p->next; r=q->next;
  while (q!=L)
    { while (p!=L) && (p->data>q->data)  p=p->prior;
      q->prior->next=r; (1) _____;
```

```
        q->next=p->next; q->prior=p;
        (2)_____; (3)_____;  q=r; p=q->prior;
        (4)_____;
      }
    }
```

【北京理工大学 1999 第二部分 数据结构［7］（8 分)】

五、算法设计题

1. **假设有两个按元素值递增次序排列的线性表，均以单链表形式存储。请编写算法将这**两个单链表归并为一个按元素值递减次序排列的单链表，并要求利用原来两个单链表的结点存放归并后的单链表。

【北京大学 1998 三、1 （5 分)】

类似本题的另外叙述有：

（1）设有两个无头结点的单链表,头指针分别为 ha,hb,链中有数据域 data,链域 next,两链表的数据都按递增序存放,现要求将 hb 表归到 ha 表中,且归并后 ha 仍递增序,归并中 ha 表中已有的数据若 hb 中也有,则 hb 中的数据不归并到 ha 中,hb 的链表在算法中不允许破坏。【南京理工大学 1997 四、3（15 分)】

```
PROCEDURE   merge(ha,hb);
```

（2）已知头指针分别为 la 和 lb 的带头结点的单链表中，结点按元素值非递减有序排列。写出将 la 和 lb 两链表归并成一个结点按元素值非递减有序排列的单链表（其头指针为 lc)，并计算算法的时间复杂度。【燕山大学 1998 五 （20 分)】

2. 图（编者略）中带头结点且头指针为 ha 和 hb 的两线性表 A 和 B 分别表示两个集合。两表中的元素皆为递增有序。请写一算法求 A 和 B 的并集 AUB。要求该并集中的元素仍保持递增有序。且要利用 A 和 B 的原有结点空间。【北京邮电大学 1992 二 （15 分)】

类似本题的另外叙述有：

（1） 已知递增有序的两个单链表 A，B 分别存储了一个集合。设计算法实现求两个集合的并集的运算 A:=A∪B【合肥工业大学 1999 五、1（8 分)】

（2）已知两个链表 A 和 B 分别表示两个集合，其元素递增排列。编一函数，求 A 与 B 的交集，并存放于 A 链表中。【南京航空航天大学 2001 六（10 分)】

（3)设有两个从小到大排序的带头结点的有序链表。试编写求这两个链表交运算的算法(即 L1∩L2)。要求结果链表仍是从小到大排序，但无重复元素。【南京航空航天大学 1996 十一（10 分)】

（4）已知两个线性表 A ，B 均以带头结点的单链表作存储结构，且表中元素按值递增有序排列。设计算法求出 A 与 B 的交集 C，要求 C 另开辟存储空间，要求 C 同样以元素值的递增序的单链表形式存贮。

【西北大学 2000 五 （ 8 分)】

（5）已知递增有序的单链表 A,B 和 C 分别存储了一个集合，设计算法实现 A:=A∪（B∩C)，并使求解结构 A 仍保持递增。要求算法的时间复杂度为 O(|A|+|B|+|C|)。其中,|A|为集合 A 的元素个数。

【合肥工业大学 2000 五、1（8 分)】

3. 知 L1、L2 分别为两循环单链表的头结点指针，m,n 分别为 L1、L2 表中数据结点个数。要求设计一算法，用最快速度将两表合并成一个带头结点的循环单链表。【东北大学 1996 二 (12 分)】

类似本题的另外叙述有：

（1）试用类 Pascal 语言编写过程 PROC join（VAR la：link； lb：link） 实现连

接线性表 la 和 lb(lb 在后)的算法，要求其时间复杂度为 0（1）， 占用辅助空间尽量小。描述所用结构。

【北京工业大学 1997 一、1 (8 分)】

（2）设有两个链表，ha 为单向链表，hb 为单向循环链表。编写算法，将两个链表合并成一个单向链表，要求算法所需时间与链表长度无关。【南京航空航天大学 1997 四（8 分)】

4. 顺序结构线性表 LA 与 LB 的结点关键字为整数。LA 与 LB 的元素按非递减有序，线性表空间足够大。试用类 PASCAL 语言给出一种高效算法，将 LB 中元素合到 LA 中，使新的 LA 的元素仍保持非递减有序。高效指最大限度的避免移动元素。【北京工业大学 1997 一、2 (12 分)】

5. 已知不带头结点的线性链表 list，链表中结点构造为（data、link)，其中 data 为数据域，link 为指针域。请写一算法，将该链表按结点数据域的值的大小从小到大重新链接。要求链接过程中不得使用除该链表以外的任何链结点空间。【北京航空航天大学 1998 五(15 分)】

6. 设 L 为单链表的头结点地址，其数据结点的数据都是正整数且无相同的，试设计利用直接插入的原则把该链表整理成数据递增的有序单链表的算法。【东北大学 1996 六 (14 分)】

类似本题的另外叙述有：

（1）设一单向链表的头指针为 head,链表的记录中包含着整数类型的 key 域,试设计算法,将此链表的记录按照 key 递增的次序进行就地排序.【中科院计算所 1999 五、1（10 分)】

7. 设 Listhead 为一单链表的头指针,单链表的每个结点由一个整数域 DATA 和指针域 NEXT 组成，整数在单链表中是无序的。编一 PASCAL 过程，将 Listhead 链中结点分成一个奇数链和一个偶数链，分别由 P,Q 指向，每个链中的数据按由小到大排列。程序中不得使用 NEW 过程申请空间。【山东大学 1993 六(15 分)】

类似本题的另外叙述有：

（1）设计算法将一个带头结点的单链表 A 分解为两个具有相同结构的链表 B、C，其中 B 表的结点为 A 表中值小于零的结点，而 C 表的结点为 A 表中值大于零的结点（链表 A 的元素类型为整型，要求 B、C 表利用 A 表的结点)。【北京理工大学 2000 四、2（4 分)】

(2) 设 L 为一单链表的头指针，单链表的每个结点由一个整数域 data 和指针域 NEXT 组成，整数在单链表中是无序的。设计算法，将链表中结点分成一个奇数链和一个偶数链，分别由 P，Q 指向，每个链中的数据按由小到大排列，算法中不得申请新的结点空间。【青岛海洋大学 1999 三（12 分)】

(3) 将一个带头结点的单链表 A 分解为两个带头结点的单链表 A 和 B，使得 A 表中含有原表中序号为奇数的元素，而 B 表中含有原表中序号为偶数的元素，且保持其相对顺序不变。

1) 写出其类型定义：

2) 写出算法。【山东大学 1998 九 (9 分)】 【山东工业大学 2000 九（9 分)】

8. 已知线性表（a1 a2 a3 …an）按顺序存于内存，每个元素都是整数，试设计用最少时间把所有值为负数的元素移到全部正数值元素前边的算法：例：(x,-x,-x,x,x,-x …x）变为（-x,-x,-x…x,x,x)。

【东北大学 1998 二 (15 分)】

类似本题的另外叙述有：

（1）设有一元素为整数的线性表 L=(a_1,a_2,a3,…,a_n),存放在一维数组 A[N]中,设计一个算法,以表中 a_n 作为参考元素,将该表分为左、右两部分,其中左半部分每个元素小于等于 a_n,右半部分每个元素都大于 a_n, a_n 位于分界位置上(要求结果仍存放在 A[N]中)。【北京理工大学 1999 八（6 分)】

（2）顺序存储的线性表 A,其数据元素为整型,试编写一算法,将 A 拆成 B 和 C 两个表,使 A 中元素值大于等于 0 的元素放入 B,小于 0 的放入 C 中.. 要求:

1）表 B 和 C 另外设置存储空间;

2）表 B 和 C 不另外设置,而利用 A 的空间.【山东大学 2001 九、1 (12 分)】

（3）知线性表（a1, a2,a3,…,an）按顺序存储，且每个元素都是整数均不相同，设计把所有奇数移到所有偶数前边的算法。(要求时间最少,辅助空间最少)【东北大学 1997 三 (15 分)】

(4) 编写函数将一整数序列中所有负数移到所有正数之前，要求时间复杂度为 0（n）

【南京航空航天大学 2001 八（10 分)】

(5) 已知一个由 n（ 设 n=1000）个整数组成的线性表，试设计该线性表的一种存储结构，并用标准 pascal 语言描述算法，实现将 n 个元素中所有大于等于 19 的整数放在所有小于 19 的整数之后。要求算法的时间复杂度为 O(n),空间复杂度 O(1)。【西安交通大学 1996 六（11 分）】

9. 试编写在带头结点的单链表中删除（一个）最小值结点的（高效）算法。void delete（Linklist &L）

【北京理工大学 2001 九、3 （8 分)】

10. 已知非空线性链表由 list 指出，链结点的构造为（data,link）.请写一算法，将链表中数据域值最小的那个链结点移到链表的最前面。要求：不得额外申请新的链结点。【北京航空航天大学 2001 四（10 分)】

11. 已知 p 指向双向循环链表中的一个结点，其结点结构为 data、llink、rlink 三个域，写出算法 change(p),交换 p 所指向的结点和它的前缀结点的顺序。【首都经贸大学 1997 二、2（15 分)】

12. 线性表($a_1, a_2, a_3, \cdots, a_n$)中元素递增有序且按顺序存储于计算机内。要求设计一算法完成：

（1） 用最少时间在表中查找数值为 x 的元素。

（2） 若找到将其与后继元素位置相交换。

（3） 若找不到将其插入表中并使表中元素仍递增有序。【东北大学 1996 三 (12 分)】

13. 设单链表的表头指针为 h，结点结构由 data 和 next 两个域构成，其中 data 域为字符型。写出算法 dc(h,n),判断该链表的前 n 个字符是否中心对称。例如 xyx, xyyx 都是中心对称。【首都经贸大学 1998 三、9（15 分)】

14. 已知两个单链表 A 和 B,其头指针分别为 heada 和 headb,编写一个过程从单链表 A 中删除自第 i 个元素起的共 len 个元素，然后将单链表 A 插入到单链表 B 的第 j 个元素之前。

【中国矿业大学 2000 三（10 分)】

类似本题的另外叙述有：

（1）h1、h2 为两个链表的表头指针，结点结构为 data 和 link 两个域组成。写出算法 inde(h1,h2,i,j,l),将链表 h1 从第 i 个结点起的 l 个结点删除,并插入到 h2 表的第 j 个结点之前。

【首都经贸大学 1998 三、10（20 分)】

15. 设线性表存于 A[1..size]的前 num 各分量中，且递增有序。请设计一个算法，将 x 插入到线性表的适当位置上，以保持线性表的有序性，并在设计前说明设计思想，最后说明所设计算法的时间复杂度。

【西安电子科技大学 1999 计应用 1997 二 （10 分)】

类似本题的另外叙述有：

(1) 试编制在线性表 L={12,13,21,24,28,30,42,}中插入数据元素 26 的程序。(要求该

程序用 turbo Pascal 语言编制并能在计算机上运行，结点类型为链式结构)【大连海事大学 1996 二、1 (16 分)】

16. 假设一个单循环链表，其结点含有三个域 pre、data、link。其中 data 为数据域；pre 为指针域，它的值为空指针（NIL）；link 为指针域，它指向后继结点。请设计算法，将此表改成双向循环链表。

【西安电子科技大学 1999 软件 五（10 分)】

17. 已知递增有序的单链表 A,B 分别存储了一个集合,请设计算法以求出两个集合 A 和 B 的差集 A-B（即仅由在 A 中出现而不在 B 中出现的元素所构成的集合)，并以同样的形式存储，同时返回该集合的元素个数。

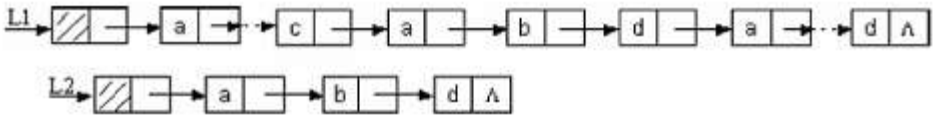
【西安电子科技大学 2000 计应用 1997 二 （10 分)】

18. 已知一个单链表中每个结点存放一个整数，并且结点数不少于 2，请设计算法以判断该链表中第二项起的每个元素值是否等于其序号的平方减去其前驱的值，若满足则返回 ture，否则返回 false.

【西安电子科技大学 2000 软件 1997 二（10 分)】

19. 两个整数序列 A=a1,a2,a3,…,am 和 B=b1,b2,b3,…,bn 已经存入两个单链表中，设计一个算法，判断序列 B 是否是序列 A 的子序列。【东北大学 1999 二 (10 分)】

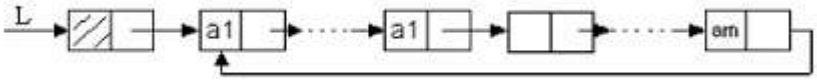
20. L1 与 L2 分别为两单链表头结点地址指针，且两表中数据结点的数据域均为一个字母。设计把 L1 中与 L2 中数据相同的连续结点顺序完全倒置的算法。【东北大学 1997 四 （1 5 分)】



例：

类似本题的另外叙述有：

(1) 知 L 为链表的头结点地址，表中共有 m(m>3)个结点，从表中第 i 个结点(1<i<m)起到第 m 个结点构成一个循环部分链表,设计将这部分循环链表中所有结点顺序完全倒置的算法。



【东北大学 1998 三 (15 分)】

21. 请写一个算法将顺序存储结构的线性表(a_1...a_n)逆置为(a_n...a_1)。【大连海事大学 1996 八(6 分)】

类似本题的另外叙述有：

(1) 设有一带头结点的单链表,编程将链表颠倒过来.要求不用另外的数组或结点完成.

【南京航空航天大学 1999 八 （10 分)】

(2) 设有一个带头结点的单向链表，数据项递减有序。写一算法，重新排列链表，使数据项递增有序，要求算法时间复杂度为 O（n)。(注：用程序实现)【南京航空航天大学 1997 七 （12 分)】

(3) 试编写求倒排循环链表元素的算法。【南京航空航天大学 1995 十二 （10 分)】

(4) 请设计算法将不带头结点的单链表就地逆置。【北方交通大学 2001 三 （12 分)】

(5) 试编写算法 ，将不设表头结点的、不循环的单向链表就地逆转。【北方交通大学 1997 五（10 分)】

(6) 有一个单链表 L（至少有 1 个结点），其头结点指针为 head，编写一个过程将 L 逆置，即最后一个结点变成第一个结点，原来倒数第二个结点变成第二个结点，如此等等。【燕山大学 2001 四、2（8 分）】

22. 设有一个由正整数组成的无序（向后）单链表，编写完成下列功能的算法：

（1）找出最小值结点，且打印该数值；

（2）若该数值是奇数，则将其与直接后继结点的数值交换；

（3）若该数值是偶数，则将其直接后继结点删除。【东北大学 2000 二（15 分）】

23. 已知 L 为没有头结点的的单链表中第一个结点的指针，每个结点数据域存放一个字符，该字符可能是英文字母字符或数字字符或其它字符，编写算法构造三个以带头结点的单循环链表表示的线性表，使每个表中只含同一类字符。（要求用最少的时间和最少的空间）【东北大学 2002 三(15 分)】

24. 在一个递增有序的线性表中，有数值相同的元素存在。若存储方式为单链表，设计算法去掉数值相同的元素，使表中不再有重复的元素。例如：（7，10，10，21，30，42，42，42，51，70）将变作（7，10，21，30，42，51，70），分析算法的时间复杂度。【北京工业大学 1996 三（15 分）】

25. 在输入数据无序的情况下，建立一个数据值为整型的递增有序的顺序存储线性表 L，且要求当输入相同数据值时，线性表中不能存在数据值相同的数据元素，试写出其算法。

顺序存储结构的线性表描述为：

```
CONST   maxlen={线性表可能达到的最大长度};
TYPE    sqlisttp=RECORD
                   elem:array[1..maxlen] of integer;
                   last :0..maxlen
                END;
VAR L: sqlisttp;
```

【同济大学 1998 二（12 分 ）】

26. 设有一个正整数序列组成的有序单链表（按递增次序有序，且允许有相等的整数存在），试编写能实现下列功能的算法 ：（要求用最少的时间和最小的空间）

(1) 确定在序列中比正整数 x 大的数有几个（相同的数只计算一次，如序列{20,20,17,16,15,15,11,10,8,7,7,5,4}中比 10 大的数有 5 个）；

(2) 在单链表将比正整数 x 小的数按递减次序排列；

(3) 将正整数（比）x 大的偶数从单链表中删除。【东北大学 2001 二（17 分）】

27. 编写一个算法来交换单链表中指针 P 所指结点与其后继结点，HEAD 是该链表的头指针，P 指向该链表中某一结点。【吉林大学 2001 二、1（7 分）】

类似本题的另外叙述有：

(1) 已知非空线性链表第一个结点由 List 指出，请写一算法，交换 p 所指的结点与其下一个结点在链表中的位置(设 p 指向的不是链表最后那个结点)。【北京航空航天大学 1999 五（10 分）】

(2) 已知任意单链表如图所示(编者略去图)。Head 为表头指针，指向表的第一个元素，p 为指向表中任意结点的指针，试设计一个算法，将 p 指向的结点和其后面结点交换位置（可采用任何高级语言描述算法）。

【山东大学 1992 二（ 12 分）】

28. 设键盘输入 n 个英语单词，输入格式为 n, w_1, w_2, …, w_n, 其中 n 表示随后输入英语单词个数，试编一程序，建立一个单向链表，实现：（10 分）

（1）如果单词重复出现，则只在链表上保留一个。（单考生做）。

（2）除满足（1）的要求外。链表结点还应有一个计数域，记录该单词重复出现的次数，

然后输出出现次数最多的前 k(k<=n)个单词（统考生做）。【南京航空航天大学 1998 九 （10 分）】

29. 已知一双向循还链表，从第二个结点至表尾递增有序，（设 $a_1<x<a_n$）如下图（“第二个结点至表尾”指 $a_1..a_n$ ，因篇幅所限，编者略去图）。试编写程序，将第一个结点删除并插入表中适当位置，使整个链表递增有序。【南京航空航天大学 1998 八（10 分）】

30. 已知长度为 n 的线性表 A 采用顺序存储结构，请写一时间复杂度为 0(n)、空间复杂度为 0(1)的算法，该算法删除线性表中所有值为 item 的数据元素。（0（1）表示算法的辅助空间为常量）。

【北京航空航天大学 2000 五（10 分）】

31. 设民航公司有一个自动预订飞机票的系统，该系统中有一张用双重链表示的乘客表，表中结点按乘客姓氏的字母序相链。例如，下面是张某个时刻的乘客表。试为该系统写出一个当任一乘客要订票时修改乘客表的算法。

序号	data	Llink	Rlink
1	Liu	6	5
2	Chan	4	9
3	Wang	5	7
4	Bao	0	2
5	Mai	1	3
6	Dong	8	1
7	Xi	3	0
8	Deng	9	6
9	Cuang	2	8

【北方交通大学 2000 六（17 分）】

32. 设有一头指针为 L 的带有表头结点的非循环双向链表，其每个结点中除有 pred（前驱指针），data（数据）和 next（后继指针）域外，还有一个访问频度域 freq。在链表被起用前，其值均初始化为零。每当在链表中进行一次 Locate(L,x)运算时，令元素值为 x 的结点中 freq 域的值增 1，并使此链表中结点保持按访问频度非增（递减）的顺序排列，同时最近访问的结点排在频度相同的结点的最后，以便使频繁访问的结点总是靠近表头。试编写符合上述要求的 Locate(L,x)运算的算法，该运算为函数过程，返回找到结点的地址，类型为指针型。【清华大学 1997 二 （10 分）】

33. 给定(已生成)一个带表头结点的单链表，设 head 为头指针，结点的结构为(data,next),data 为整型元素，next 为指针，试写出算法:按递增次序输出单链表中各结点的数据元素，并释放结点所占的存储空间。(要求;不允许使用数组作辅助空间)【华中理工大学 2000 八、2 （13 分）】

34. 已知三个带头结点的线性链表 A、B 和 C 中的结点均依元素值自小至大非递减排列（可能存在两个以上值相同的结点），编写算法对 A 表进行如下操作：使操作后的链表 A 中仅留下三个表中均包含的数据元素的结点，且没有值相同的结点，并释放所有无用结点。限定算法的时间复杂度为 O(m+n+p)，其中 m、n 和 p 分别为三个表的长度。【清华大学 1995 一（15 分）】

第 2 章 线性表（答案）

一. 选择题

1.A	2.B	3.C	4.A	5.D	6.D	7.D	8.C	9.B	10.B,C	11.1I	11.2I	11.3E
11.4B	11.5C	12.B	13.C	14.C	15.C		16.A	17.A	18.A	19.D	20.C	21.B
22.D	23.C	24.B	25.B	26.A	27.D							

二．判断题

1. ×	2.√	3. √	4.×	5.×	6. ×	7. ×	8.×	9.×	10.×	11.×	12.×
13. ×	14. √	15.×	16. √								

部分答案解释如下。

1、 头结点并不“仅起”标识作用，并且使操作统一。另外，头结点数据域可写入链表长度，或作监视哨。

4．两种存储结构各有优缺点，应根据实际情况选用，不能笼统说哪一个好。

7．集合中元素无逻辑关系。

9．非空线性表第一个元素无前驱，最后一个元素无后继。

13．线性表是逻辑结构，可以顺序存储，也可链式存储。

三．填空题

1．顺序　　2．(n-1)/2　　3．py->next=px->next; px->next=py

4．n-i+1

5．主要是使插入和删除等操作统一，在第一个元素之前插入元素和删除第一个结点不必另作判断。另外，不论链表是否为空，链表指针不变。

6．O(1)，O(n)　　7．单链表，多重链表，(动态）链表，静态链表

8．f->next=p->next; f->prior=p; p->next->prior=f; p->next=f;

9．p^.prior　　s^.prior^.next

10．指针　　11．物理上相邻　指针　　12．4　　2

13．从任一结点出发都可访问到链表中每一个元素。

14．u=p->next; p->next=u->next; free(u);　　15．L->next->next==L

16．p->next!=null

17．L->next==L && L->prior==L　　18．s->next=p->next;p->next=s;

19．(1) **IF** pa=NIL **THEN return**(true);

(2) pb<>NIL **AND** pa^.data>=pb^.data

(3) **return**(inclusion(pa,pb));

(4) pb:=pb^.next;

(5) **return**(false);

非递归算法：

(1)pre:=pb; (2) pa<>NIL AND pb<>NIL AND pb^.data>=pa^.data　(3)pa:=pa^.next; pb:=pb->next;

(4)pb:=pre^.next;pre:=pb;pa:=pa^.next;(5)**IF** pa=NIL **THEN return**(true) **ELSE return**(false);

[注]：本题是在链表上求模式匹配问题。非递归算法中用指针 pre 指向主串中开始结点（初始时为第一元素结点)。若主串与子串对应数据相等，两串工作指针 pa 和 pb 后移；否则，主串工作指针从 pre 的下一结点开始（这时 pre 又指向新的开始结点)，子串工作指针从子串第一元素开始，比较一直继续到循环条件失败。若 pa 为空，则匹配成功，返回 true，否则，返回 false。

20. A. VAR head:ptr　B. new(p)　C. p^.data:=k　D. q^.next:=p　E. q:=p(带头结点)

21.（1）new(h);//生成头结点，以便于操作。

（2）r^.next:=p;　（3）r^.next:=q;　(4) IF (q=NIL) THEN r^.next:=p;

22. A: r^.link^.data<>max AND q^.link^.data<>max

B: r:=r^.link　C: q^.link　D: q^.link　E: r^.link　F: r^.link

G: r:=s（或r:= r^.link）　H: r:=r^.link　I: q^.link:=s^.link

23.(1)la　(2)0　(3)j<i-1　(4)p↑.next　(5)i<1

24.(1)head^.left:=s　//head的前驱指针指向插入结点

(2)j:=1;

(3)p:=p^.right　//工作指针后移

(4)s^.left:=p

(5)p^.right^.left:=s;　//p后继的前驱是s

(6)s^.left:=p;

25.(1)i<=L.last　//L.last 为元素个数

(2)j:=j+1　//有值不相等的元素

(3)L.elem[j]:=L.elem[i]　//元素前移

(4)L.last:=j　//元素个数

26.(A)p^.link:=q;//拉上链，前驱指向后继

(B)p:=q;//新的前驱

(C)p^.link:=head;//形成循环链表

(D)j:=0;　//计数器，记被删结点

(E)q:=p^.link//记下被删结点

(F)p^.link=q^.link　//删除结点

27. (1)p:=r;//r指向工作指针s的前驱，p指向最小值的前驱。

(2)q:=s;//q指向最小值结点，s是工作指针

(3)s:=s^.link//工作指针后移

(4)head:=head^.next;//第一个结点值最小;

(5)p^link:=q^.link;//跨过被删结点（即删除一结点）

28. (1) l^.key:=x;//头结点l这时起监视哨作用

(2) l^.freq:=p^.freq　//头结点起监视哨作用

(3) q->pre->next=q->next; q->next->pre=q->pre;　//先将q结点从链表上摘下

q^.next:=p; q^.pre:=p^.pre; p^.pre->next:=q; p^.pre:=q; //结点q插入结点p前

(4) q^.freq=0　//链表中无值为x的结点，将新建结点插入到链表最后（头结点前）。

29. (1)a^.key:=’@’ //a的头结点用作监视哨，取不同于a链表中其它数据域的值

(2)b^.key:=p^.key//b的头结点起监视哨作用

(3)p:=p^.next//找到a,b表中共同字母，a表指针后移

(4)O(m*n)

30. C 部分：(1)p!=null　//链表未到尾就一直作

(2)q　//将当前结点作为头结点后的第一元素结点插入

31. (1)L=L->next;　//暂存后继

(2)q=L;　//待逆置结点

(3)L=p;　　//头指针仍为 L

32. (1) p^.next<>p_0　　(2)r:= p^.next　　(3) p^.next:= q_0;
(4) q_0:= p;　　(5) p:=r

33. (1)r　　(2)NIL　　(3)x<head^.data　　(4)p^.data<x
(5)p:=p^.next　(6)p^.data>=x;　(7)r　　(8)p
(9)r　　(10)NIL　　(11)NIL

34. (1)pa!=ha　　//或 pa->exp!=-1
(2)pa->exp==0　　//若指数为 0，即本项为常数项
(3)q->next=pa->next　//删常数项
(4)q->next　　//取下一元素
(5)=pa->coef*pa->exp
(6)--　　//指数项减 1
(7)pa　　//前驱后移,或 q->next
(8)pa->next　　//取下一元素

35. (1)q:=p;　　//q 是工作指针 p 的前驱
(2)p^.data>m　　//p 是工作指针
(3)r:=q;　　//r 记最大值的前驱，
(4)q:=p;　　//或 q:=q^.next;
(5)r^.next:=q^.next;　　//或 r^.next:=r^.next^.next 删最大值结点

36. (1)L->next=null　　//置空链表,然后将原链表结点逐个插入到有序表中
(2)p!=null　　//当链表尚未到尾，p 为工作指针
(3)q!=null　　//查 p 结点在链表中的插入位置，这时 q 是工作指针。
(4)p->next=r->next　//将 p 结点链入链表中
(5)r->next=p　　//r 是 q 的前驱，u 是下个待插入结点的指针。

37. 程序（a）　PASCAL 部分(编者略)
程序（b）　C 部分
(1)(A!=null && B!=null)　　//两均未空时循环
(2)A->element==B->element　　//两表中相等元素不作结果元素
(3)B=B->link　　//向后移动 B 表指针
(4)A!=null　　//将 A 表剩余部分放入结果表中
(5)last->link=null　　//置链表尾

四、　应用题

1.（1）选链式存储结构。它可动态申请内存空间，不受表长度（即表中元素个数）的影响，插入、删除时间复杂度为 O（1）。

（2）选顺序存储结构。顺序表可以随机存取，时间复杂度为 O（1）。

2. 链式存储结构一般说克服了顺序存储结构的三个弱点。首先，插入、删除不需移动元素，只修改指针，时间复杂度为 O(1)；其次，不需要预先分配空间，可根据需要动态申请空间；其三，表容量只受可用内存空间的限制。其缺点是因为指针增加了空间开销，当空间不允许时，就不能克服顺序存储的缺点。

3. 采用链式存储结构，它根据实际需要申请内存空间，而当不需要时又可将不用结点空间返还给系统。在链式存储结构中插入和删除操作不需要移动元素。

4. 线性表　栈　队列　串　　顺序存储结构和链式存储结构。

顺序存储结构的定义是：

```
CONST maxlen=线性表可能达到的最大长度;
TYPE  sqlisttp=RECORD
                 elem:ARRAY[1..maxlen] OF ElemType;
                 last:0..maxlen;
               END;
```

链式存储结构的定义是：

```
TYPE pointer=↑nodetype;
     nodetype=RECORD
                data:ElemType;
                next:pointer;
              END;
     linklisttp=pointer;
```

5.顺序映射时，a_i 与 a_{i+1} 的物理位置相邻；链表表示时 a_i 与 a_{i+1} 的物理位置不要求相邻。

6. 在线性表的链式存储结构中，头指针指链表的指针，若链表有头结点则是链表的头结点的指针，头指针具有标识作用，故常用头指针冠以链表的名字。头结点是为了操作的统一、方便而设立的，放在第一元素结点之前，其数据域一般无意义（当然有些情况下也可存放链表的长度、用做监视哨等等），有头结点后，对在第一元素结点前插入结点和删除第一结点，其操作与对其它结点的操作统一了。而且无论链表是否为空，头指针均不为空。首元结点也就是第一元素结点，它是头结点后边的第一个结点。

7. 见上题 6。

8. (1)将 next 域变为两个域：pre 和 next，其值域均为 0..maxsize。初始化时，头结点（下标为 0 的元素）其 next 域值为 1，其 pre 域值为 n（设 n 是元素个数，且 n<maxsize）

(2) stalist[stalist[p].pre].pre;

(3) stalist[p].next;

9. 在单链表中不能从当前结点（若当前结点不是第一结点）出发访问到任何一个结点，链表只能从头指针开始，访问到链表中每个结点。在双链表中求前驱和后继都容易，从当前结点向前到第一结点，向后到最后结点，可以访问到任何一个结点。

10. 本题是链表的逆置问题。设该链表带头结点，将头结点摘下，并将其指针域置空。然后从第一元素结点开始，直到最后一个结点为止，依次前插入头结点的后面，则实现了链表的逆置。

11.该算法的功能是判断链表 L 是否是非递减有序，若是则返回“true”；否则返回“false“。pre 指向当前结点，p 指向 pre 的后继。

12. q=p->next; p->next=q->next; free(q);

13. 设单链表的头结点的头指针为 head,且 pre=head;

```
while(pre->next!=p) pre=pre->next;
s->next=p; pre->next=s;
```

14.设单链表带头结点，工作指针 p 初始化为 p=H->next;

(1)
```
while(p!=null && p->data!=X) p=p->next;
if(p= =null) return(null);//查找失败
else return(p);//查找成功
```

(2)
```
while(p!=null && p->data<X ) p=p->next;
if(p==null || p->data>X) return(null);//查找失败
else return(p);
```

(3) **while**(p!=null && p->data>X) p=p->next;
if(p==null || p->data<X) **return**(null); //查找失败
else return(p); //查找成功

15.本程序段功能是将 pa 和 pb 链表中的值相同的结点保留在 pa 链表中（pa 中与 pb 中不同结点删除），pa 是结果链表的头指针。链表中结点值与从前逆序。S1 记结果链表中结点个数（即 pa 与 pb 中相等的元素个数）。S2 记原 pa 链表中删除的结点个数。

16. 设 q:=p^.llink; 则
q^.rlink:=p^.rlink; p^.rlink^.llink:=q; p^.llink:=q^.llink;
q^.llink^.rlink:=p; p^.rlink:=q; q^.llink:=p

17.(1)前两个语句改为：
p.llink^.rlink<- p^.rlink;
p^.rlink^.llink<- p^.llink;
(2)后三个语句序列应改为：
q^.rlink<- p^.rlink;//以下三句的顺序不能变
p^.rlink^.llink<- q;
p^.rlink<- q;

18. mp 是一个过程，其内嵌套有过程 subp。
subp(s,q)的作用是构造从 s 到 q 的循环链表。
subp(pa,pb)调用结果是将 pa 到 pb 的前驱构造为循环链表。
subp(pb,pa)调用结果是将 pb 到 pa 的前驱（指在 L 链表中，并非刚构造的 pa 循环链表中）构造为循环链表。

总之，两次调用将 L 循环链表分解为两个。第一个循环链表包含从 pa 到 pb 的前驱,L 中除刚构造的 pa 到 pb 前驱外的结点形成第二个循环链表。

19. 在指针 p 所指结点前插入结点 s 的语句如下：
s->pre=p->pre; s->next=p; p->pre->next=s; p->pre=s;

20. (A) f1<>NIL 并且 f2<>NIL
(B) f1↑.data < f2↑.data
(C) f2↑.data<f1↑.data
(D) f3↑.data<f1↑.data
(E) f1<- f1↑.link 或 f2=f2↑.link;

21. 1）本算法功能是将双向循环链表结点的数据域按值自小到大排序，成为非递减（可能包括数据域值相等的结点）有序双向循环链表。
2）(1)r->prior=q->prior;//将 q 结点摘下，以便插入到适当位置。
(2)p->next->prior=q;//（2）（3）将 q 结点插入
(3)p->next=q;
(4)r=r->next;或 r=q->next;//后移指针，再将新结点插入到适当位置。

五、 算法设计题

1. [题目分析]因为两链表已按元素值递增次序排列，将其合并时，均从第一个结点起进行比较，将小的链入链表中，同时后移链表工作指针。该问题要求结果链表按元素值递减次序排列。故在合并的同时，将链表结点逆置。

```
LinkedList Union(LinkedList la,lb)
//la,lb 分别是带头结点的两个单链表的头指针，链表中的元素值按递增序排列，本算法
  将两链表合并成一个按元素值递减次序排列的单链表。
```

```
{ pa=la->next; pb=lb->next;//pa，pb 分别是链表 la 和 lb 的工作指针
  la->next=null;            //la 作结果链表的头指针，先将结果链表初始化为空。
  while(pa!=null && pb!=null) //当两链表均不为空时作
    if(pa->data<=pb->data)
      { r=pa->next;           //将 pa 的后继结点暂存于 r。
        pa->next=la->next;    //将 pa 结点链于结果表中，同时逆置。
        la->next=pa;
        pa=r;                 //恢复 pa 为当前待比较结点。
       }
     else
      {r=pb->next;// 将 pb 的后继结点暂存于 r。
       pb->next=la->next; //将 pb 结点链于结果表中，同时逆置。
       la->next=pb;
       pb=r; //恢复 pb 为当前待比较结点。
       }
    while(pa!=null) //将 la 表的剩余部分链入结果表，并逆置。
     {r=pa->next; pa->next=la->next; la->next=pa; pa=r; }
    while(pb!=null)
     {r=pb->next; pb->next=la->next; la->next=pb; pb=r; }
  }//算法 Union 结束。
```

[算法讨论] 上面两链表均不为空的表达式也可简写为 while(pa&&pb),两递增有序表合并成递减有序表时，上述算法是边合并边逆置。也可先合并完，再作链表逆置。后者不如前者优化。算法中最后两个 while 语句，不可能执行两个，只能二者取一，即哪个表尚未到尾，就将其逆置到结果表中，即将剩余结点依次前插入到结果表的头结点后面。

与本题类似的其它题解答如下：

（1）[问题分析] 与上题类似，不同之处在于：一是链表无头结点，为处理方便，给加上头结点，处理结束再删除之；二是数据相同的结点，不合并到结果链表中；三是 hb 链表不能被破坏，即将 hb 的结点合并到结果链表时，要生成新结点。

```
LinkedList Union(LinkedList ha, hb)
//ha 和 hb 是两个无头结点的数据域值递增有序的单链表，本算法将 hb 中并不出现在 ha 中的数据合并到 ha 中，合并中不能破坏 hb 链表。
{LinkedList la;
 la=(LinkedList)malloc(sizeof(LNode));
 la->next=ha;//申请头结点，以便操作。
 pa=ha;       //pa 是 ha 链表的工作指针
 pb=hb;       //pb 是 hb 链表的工作指针
 pre=la;      //pre 指向当前待合并结点的前驱。
 while(pa&&pb)
   if(pa->data<pb->data)//处理 ha 中数据
     {pre->next=pa;pre=pa;pa=pa->next;}
   else if(pa->data>pb->data)//处理 hb 中数据。
      {r=(LinkedList)malloc(sizeof(LNode));//申请空间
       r->data=pb->data;  pre->next=r;
```

```
        pre=r;//将新结点链入结果链表。
        pb=pb->next;//hb 链表中工作指针后移。
       }
      else//处理 pa->data=pb->data;
        {pre->next=pa;  pre=pa;
         pa=pa->next;//两结点数据相等时，只将 ha 的数据链入。
         pb=pb->next;//不要 hb 的相等数据
        }
    if(pa!=null)pre->next=pa;//将两链表中剩余部分链入结果链表。
    else pre->next=pb;
    free(la);//释放头结点.ha,hb 指针未被破坏。
    }//算法 nion 结束。
```

(2)本题与上面两题类似，要求结果指针为 lc，其核心语句段如下：

```
pa=la->next;pb=hb->next;
lc=(LinkedList )malloc(sizeof(LNode));
pc=lc;//pc 是结果链表中当前结点的前驱
while(pa&&pb)
  if(pa->data<pb->data)
    {pc->next=pa;pc=pa;pa=pa->next;}
  else {pc->next=pb;pc=pb;pb=pb->next;}
if(pa)pc->next=pa; else pc->next=pb;
free(la);free(lb);//释放原来两链表的头结点。
```

算法时间复杂度为 O（m+n），其中 m 和 n 分别为链表 la 和 lb 的长度。

2.[题目分析]本组题有 6 个，本质上都是链表的合并操作，合并中有各种条件。与前组题不同的是，叙述上是用线性表代表集合，而操作则是求集合的并、交、差（A∪B，A∩B，A-B）等。

本题与上面 1.（2）基本相同，不同之处 1.（2）中链表是“非递减有序”，（可能包含相等元素），本题是元素“递增有序”（不准有相同元素）。因此两表中合并时，如有元素值相等元素，则应删掉一个。

```
LinkedList Union(LinkedList ha,hb)
//线性表 A 和 B 代表两个集合，以链式存储结构存储，元素递增有序。ha 和 hb 分别
  是其链表的头指针。本算法求 A 和 B 的并集 A∪B，仍用线性表表示，结果链表元
  素也是递增有序。
  { pa=ha->next;pb=hb->next;//设工作指针 pa 和 pb。
    pc=ha;//pc 为结果链表当前结点的前驱指针。
    while(pa&&pb)
      if(pa->data<pb->data)
        {pc->next=pa;pc=pa;pa=pa->next;}
      else if(pa->data>pb->data)
             {pc->next=pb;pc=pb;pb=pb->next;}
           else//处理 pa->data=pb->data.
             {pc->next=pa;pc=pa;pa=pa->next;
              u=pb;pb=pb->next;free(u);}
      if(pa) pc->next=pa;//  若 ha 表未空，则链入结果表。
```

```
        else pc->next=pb;//若 hb 表未空，则链入结果表。
       free(hb);  //释放 hb 头结点
       return(ha);
     }//算法 Union 结束。
```

与本题类似的其它几个题解答如下：

(1) 解答完全同上 2。

(2) 本题是求交集，即只有同时出现在两集合中的元素才出现在结果表中。其核心语句段如下：

```
pa=la->next;pb=lb->next;//设工作指针 pa 和 pb;
pc=la;//结果表中当前合并结点的前驱的指针。
while(pa&&pb)
  if(pa->data==pb->data)//交集并入结果表中。
   { pc->next=pa;pc=pa;pa=pa->next;
     u=pb;pb=pb->next;free(u);}
  else if(pa->data<pb->data) {u=pa;pa=pa->next;free(u);}
       else {u=pb; pb=pb->next; free(u);}
  while(pa){ u=pa; pa=pa->next; free(u);}// 释放结点空间
  while(pb) {u=pb; pb=pb->next; free(u);}//释放结点空间
  pc->next=null;//置链表尾标记。
  free(lb);   //注： 本算法中也可对 B 表不作释放空间的处理
```

（3）本题基本与（2）相同，但要求无重复元素，故在算法中，待合并结点数据要与其前驱比较，只有在与前驱数据不同时才并入链表。其核心语句段如下。

```
pa=L1->next;pb=L2->next;//pa、pb 是两链表的工作指针。
pc=L1;//L1 作结果链表的头指针。
while(pa&&pb)
  if(pa->data<pb->data) {u=pa;pa=pa->next;free(u);}//删除 L1 表多余元素
  else if (pa->data>pb->data) pb=pb->next; //pb 指针后移
  else      //处理交集元素
    {if(pc==L1) {pc->next=pa;pc=pa;pa=pa->next;} //处理第一个相等的元素。
      else if(pc->data==pa->data){ u=pa;pa=pa->next;free(u);} //重复元素不进入 L1 表。
           else{ pc->next=pa;pc=pa;pa=pa->next;} //交集元素并入结果表。
    } //while
  while(pa) {u=pa;pa=pa->next;free(u);} // 删 L1 表剩余元素
  pc->next=null;  //置结果链表尾。
```

注： 本算法中对 L2 表未作释放空间的处理。

（4） 本题与上面（3）算法相同，只是结果表要另辟空间。

（5） [题目分析]本题首先求 B 和 C 的交集，即求 B 和 C 中共有元素，再与 A 求并集，同时删除重复元素，以保持结果 A 递增。

```
LinkedList union(LinkedList A,B,C)
//A,B 和 C 均是带头结点的递增有序的单链表，本算法实现 A= A∪（B∩C），使求解结构保持递增有序。
{pa=A->next;pb=B->next;pc=C->next;//设置三个工作指针。
```

```
pre=A;  //pre 指向结果链表中当前待合并结点的前驱。
if(pa->data<pb->data||pa->data<pc->data)//A 中第一个元素为结果表的第一元素。
   {pre->next=pa;pre=pa;pa=pa->next;}
else{while(pb&&pc)            //找 B 表和 C 表中第一个公共元素。
       if(pb->data<pc->data)pb=pb->next;
       else if(pb->data>pc->data)pc=pc->next;
            else break;     //找到 B 表和 C 表的共同元素就退出 while 循环。
     if(pb&&pc)// 因共同元素而非 B 表或 C 表空而退出上面 while 循环。
       if(pa->data>pb->data)//A 表当前元素值大于 B 表和 C 表的公共元素,先将 B 表元素链入。
        {pre->next=pb;pre=pb;pb=pb->next;pc=pc->next;}// B,C 公共元素为结果表第一元素。
     }//结束了结果表中第一元素的确定
  while(pa&&pb&&pc)
    {while(pb&&pc)
       if(pb->data<pc->data) pb=pb->next;
       else if(pb->data>pc->data) pc=pc->next;
            else break;  //B 表和 C 表有公共元素。
     if(pb&&pc)
       {while(pa&&pa->data<pb->data)     //先将 A 中小于 B,C 公共元素部分链入。
          {pre->next=pa;pre=pa;pa=pa->next;}

if(pre->data!=pb->data){pre->next=pb;pre=pb;pb=pb->next;pc=pc->next;}
          else{pb=pb->next;pc=pc->next;}// 若 A 中已有 B,C 公共元素,则不再存入结果表。
       }
    }// while(pa&&pb&&pc)
   if(pa) pre->next=pa;  //当 B,C 无公共元素(即一个表已空),将 A 中剩余链入。
  }//算法 Union 结束
```

[算法讨论]本算法先找结果链表的第一个元素，这是因为题目要求结果表要递增有序（即删除重复元素)。这就要求当前待合并到结果表的元素要与其前驱比较。由于初始 pre=A（头结点的头指针)，这时的 data 域无意义，不能与后继比较元素大小，因此就需要确定第一个元素。当然，不要这样作，而直接进入下面循环也可以，但在链入结点时，必须先判断 pre 是否等于 A，这占用了过多的时间。因此先将第一结点链入是可取的。

算法中的第二个问题是要求时间复杂度为 O(|A|+|B|+|C|)。这就要求各个表的工作指针只能后移（即不能每次都从头指针开始查找)。本算法满足这一要求。

最后一个问题是，当 B，C 有一表为空（即 B 和 C 已无公共元素时)，要将 A 的剩余部分链入结果表。

3.[题目分析]循环单链表 L1 和 L2 数据结点个数分别为 m 和 n ，将二者合成一个循环单链表时，需要将一个循环链表的结点（从第一元素结点到最后一个结点）插入到另一循环链表的第一元素结点前即可。题目要求“用最快速度将两表合并“，因此应找结点个数少的链表查其尾结点。

```
LinkedList Union(LinkedList L1,L2;int m,n)
```

```
//L1 和 L2 分别是两循环单链表的头结点的指针，m 和 n 分别是 L1 和 L2 的长度。
//本算法用最快速度将 L1 和 L2 合并成一个循环单链表。
{if(m<0||n<0) {printf(“表长输入错误\n”); exit(0);}
 if(m<n)                  //若 m<n，则查 L1 循环单链表的最后一个结点。
  {if(m==0)return(L2);//L1 为空表。
   else{p=L1;
        while(p->next!=L1) p=p->next;//查最后一个元素结点。
        p->next=L2->next;//将 L1 循环单链表的元素结点插入到 L2 的第一元素结点前。
        L2->next=L1->next;
        free(L1);//释放无用头结点。
       }
  }//处理完 m<n 情况
 else// 下面处理 L2 长度小于等于 L1 的情况
    {if(n==0)return(L1);//L2 为空表。
     else{p=L2;
          while(p->next!=L2) p=p->next;//查最后元素结点。
          p->next=L1->next;//将 L2 的元素结点插入到 L1 循环单链表的第一元素结点前。
          L1->next=L2->next;
          free(L2);//释放无用头结点。
    }
 }//算法结束。
```

类似本题叙述的其它题解答如下：

（1）[题目分析]本题将线性表 la 和 lb 连接，要求时间复杂度为 O（1），且占用辅助空间尽量小。应该使用只设尾指针的单循环链表。

```
 LinkedList Union(LinkedList la,lb)
 //la 和 lb 是两个无头结点的循环单链表的尾指针，本算法将 lb 接在 la 后，成为一个单循环链表。
 { q=la->next;           //q 指向 la 的第一个元素结点。
   la->next=lb->next;    //将 lb 的最后元素结点接到 lb 的第一元素。
   lb->next=q;           //将 lb 指向 la 的第一元素结点，实现了 lb 接在 la 后。
   return(lb);           //返回结果单循环链表的尾指针 lb。
 }//算法结束。
```

[算法讨论]若循环单链表带有头结点，则相应算法片段如下：

```
q=lb->next;          //q 指向 lb 的头结点；
lb->next=la->next;  //lb 的后继结点为 la 的头结点。
la->next=q->next;   //la 的后继结点为 lb 的第一元素结点。
free(q);             //释放 lb 的头结点
return(lb);   //返回结果单循环链表的尾指针 lb。
```

（2）[题目分析]本题要求将单向链表 ha 和单向循环链表 hb 合并成一个单向链表，要求算法所需时间与链表长度无关，只有使用带尾指针的循环单链表，这样最容易找到链表的首、尾结点，将该结点序列插入到单向链表第一元素之前即可。

其核心算法片段如下（设两链表均有头结点）

```
q=hb->next;          //单向循环链表的表头指针
hb->next=ha->next;  //将循环单链表最后元素结点接在 ha 第一元素前。
ha->next=q->next;   //将指向原单链表第一元素的指针指向循环单链表第一结点
free(q);             //释放循环链表头结点。
```

若两链表均不带头结点，则算法片段如下：

```
q=hb->next;          //q 指向 hb 首元结点。
hb->next=ha;         //hb 尾结点的后继是 ha 第一元素结点。
ha=q;                //头指针指向 hb 的首元结点。
```

4.[题目分析]顺序存储结构的线性表的插入，其时间复杂度为 O（n），平均移动近一半的元素。线性表 LA 和 LB 合并时，若从第一个元素开始，一定会造成元素后移，这不符合本题“高效算法”的要求。另外，题中叙述“线性表空间足够大”也暗示出另外合并方式，即应从线性表的最后一个元素开始比较，大者放到最终位置上。设两线性表的长度各为 m 和 n ，则结果表的最后一个元素应在 m+n 位置上。这样从后向前，直到第一个元素为止。

```
PROC Union(VAR LA:SeqList;LB:SeqList)
//LA 和 LB 是顺序存储的非递减有序线性表，本算法将 LB 合并到 LA 中，元素仍非递减有序。
    m:=LA.last;n:=LB.last;//m，n 分别为线性表 LA 和 LB 的长度。
    k:=m+n;     //k 为结果线性表的工作指针（下标）。
    i:=m;j:=n; //i，j 分别为线性表 LA 和 LB 的工作指针（下标）。
    WHILE(i>0)AND(j>0)DO
      IF LA.elem[i]>=LB.elem[j]
      THEN[LA.elem[k]:=LA.elem[i];k:=k-1;i:=i-1;]
      ELSE[LA.elem[k]:=LB.elem[j];k:=k-1;j:=j-1;]
    WHILE(j>0) DO [LA.elem[k]:=LB.elem[j];k:=k-1;j:=j-1;]
    LA.last:=m+n;
 ENDP;
```

[算法讨论]算法中数据移动是主要操作。在最佳情况下（LB 的最小元素大于 LA 的最大元素），仅将 LB 的 n 个元素移（拷贝）到 LA 中，时间复杂度为 O（n），最差情况，LA 的所有元素都要移动，时间复杂度为 O（m+n）。因数据合并到 LA 中，所以在退出第一个 WHILE 循环后，只需要一个 WHILE 循环，处理 LB 中剩余元素。第二个循环只有在 LB 有剩余元素时才执行，而在 LA 有剩余元素时不执行。本算法利用了题目中“线性表空间足够大”的条件，“最大限度的避免移动元素”，是“一种高效算法”。

5.[题目分析]本题实质上是一个排序问题，要求“不得使用除该链表结点以外的任何链结点空间”。链表上的排序采用直接插入排序比较方便，即首先假定第一个结点有序，然后，从第二个结点开始，依次插入到前面有序链表中，最终达到整个链表有序。

```
LinkedList LinkListSort(LinkedList list)
//list 是不带头结点的线性链表，链表结点构造为 data 和 link 两个域，data 是数据域，link 是指针域。本算法将该链表按结点数据域的值的大小，从小到大重新链接。
    {p=list->link;    //p 是工作指针，指向待排序的当前元素。
     list->link=null;//假定第一个元素有序，即链表中现只有一个结点。
     while(p!=null)
      {r=p->link;     //r 是 p 的后继。
       q=list;
```

```
        if(q->data>p->data)//处理待排序结点 p 比第一个元素结点小的情况。
          {p->link=list;
           list=p;//链表指针指向最小元素。
           }
        else//查找元素值最小的结点。
           {while(q->link!=null&&q->link->data<p->data)q=q->link;
            p->link=q->link;//将当前排序结点链入有序链表中。
            q->link=p;   }
        p=r;//p 指向下个待排序结点。
      }
    }
```

[算法讨论]算法时间复杂度的分析与用顺序存储结构时的情况相同。但顺序存储结构将第 i(i>1)个元素插入到前面第 1 至第 i-1 个元素的有序表时，是将第 i 个元素先与第 i-1 个元素比较。而在链表最佳情况均是和第一元素比较。两种存储结构下最佳和最差情况的比较次数相同，在链表情况下，不移动元素，而是修改结点指针。

另一说明是，本题中线性链表 list 不带头结点，而且要求“不得使用除该链表以外的任何链结点空间“，所以处理复杂，需要考虑当前结点元素值比有序链表第一结点的元素值还小的情况，这时要修改链表指针 list。如果 list 是头结点的指针，则相应处理要简单些，其算法片段如下：

```
p=list->link;//p 指向第一元素结点。
list->link=null;//有序链表初始化为空
while(p!=null)
  {r=p->link;//保存后继
   q=list;
   while(q->link!=null && q->link->data<p->data)q=q->link;
   p->link=q->link;
   q->link=p;
   q=r;
  }
```

6.[题目分析]本题明确指出单链表带头结点，其结点数据是正整数且不相同，要求利用直接插入原则把链表整理成递增有序链表。这就要求从第二结点开释，将各结点依次插入到有序链表中。

```
LinkedList LinkListInsertSort(LinkedList la)
 //la 是带头结点的单链表，其数据域是正整数。本算法利用直接插入原则将链表整理成递增的有序链表。
 {if(la->next!=null)//链表不为空表。
   {p=la->next->next;//p 指向第一结点的后继。
    la->next->next=null;//直接插入原则认为第一元素有序，然后从第二元素起依次插入。
    while(p!=null)
      {r=p->next;//暂存 p 的后继。
       q=la;
       while(q->next!=null&&q->next->data<p->data)q=q->next;//查找插入位置。
       p->next=q->next;//将 p 结点链入链表。
```

```
        q->next=p;
        p=r;
    }
```

与本题有类似叙述的题的解答：

（1）本题也是链表排序问题，虽没象上题那样明确要求“利用直接插入的原则”来排序，仍可用上述算法求解，这里不再赘述。

7.[题目分析]本题要求将一个链表分解成两个链表，两个链表都要有序，两链表建立过程中不得使用 NEW 过程申请空间，这就是要利用原链表空间，随着原链表的分解，新建链表随之排序。

```
PROC discreat(VAR listhead,P,Q:linkedlist)
```

//listhead 是单链表的头指针，链表中每个结点由一个整数域 DATA 和指针域 NEXT 组成。本算法将链表 listhead 分解成奇数链表和偶数链表，分解由 P 和 Q 指向，且 P 和 Q 链表是有序的。

```
    P:=NIL;Q:=NIL;//P 和 Q 链表初始化为空表。
  s:=listhead;
  WHILE(s<>NIL)DO
    [r:=s^.NEXT;                    //暂存 s 的后继。
     IF s^.DATA DIV 2=0             //处理偶数。
     THEN IF P=NIL THEN[P:=s;P^.NEXT:=NIL;]    //第一个偶数链结点。
          ELSE[pre:=P;
               IF pre^.DATA>s^.DATA THEN[s^.NEXT:=pre;P:=s;//插入当前最小值结点修改头指针]
               ELSE[WHILE pre^.NEXT<>NIL DO
                      IF pre^.NEXT^.DATA<s^.DATA THEN pre:=pre^.NEXT;//查找插入位置。
                      s^.NEXT:=pre^.NEXT;       //链入此结点。
                      pre^.NEXT:=s;
                    ]
                    ]
       ELSE//处理奇数链。
         IF Q=NIL THEN[Q:=s;Q^.NEXT:=NIL;]     //第一奇数链结点。
         ELSE[pre:=Q;
              IF pre^.DATA>s^.DATA THEN[s^.NEXT:=pre; Q:=s;  ]//修改头指针。
               ELSE[WHILE pre^.NEXT<>NIL DO //查找插入位置。
                      IF pre^.NEXT^.DATA<s^.DATA THEN pre:=pre^.NEXT;
                     s^.NEXT:=pre^.NEXT;//链入此结点。
                     pre^.NEXT:=s;
                    ]
             ]//结束奇数链结点
      s:=r;//s 指向新的待排序结点。
  ]//结束“WHILE(s<>NIL)DO”
ENDP; //结束整个算法。
```

[算法讨论]由于算法要求“不得使用 NEW 过程申请空间，也没明确指出链表具有头结点，所以上述算法复杂些，它可能需要在第一个结点前插入新结点，即链表的头指针会发生变化。

如有头结点，算法不必单独处理在第一个结点前插入结点情况，算法会规范统一，下面的（1）是处理带头结点的例子。算法中偶数链上结点是靠数据整除 2 等于 0（DATA DIV 2=0）判断的。

类似本题的其它题解答如下：

（1）[题目分析]本题基本类似于上面第 7 题，不同之处有二。一是带头结点，二是分解后的两个链表，一个是数据值小于 0，另一个是数据值大于 0。由于没明确要求用类 PASCAL 书写算法，故用 C 书写如下。

```
void DisCreat1(LinkedList A)
//A 是带头结点的单链表，链表中结点的数据类型为整型。本算法将 A 分解成两个单链表 B 和 C，B 中结点的数据小于零，C 中结点的数据大于零。
{B=A;
 C=(LinkedList )malloc(sizeof(LNode));//为 C 申请结点空间。
 C->next=null      //C 初始化为空表。
 p=A->next;        //p 为工作指针。
 B->next=null;     //B 表初始化。
 while(p!=null)
   {r=p->next;     //暂存 p 的后继。
    if (p->data<0)//小于 0 的放入 B 表。
      {p->next=B->next; B->next=p; }//将小于 0 的结点链入 B 表。
    else {p->next=C->next; C->next=p; }
    p=r;//p 指向新的待处理结点。
   }
}//算法结束。
```

[算法讨论]因为本题并未要求链表中结点的数据值有序，所以算法中采取最简单方式：将新结点前插到头结点后面（即第一元素之前）。

（2）本题同上面第 7 题，除个别叙述不同外，本质上完全相同，故不再另作解答。

（3）[题目分析]本题中的链表有头结点，分解成表 A 和表 B，均带头结点。分解后的 A 表含有原表中序号为奇数的元素，B 表含有原 A 表中序号为偶数的元素。由于要求分解后两表中元素结点的相对顺序不变，故采用在链表尾插入比较方便，这使用一指向表尾的指针即可方便实现。

```
void DisCreat3(LinkedList A)
//A 是带头结点的单链表，本算法将其分解成两个带头结点的单链表，A 表中含原表中序号为奇数
//的结点，B 表中含原表中序号为偶数的结点。链表中结点的相对顺序同原链表。
{i=0;//i 记链表中结点的序号。
 B=(LinkedList)malloc(sizeof(LNode));//创建 B 表表头。
 B->next=null;     //B 表的初始化。
 LinkedList ra,rb;//ra 和 rb 将分别指向将创建的 A 表和 B 表的尾结点。
 ra=A;rb=B;
 p=A->next;        //p 为链表工作指针，指向待分解的结点。
 A->next=null;     //置空新的 A 表
 while(p!=null)
 {r=p->next;       //暂存 p 的后继。
  i++;
```

```
    if(i%2==0)        //处理原序号为偶数的链表结点。
      {p->next=rb->next;//在B表尾插入新结点;
       rb->next=p; rb=p;//rb指向新的尾结点;
      }
    else//处理原序号为奇数的结点。
      {p->next=ra->next; ra->next=p; ra=p; }
    p=r;            //将p恢复为指向新的待处理结点。
  }//算法结束
```

8. [题目分析]题目要求重排n个元素且以顺序存储结构存储的线性表，使得所有值为负数的元素移到正数元素的前面。这可采用快速排序的思想来实现，只是提出暂存的第一个元素（枢轴）并不作为以后的比较标准，比较的标准是元素是否为负数。

```
int Rearrange (SeqList a; int n)
//a是具有n个元素的线性表，以顺序存储结构存储，线性表的元素是整数。本算法重排线性表a，
//使所有值为负数的元素移到所有值为正数的数的前面。
{i=0; j=n-1;     // i,j为工作指针（下标），初始指向线性表a的第1个和第n个元素。
  t=a[0];      //暂存枢轴元素。
  while(i<j)
      {while(i<j && a[j]>=0) j--;  // 若当前元素为大于等于零，则指针前移。
       if(i<j){a[i]=a[j];i++;}    //  将负数前移。
       while(i<j &&a[i]<0)i++;    // 当前元素为负数时指针后移。
       if(i<j) a[j--]=a[i];       // 正数后移。
      }
      a[i]=t;    //将原第一元素放到最终位置。
}
```

[算法讨论] 本算法时间复杂度为 O（n）。算法只是按题目要求把正负数分开，如要求统计负数和大于等于零的个数,则最后以t来定。如t为负数,则0至i共i+1个负数,n-1-i个正数（包括零）。另外，题目并未提及零的问题，笔者将零放到正数一边。对此问题的扩充是若元素包含正数、负数和零，并要求按负数、零、正数的顺序重排线性表，统计负数、零、正数的个数。请读者利用上面解题思想自行解答。

类似本题的选了5个题，其解答如下：

（1）与上面第8题不同的是，这里要求以a_n为参考元素，将线性表分成左右两部分。左半部分的元素都小于等于a_n，右半部分的元素都大于a_n，a_n位于分界位置上。其算法主要片段语句如下:

```
i=1;j=n;
t=a[n];   //暂存参考元素。
while(i<j)
   {while(i<j && a[i]<=t)  i++;  //当前元素不大于参考元素时，指针i后移。
    if(i<j)  a[j--]=a[i];        //将大于参考元素的元素后移。
    while(i<j && a[j]>t)  j--;   //当前元素大于参考元素时指针前移。
    if(i<j)  a[i++]=a[j];        //将小于参考元素的当前元素前移。
   }
a[i]=t; //参考元素置于分界位置。
```

(2) [题目分析]本题要求将线性表A分成B和C两个表，表B和表C不另占空间，而是利用表A的空间，其算法与第8题相同。这里仅把表B和表C另设空间的算法解答如下：

```
void  Rearrange2(int A[],B[],C[])
  //线性表A有n个整型元素，顺序存储。本算法将A拆成B和C 两个表，B中存放大于
  //等于零的元素，C中存放小于零的元素。
  {i=0;       //i，j，k是工作指针，分别指向A、B和C表的当前元素。
   j=k=-1;    //j，k初始化为-1。
   while(i<n)
     {if(A[i]<0)  C[++k]=A[i++];  //将小于零的元素放入C表。
      else  B[++j]=A[i++];        //将大于零的元素放入B表。
```

[算法讨论]本题用一维数组存储线性表，结果线性表B和C中分别有j+1和k+1个元素。若采用教材中的线性表，则元素的表示作相应改变，例如 A.elem[i]，而最后 B和C表应置上表的长度，如B.length=j和C.length=k。

(3) 本题与第8题本质上相同，第8题要求分开正数和负数，这里要求分开奇数和偶数，判别方式是a[i]%2==0，满足时为偶数，反之为奇数。

(4) 本题与第8题相同，只是叙述不同。

(5) 本题与第8题基本相同，不同之处在于这里的分界元素是整数19（链表中并不要求一定有19）。本题要求用标准pascal描述算法，如下所示。

```
TYPE  arr=ARRAY[1..1000]  OF  integer;
VAR   a：arr;
PROCEDURE  Rearrange5（VAR a：arr）;
//a是n（设n=1000）个整数组成的线性表，用一维数组存储。本算法将n个元素中所有大于等于19的整数放在所有小于19的整数之后。
VAR  i,j,t：integer;
BEGIN
   i:=1; j:=n; t:=a[1] ; //i,j指示顺序表的首尾元素的下标，t暂存分界元素
   WHILE（i<j）DO
      BEGIN
         WHILE （i<j）AND（a[j]>=19） DO  j:=j-1;
         IF（i<j）THEN  BEGIN  A[i]:=A[j]; i:=i+1  END;
         WHILE （i<j）AND（a[i] <19） DO  i:=i+1;
         IF（i<j）THEN  BEGIN  A[j]:=A[i]; j:=j-1  END;
      END;
   a[i]:=t;
END;
```

[算法讨论] 分界元素t放入a[i]，而不论它的值如何。算法中只用了一个t中间变量，符合空间复杂度O(1)的要求。算法也满足时间复杂度O(n)的要求。

9.[题目分析] 本题要求在单链表中删除最小值结点。单链表中删除结点，为使结点删除后不出现“断链”，应知道被删结点的前驱。而“最小值结点”是在遍历整个链表后才能知道。所以算法应首先遍历链表，求得最小值结点及其前驱。遍历结束后再执行删除操作。

```
LinkedList Delete（LinkedList L）
//L是带头结点的单链表，本算法删除其最小值结点。
{p=L->next;    //p为工作指针。指向待处理的结点。假定链表非空。
```

```
    pre=L;          //pre 指向最小值结点的前驱。
    q=p;            //q 指向最小值结点，初始假定第一元素结点是最小值结点。
    while (p->next!=null)
     {if (p->next->data<q->data) {pre=p; q=p->next; }     //查最小值结点
      p=p->next;        //指针后移。
     }
    pre->next=q->next; //从链表上删除最小值结点
    free (q);           //释放最小值结点空间
  }//结束算法 delete。
```

[算法讨论] 算法中函数头是按本教材类 C 描述语言书写的。原题中 **void** delete (linklist &L)，是按 C++的“引用”来写的，目的是实现变量的“传址”，克服了 C 语言函数传递只是“值传递”的缺点。

10.[题目分析] 本题要求将链表中数据域值最小的结点移到链表的最前面。首先要查找最小值结点。将其移到链表最前面，实质上是将该结点从链表上摘下（不是删除并回收空间），再插入到链表的最前面。

```
LinkedList  delinsert (LinkedList  list)
  //list 是非空线性链表，链结点结构是 (data，link)，data 是数据域，link 是链域。
  //本算法将链表中数据域值最小的那个结点移到链表的最前面。
 {p=list->link; //p 是链表的工作指针
  pre=list;      //pre 指向链表中数据域最小值结点的前驱。
  q=p;           //q 指向数据域最小值结点，初始假定是第一结点
  while  (p->link!=null)
    {if (p->link->data<q->data) {pre=p; q=p->link; } //找到新的最小值结点;
     p=p->link;
    }
  if (q!=list->link)    //若最小值是第一元素结点，则不需再操作
    {pre->link=q->link;    //将最小值结点从链表上摘下;
     q->link= list->link; //将 q 结点插到链表最前面。
     list->link=q;
    }
 }//算法结束
```

[算法讨论] 算法中假定 list 带有头结点，否则，插入操作变为 q->link=list;list=q。

11.[题目分析] 知道双向循环链表中的一个结点，与前驱交换涉及到四个结点（p 结点，前驱结点，前驱的前驱结点，后继结点）六条链。

```
void  Exchange (LinkedList p)
//p 是双向循环链表中的一个结点，本算法将 p 所指结点与其前驱结点交换。
{q=p->llink;
 q->llink->rlink=p;     //p 的前驱的前驱之后继为 p
 p->llink=q->llink;     //p 的前驱指向其前驱的前驱。
 q->rlink=p->rlink;     //p 的前驱的后继为 p 的后继。
 q->llink=p;            //p 与其前驱交换
 p->rlink->llink=q;     //p 的后继的前驱指向原 p 的前驱
 p->rlink=q;            //p 的后继指向其原来的前驱
```

}//算法 exchange 结束。

12.[题目分析] 顺序存储的线性表递增有序，可以顺序查找，也可折半查找。题目要求“用最少的时间在表中查找数值为 x 的元素”，这里应使用折半查找方法。

```
void  SearchExchangeInsert(ElemType a[]; ElemType x)
//a 是具有 n 个元素的递增有序线性表，顺序存储。本算法在表中查找数值为 x 的元素，
如查到则与其后继交换位置；如查不到，则插入表中，且使表仍递增有序。
{ low=0; high=n-1;                    //low 和 high 指向线性表下界和上界的下标
 while(low<=high)
   {mid=(low+high)/2;                 //找中间位置
    if(a[mid]==x)  break;             //找到 x，退出 while 循环。
    else if  (a[mid]<x)  low=mid+1;  //到中点 mid 的右半去查。
        else high=mid-1;              //到中点 mid 的左部去查。
   }
 if(a[mid]==x && mid!=n) // 若最后一个元素与 x 相等，则不存在与其后继交换的
                                操作。
   {t=a[mid]; a[mid]=a[mid+1]; a[mid+1]=t; } // 数值 x 与其后继元素位置交换。
 if(low>high)       //查找失败，插入数据元素 x
   {for(i=n-1; i>high; i--)  a[i+1]=a[i]; //后移元素。
    a[i+1]=x; //插入 x。
   } //结束插入
}//结束本算法。
```

[算法讨论] 首先是线性表的描述。算法中使用一维数组 a 表示线性表，未使用包含数据元素的一维数组和指示线性表长度的结构体。若使用结构体，对元素的引用应使用 a.elem[i]。另外元素类型就假定是 ElemType，未指明具体类型。其次，C 中一维数组下标从 0 开始，若说有 n 个元素的一维数组，其最后一个元素的下标应是 n-1。第三，本算法可以写成三个函数，查找函数，交换后继函数与插入函数。写成三个函数显得逻辑清晰，易读。

13.[题目分析] 判断链表中数据是否中心对称，通常使用栈。将链表的前一半元素依次进栈。在处理链表的后一半元素时，当访问到链表的一个元素后，就从栈中弹出一个元素，两元素比较，若相等，则将链表中下一元素与栈中再弹出元素比较，直至链表到尾。这时若栈是空栈，则得出链表中心对称的结论；否则，当链表中一元素与栈中弹出元素不等时，结论为链表非中心对称，结束算法的执行。

```
int  dc(LinkedList  h，int n)
// h 是带头结点的 n 个元素单链表，链表中结点的数据域是字符。本算法判断链表是
                          否是中心对称。
{char s[]; int i=1; //i 记结点个数， s 字符栈
 p=h->next; //p 是链表的工作指针，指向待处理的当前元素。
 for(i=1; i<=n/2; i++) // 链表前一半元素进栈。
  {s[i]=p->data; p=p->next; }
 i--; //恢复最后的 i 值
 if(n%2==1) p=p->next; }  //若 n 是奇数，后移过中心结点。
 while(p!=null && s[i]==p->data){i--; p=p->next; }   //测试是否中心对称。
 if(p==null) return(1); //链表中心对称
 else  return(0);          //链表不中心对称
}//算法结束。
```

[算法讨论] 算法中先将“链表的前一半”元素（字符）进栈。当 n 为偶数时，前一半和后一半的个数相同；当 n 为奇数时，链表中心结点字符不必比较，移动链表指针到下一字符开始比较。比较过程中遇到不相等时，立即退出 **while** 循环，不再进行比较。

14. [题目分析] 在单链表中删除自第 i 个元素起的共 len 个元素，应从第 1 个元素起开始计数，记到第 i 个时开始数 len 个，然后将第 i-1 个元素的后继指针指向第 i+len 个结点，实现了在 A 链表中删除自第 i 个起的 len 个结点。这时应继续查到 A 的尾结点，得到删除元素后的 A 链表。再查 B 链表的第 j 个元素，将 A 链表插入之。插入和删除中应注意前驱后继关系，不能使链表“断链”。另外，算法中应判断 i，len 和 j 的合法性。

```
LinkedList  DelInsert (LinkedList heada, headb, int i, j, len)
//heada 和 headb 均是带头结点的单链表。本算法删除 heada 链表中自第 i 个元素起的共 len 个元素，然后将单链表 heada 插入到 headb 的第 j 个元素之前。
{if (i<1 || len<1 || j<1) {printf ("参数错误\n"); exit (0); }//参数错，退出算法。
 p=heada; //p 为链表 A 的工作指针，初始化为 A 的头指针，查到第 i 个元素时，p 指向第 i-1 个元素
 k=0; //计数
 while (p!=null && k<i-1) //查找第 i 个结点。
    {k++; p=p->next; }
 if (p==null) {printf ("给的%d 太大\n",i); exit (0); } //i 太大，退出算法
 q=p->next; //q 为工作指针，初始指向 A 链表第一个被删结点。
 k=0;
while (q!=null && k<len) {k++; u=q, q=q->next; free (u); } //删除结点，后移指针。
if (k<len) {printf ("给的%d 太大\n",len); exit (0); }
p->next=q; //A 链表删除了 len 个元素。
if (heada->next!=null)  //heada->next=null 说明链表中结点均已删除，无需往 B 表插入
 {while (p->next!=null) p= p->next; //找 A 的尾结点。
  q=headb; //q 为链表 B 的工作指针。
  k=0; //计数
  while (q!=null && k<j-1)  //查找第 j 个结点。
       {k++; q= q->next; }       //查找成功时，q 指向第 j-1 个结点
  if (q==null) {printf ("给的%d 太大\n",j); exit (0); }
  p->next=q->next;       //将 A 链表链入
  q->next=heada->next;  //A 的第一元素结点链在 B 的第 j-1 个结点之后
}//if
free (heada); //释放 A 表头结点。
}//算法结束。
```

与本题类似的题的解答如下：

（1）本题与第 14 题基本相同，不同之处仅在于插入 B 链表第 j 个元素之前的，不是删除了 len 个元素的 A 链表，而是被删除的 len 个元素。按照上题，这 len 个元素结点中第一个结点的指针 p->next，查找从第 i 个结点开始的第 len 个结点的算法修改为：

```
k=1; q=p->next;              //q 指向第一个被删除结点
while (q!=null && k<len)    //查找成功时，q 指向自 i 起的第 len 个结点。
```

```
    {k++; q= q->next; }
  if (k<len) {printf ("给的%d 太大\n",len); exit (0); }
```

15.[题目分析] 在递增有序的顺序表中插入一个元素 x，首先应查找待插入元素的位置。因顺序表元素递增有序，采用折半查找法比顺序查找效率要高。查到插入位置后，从此位置直到线性表尾依次向后移动一个元素位置，之后将元素 x 插入即可。

```
void  Insert (ElemType A[], int size, ElemType x)
// A 是有 size 个元素空间目前仅有 num (num<size) 个元素的线性表。本算法将元素 x 插入到线性表中，并保持线性表的有序性。
   {low=1; high=num;  //题目要求下标从 1 开始
    while (low<=high) //对分查找元素 x 的插入位置。
      {mid= (low+high) /2;
       if (A[mid]==x) {low=mid+1; break; }
       else if (A[mid]>x) high=mid-1 ; else  low=mid+1 ;
       }
     for (i=num; i>=low; i--)  A[i+1]=A[i]; //元素后移。
     A[i+1]=x;  //将元素 x 插入。
    }算法结束。
```

[算法讨论] 算法中当查找失败（即线性表中无元素 x）时，变量 low 在变量 high 的右面（low=high+1)。移动元素从 low 开始，直到 num 为止。特别注意不能写成 for (i=low; i<=num; i++) A[i+1]=A[i]，这是一些学生容易犯的错误。另外，题中未说明若表中已有值为 x 的元素时不再插入，故安排在 A[mid]= =x 时，用 low (=mid+1) 记住位置，以便后面统一处理。查找算法时间复杂度为 0 (logn)，而插入时的移动操作时间复杂度为 0 (n)，若用顺序查找，则查找的时间复杂度亦为 0 (n)。

类似本题的其它题的解答：

(1)[题目分析] 本题与上面 15 题类似，不同之处是给出具体元素值，且让编写 turbo pascal 程序，程序如下：

```
PROGRAM  example (input, output);
TYPE  pointer=^node;
          node=RECORD
                 data: integer;
                 next: pointer;
                END;
VAR  head, q: pointer;
PROCEDURE  create (VAR la: pointer);
  VAR  x: integer;
       p,q: pointer;
  BEGIN
      new (la); la^.next:=NIL; {建立头结点。}
      read (x); q:=la; {q 用以指向表尾。}
      WHILE  NOT  EOF  DO  {建立链表}
        BEGIN
          new(p); p^.data:=x; p^.next:=q^.next; q^.next:=p; q:=p; read(x);
        END;
  END;
```

```
PROCEDURE  insert (VAR la: pointer; s: pointer);
VAR  p,q: pointer; found: boolean;
BEGIN
    p:= la^.next; {p 为工作指针。}
    q:=la; {q 为 p 的前驱指针。}
    found:=false;
    WHILE (p<>NIL) AND  NOT  found
      IF (p^.data<x) THEN  BEGIN  q:=p; p:= p^.next;  END
      ELSE  found:=true;
    s^.next:=p; {将 s 结点插入链表}
    q^.next:=s;
END;
BEGIN      {main}
    writeln ("请按顺序输入数据，建立链表")
    create (head);
    writeln ("请输入插入数据")
    new (q);
    readln (q^.data);
    insert (head, q);
END.{程序结束}
```

[程序讨论] 在建立链表时，输入数据依次为 12，13，21，24，28，30，42，键入 CTRL-Z，输入结束。"插入数据"输 26 即可。本题编写的是完整的 pascal 程序。

16. [题目分析] 将具有两个链域的单循环链表，改造成双向循环链表，关键是控制给每个结点均置上指向前驱的指针，而且每个结点的前驱指针置且仅置一次。

```
void  StoDouble (LinkedList la)
   //la 是结点含有 pre，data，link 三个域的单循环链表。其中 data 为数据域,pre 为空指针域，link 是指向后继的指针域。本算法将其改造成双向循环链表。
{while (la->link->pre==null)
  {la->link->pre=la;    //将结点 la 后继的 pre 指针指向 la。
   la=la->link;         //la 指针后移。
  }
}  //算法结束。
```

[算法讨论] 算法中没有设置变量记住单循环链表的起始结点，至少省去了一个指针变量。当算法结束时，la 恢复到指向刚开始操作的结点，这是本算法的优点所在。

17. [题目分析] 求两个集合 A 和 B 的差集 A-B，即在 A 中删除 A 和 B 中共有的元素。由于集合用单链表存储，问题变成删除链表中的结点问题。因此，要记住被删除结点的前驱，以便顺利删除被删结点。两链表均从第一元素结点开始，直到其中一个链表到尾为止。

```
void  Difference (LinkedList  A, B, *n)
//A 和 B 均是带头结点的递增有序的单链表，分别存储了一个集合，本算法求两集合的差集，存储于单链表 A 中，*n 是结果集合中元素个数，调用时为 0
{p=A->next;                    //p 和 q 分别是链表 A 和 B 的工作指针。
 q=B->next;  pre=A;            //pre 为 A 中 p 所指结点的前驱结点的指针。
 while (p!=null && q!=null)
   if (p->data<q->data) {pre=p; p=p->next; *n++; }  // A 链表中当前结点指针后移。
```

```
    else if (p->data>q->data) q=q->next;        //B 链表中当前结点指针后移。
        else {pre->next=p->next;                //处理 A，B 中元素值相同的结点，应删除。
              u=p;  p=p->next;  free (u); }     //删除结点
```

18.[题目分析] 本题要求对单链表结点的元素值进行运算，判断元素值是否等于其序号的平方减去其前驱的值。这里主要技术问题是结点的序号和前驱及后继指针的正确指向。

```
 int  Judge (LinkedList  la)
//la 是结点的元素为整数的单链表。本算法判断从第二结点开始，每个元素值是否等于其序号的平方减去其前驱的值，如是返回 true；否则，返回 false。
{p=la->next->next; //p 是工作指针，初始指向链表的第二项。
 pre=la->next;     //pre 是 p 所指结点的前驱指针。
 i=2;              //i 是 la 链表中结点的序号，初始值为 2。
 while (p!=null)
  if (p->data==i*i-pre->data) {i++; pre=p; p=p->next; } //结点值间的关系符合题目要求
  else  break;                              //当前结点的值不等于其序号的平方减去前驱的值。
 if (p!=null) return (false);               //未查到表尾就结束了。
 else  return (true);                       //成功返回。
}//算法结束。
```

[算法讨论]本题不设头结点也无影响。另外，算法中还可节省前驱指针 pre，其算法片段如下：

```
  p=la; //假设无头结点，初始 p 指向第一元素结点。
  i=2;
  while (p->next!=null)                 //初始 p->next 指向第二项。
    if (p->next->data= =i*i-p->data)
       {i++; p=p->next; }
  if (p->next!=null) return (false); //失败
  else return (true);                //成功
```

19.[题目分析] 本题实质上是一个模式匹配问题，这里匹配的元素是整数而不是字符。因两整数序列已存入两个链表中，操作从两链表的第一个结点开始，若对应数据相等，则后移指针；若对应数据不等，则 A 链表从上次开始比较结点的后继开始，B 链表仍从第一结点开始比较，直到 B 链表到尾表示匹配成功。A 链表到尾 B 链表未到尾表示失败。操作中应记住 A 链表每次的开始结点，以便下趟匹配时好从其后继开始。

```
  int  Pattern (LinkedList A, B)
  //A 和 B 分别是数据域为整数的单链表，本算法判断 B 是否是 A 的子序列。如是，返回 1；否则，返回 0 表示失败。
  {p=A;     //p 为 A 链表的工作指针，本题假定 A 和 B 均无头结点。
   pre=p;   //pre 记住每趟比较中 A 链表的开始结点。
   q=B;     //q 是 B 链表的工作指针。
   while (p && q)
     if (p->data==q->data)  {p=p->next; q=q->next; }
     else{pre=pre->next; p=pre;   //A 链表新的开始比较结点。
          q=B; }                  //q 从 B 链表第一结点开始。
```

```
    if (q==null) return (1);      //B 是 A 的子序列。
     else   return (0);           //B 不是 A 的子序列。
    } //算法结束。
```

20. [题目分析] 本题也是模式匹配问题，应先找出链表 L2 在链表 L1 中的出现，然后将 L1 中的 L2 倒置过来。设 L2 在 L1 中出现时第一个字母结点的前驱的指针为 p，最后一个字母结点在 L1 中为 q 所指结点的前驱，则在保存 p 后继结点指针(s)的情况下，执行 p->next=q。之后将 s 到 q 结点的前驱依次插入到 p 结点之后，实现了 L2 在 L1 中的倒置。

```
  LinkedList  PatternInvert (LinkedList L1, L2)
//L1 和 L2 均是带头结点的单链表，数据结点的数据域均为一个字符。本算法将 L1 中与 L2 中数据域相同的连续结点的顺序完全倒置过来。
{p=L1;                  //p 是每趟匹配时 L1 中的起始结点前驱的指针。
 q=L1->next;            //q 是 L1 中的工作指针。
 s=L2->next;            //s 是 L2 中的工作指针。
 while (p!=null && s!=null)
    if (q->data==s->data) {q=q->next;s=s->next;} //对应字母相等，指针后移。
    else {p=p->next; q=p->next; s=L2->next; }      //失配时，L1 起始结点后移，L2 从首结点开始。
 if (s==null) //匹配成功，这时 p 为 L1 中与 L2 中首字母结点相同数据域结点的前驱，q
                为 L1 中与 L2 最后一个结点相同数据域结点的后继。
        {r=p->next;             //r 为 L1 的工作指针，初始指向匹配的首字母结点。
         p->next=q;             //将 p 与 q 结点的链接。
         while (r!=q);          //逐结点倒置。
            {s=r->next;         //暂存 r 的后继。
             r->next=p->next;   //将 r 所指结点倒置。
             p->next=r;
             r=s;               //恢复 r 为当前结点。
             }
        }
     else printf ("L2 并未在 L1 中出现");
     } //算法结束。
```

[算法讨论] 本算法只讨论了 L2 在 L1 至多出现一次(可能没出现)，没考虑在 L1 中多次出现的情况。若考虑多次出现，可在上面算法找到第一次出现后的 q 结点作 L1 中下次比较的第一字母结点，读者可自行完善之。

类似本题的另外叙述题的解答：

(1)[题目分析] 本题应先查找第 i 个结点，记下第 i 个结点的指针。然后从第 i+1 个结点起，直至第 m (1<i<m) 个结点止，依次插入到第 i-1 个结点之后。最后将暂存的第 i 个结点的指针指向第 m 结点，形成新的循环链表，结束了倒置算法。

```
  LinkedList PatternInvert1 (LinkedList L, int  i, m)
   //L 是有 m 个结点的链表的头结点的指针。表中从第 i (1<i<m) 个结点到第 m 个结点构成循环部分链表，本算法将这部分循环链表倒置。
     {if (i<1|| i>=m || m<4) {printf ("%d,%d 参数错误\n",i, m); exit (0); }
      p=L->next->next;          //p 是工作指针，初始指向第二结点 (已假定 i>1)。
      pre=L->next;              //pre 是前驱结点指针，最终指向第 i-1 个结点。
      j=1;                      //计数器
```

```
    while（j<i-1）            //查找第 i 个结点。
      {j++; pre=p; p=p->next;}//查找结束，p 指向第 i 个结点。
    q=p;                      //暂存第 i 个结点的指针。
    p=p->next;                //p 指向第 i+1 个结点，准备逆置。
    j+=2;                     //上面 while 循环结束时，j=i-1，现从第 i+1 结点开始逆置。
    while（j<=m）
       {r=p->next;            //暂存 p 的后继结点。
        p->next=pre->next;    //逆置 p 结点。
        pre->next=p;
        p=r;                  //p 恢复为当前待逆置结点。
        j++;                  //计数器增 1。
       }
    q->next=pre->next;  //将原第 i 个结点的后继指针指向原第 m 个结点。
```

[算法讨论] 算法中未深入讨论 i，m，j 的合法性，因题目的条件是 m>3 且 1<i<m。因此控制循环并未用指针判断（如一般情况下的 p!=null），结束循环也未用指针判断。注意最后一句 q->next=pre->next，实现了从原第 i 个结点到原第 m 个结点的循环。最后 pre->next 正是指向原第 m 个结点，不可用 p->next 代替 pre->next。

21.［题目分析］顺序存储结构的线性表的逆置，只需一个变量辅助空间。算法核心是选择循环控制变量的初值和终值。

```
void  SeqInvert（ElemType a[ ]，int n）
  //a 是具有 n 个元素用一维数组存储的线性表，本算法将其逆置。
  {for（i=0; i<=（n-1）/2; i++）
     {t=a[i]; a[i]= a[n-1-i]; a[n-1-i]=t; }
  }//算法结束
```

［算法讨论］算法中循环控制变量的初值和终值是关键。C 中数组从下标 0 开始，第 n 个元素的下标是 n-1。因为首尾对称交换，所以控制变量的终值是线性表长度的一半。当 n 为偶数，“一半”恰好是线性表长度的二分之一；若 n 是奇数，“一半”是小于 n/2 的最大整数，这时取大于 1/2 的最小整数的位置上的元素，恰是线性表中间位置的元素，不需要逆置。另外，由于 pascal 数组通常从下标 1 开始，所以，上下界处理上略有不同。这点请读者注意。

类似本题的其它题的解答：

这一组又选了 6 个题，都是单链表（包括单循环链表）的逆置。链表逆置的通常作法是：将工作指针指向第一个元素结点，将头结点的指针域置空。然后将链表各结点从第一结点开始直至最后一个结点，依次前插至头结点后，使最后插入的结点成为链表的第一结点，第一个插入的结点成为链表的最后结点。

（1）要求编程实现带头结点的单链表的逆置。首先建立一单链表，然后逆置。

```
typedef  struct  node
      {int  data; //假定结点数据域为整型。
       struct  node  *next;
      }node,*LinkedList;
LinkedList creat（ ）
 {LinkedList head, p
  int  x;
```

```
    head=（LinkedList）malloc（sizeof（node））;
    head->next=null;  /*设置头结点*/
    scanf（“%d”, &x）;
    while（x!=9999） /*约定输入 9999 时退出本函数*/
       {p=（LinkedList）malloc（sizeof（node））;
        p->data=x;
        p->next=head->next; /* 将新结点链入链表*/
        head->next=p;
        scanf（“%d”, &x）;
       }
     return（head）;
    }//结束 creat 函数。
  LinkedList invert1（LinkedList head）
   /*逆置单链表*/
   {LinkedList p=head->next; /*p 为工作指针*/
    head->next=null;
    while（p!=null）
      {r=p->next;          /*暂存 p 的后继*/
       p->next=head->next;
       head->next=p;
       p=r;
      }
     return（head）;
    }/*结束 invert1 函数*/
  main（）
   {LinkedList la;
    la=creat（）;     /*生成单链表*/
    la=invert1（la）; /*逆置单链表*/
    }
```

（2）本题要求将数据项递减有序的单链表重新排序，使数据项递增有序，要求算法复杂度为 O（n）。虽没说要求将链表逆置，这只是叙述不同，本质上是将单链表逆置，现编写如下：

```
  LinkedList invert2（LinkedList la）
//la 是带头结点且数据项递减有序的单链表，本算法将其排列成数据项递增有序的单链表。
 {p=la->next;      /*p 为工作指针*/
  la->next=null;
  while（p!=null）
    {r=p->next;    /*暂存 p 的后继。*/
     p->next=la->next; /*将 p 结点前插入头结点后。*/
     la->next=p; p=r;
    }
     }//结束算法
```

（3）本题要求倒排循环链表，与上面倒排单链表处理不同之处有二：一是初始化成循环链表而不是空链表;二是判断链表尾不用空指针而用是否是链表头指针。算法中语句片段如下：

```
p=la->next;          //p 为工作指针。
la->next=la;         //初始化成空循环链表。
while（p!=la）      //当 p=la 时循环结束。
  {r=p->next;        //暂存 p 的后继结点
   p->next=la->next; //逆置
   la->next=p;  p=r;
  }
```

（4）不带头结点的单链表逆置比较复杂，解决方法可以给加上头结点：

```
la=（LinkedList）malloc（sizeof（node））;
la->next=L;
```

之后进行如上面（2）那样的逆置，最后再删去头结点：

```
L=la->next;    //L 是不带头结点的链表的指针。
free（la）;     //释放头结点。
```

若不增加头结点，可用如下语句片段：

```
p=L->next;     //p 为工作指针。
L->next=null; //第一结点成为尾结点。
while（p!=null）
  {r=p->next;
   p->next=L; //将 p 结点插到 L 结点前面。
   L=p;        //L 指向新的链表“第一”元素结点。
   p=r;
  }
```

（5）同（4），只是叙述有异。

（6）同（2），差别仅在于叙述不同。

22.［题目分析］在无序的单链表上，查找最小值结点，要查遍整个链表，初始假定第一结点是最小值结点。当找到最小值结点后，判断数据域的值是否是奇数，若是，则“与其后继结点的值相交换”即仅仅交换数据域的值，用三个赋值语句即可交换。若与后继结点交换位置，则需交换指针，这时应知道最小值结点的前驱。至于删除后继结点，则通过修改最小值结点的指针域即可。

［算法设计］

```
void  MiniValue（LinkedList la）
//la 是数据域为正整数且无序的单链表，本算法查找最小值结点且打印。若最小值结点的数值是奇数，则与后继结点值交换；否则，就删除其直接后继结点。
{p=la->next;     //设 la 是头结点的头指针，p 为工作指针。
 pre=p;          //pre 指向最小值结点，初始假定首元结点值最小。
 while（p->next!=null）  //p->next 是待比较的当前结点。
   {if（p->next->data<pre->data）pre=p->next;
    p=p->next;  //后移指针
   }
 printf（“最小值=%d\n”, pre->data）;
 if（pre->data%2!=0）    //处理奇数
   if（pre->next!=null）//若该结点没有后继，则不必交换
    {t= pre->data; pre->data=pre->next->data; pre->next->data=t; }//交换完毕
```

```
else//处理偶数情况
  if（pre->next!=null）//若最小值结点是最后一个结点，则无后继
    {u=pre->next; pre->next=u->next; free（u）;} //释放后继结点空间
```

23.[题目分析] 将一个结点数据域为字符的单链表，分解成含有字母字符、数字字符和其它字符的三个循环链表，首先要构造分别含有这三类字符的表头结点。然后从原链表第一个结点开始，根据结点数据域是字母字符、数字字符和其它字符而分别插入到三个链表之一的链表。注意不要因结点插入新建链表而使原链表断链。另外，题目并未要求链表有序，插入采用“前插法”，每次插入的结点均成为所插入链表的第一元素的结点即可。

```
void  OneToThree（LinkedList L，la，ld，lo）
//L 是无头结点的单链表第一个结点的指针，链表中的数据域存放字符。本算法将链表 L 分解成含有英文字母字符、数字字符和其它字符的带头结点的三个循环链表。
{la=（LinkedList）malloc（sizeof（LNode））;  //建立三个链表的头结点
 ld=（LinkedList）malloc（sizeof（LNode））;
 lo=（LinkedList）malloc（sizeof（LNode））;
la->next=la; ld->next=ld; lo->next=lo; //置三个循环链表为空表
while（L!=null）//分解原链表。
   {r=L; L=L->next;  //L 指向待处理结点的后继
    if（r->data>=‘a’ && r->data<=‘z’ || r->data>=‘A’ && r->data<=‘Z’）
      {r->next=la->next;   la->next=r;}          //处理字母字符。
     else if（r->data>=‘0’ && r->data<=‘9’）
        {r->next=ld->next; ld->next=r;}          //处理数字字符
      else {r->next=lo->next; lo->next=r;}       //处理其它符号。
   }//结束 while（L!=null）。
}//算法结束
```

[算法讨论] 算法中对 L 链表中每个结点只处理一次，时间复杂度 O（n），只增加了必须的三个表头结点，符合题目“用最少的时间和最少的空间”的要求。

24.[题目分析] 在递增有序的线性表中，删除数值相同的元素，要知道被删除元素结点的前驱结点。

```
LinkedList DelSame（LinkedList la）
//la 是递增有序的单链表，本算法去掉数值相同的元素，使表中不再有重复的元素。
{pre=la->next; //pre 是 p 所指向的前驱结点的指针。
 p=pre->next; //p 是工作指针。设链表中至少有一个结点。
 while（p!=null）
   if（p->data==pre->data）          //处理相同元素值的结点
     {u=p; p=p->next; free（u）;} //释放相同元素值的结点
   else {pre->next=p; pre=p; p=p->next;} //处理前驱，后继元素值不同
pre->next=p; //置链表尾。
}//DelSame
```

[算法讨论] 算法中假设链表至少有一个结点，即初始时 pre 不为空，否则 p->next 无意义。算法中最后 pre->next=p 是必须的，因为可能链表最后有数据域值相同的结点，这些结点均被删除，指针后移使 p=null 而退出 while 循环，所以应有 pre->next=p 使链表有尾。若链表尾部没数据域相同的结点，pre 和 p 为前驱和后继，pre->next=p 也是对的。

顺便提及，题目应叙述为非递减有序，因为“递增”是说明各结点数据域不同，一个值比一个值大，不会存在相同值元素。

25.［题目分析］建立递增有序的顺序表，对每个输入数据，应首先查找该数据在顺序表中的位置，若表中没有该元素则插入之，如已有该元素，则不再插入，为此采用折半查找方法。

```
FUNC  BinSearch（VAR a：sqlisttp；x：integer）：integer；
//在顺序表 a 中查找值为 x 的元素，如查找成功，返回 0 值，如 x 不在 a 中，则返回查找失败时的较大下标值。
    low:=1；high:=a.last；found:=false；
    WHILE（low<=high）AND NOT found DO
        [mid:=（low+high）DIV 2；
         IF a.elem[mid]=x THEN found:=true
         ELSE IF a.elem[mid]>x THEN high:=mid-1 ELSE low:=mid+1；
        ]
    IF found=true THEN return（0）
    ELSE return（low）；//当查找失败时，low=high+1。
ENDF；//结束对分查找函数。
PROC create（VAR L：sqlisttp）
   //本过程生成顺序表 L。
   L.last:=0；        //顺序表 L 初始化。
   read（x）；
   WHILE x<>9999 DO           //设 x=9999 时退出输入
      [k:=binsearch（L，x）；//去查找 x 元素。
       IF k<>0                //不同元素才插入
         THEN [FOR i:=L.last DOWNTO k DO L.elem[i+1]:=L.elem[i]；
               L.elem[k]=x；L.last:= L.last+1；//插入元素 x，线性表长度增 1
              ]
       read（x）；
       ]
ENDP；//结束过程 creat
```

26.［题目分析］在由正整数序列组成的有序单链表中，数据递增有序，允许相等整数存在。确定比正整数 x 大的数有几个属于计数问题，相同数只计一次，要求记住前驱，前驱和后继值不同时移动前驱指针，进行计数。将比正整数 x 小的数按递减排序，属于单链表的逆置问题。比正整数 x 大的偶数从表中删除，属于单链表中结点的删除，必须记住其前驱，以使链表不断链。算法结束时，链表中结点的排列是：小于 x 的数按递减排列，接着是 x（若有的话），最后是大于 x 的奇数。

```
    void  exam（LinkedList la，int x）
    //la 是递增有序单链表，数据域为正整数。本算法确定比 x 大的数有几个；将比 x 小的数按递减排序，并将比 x 大的偶数从链表中删除。）
    {p=la->next；q=p;//p 为工作指针  q 指向最小值元素，其可能的后继将是>=x 的第一个元素。
     pre=la；          //pre 为 p 的前驱结点指针。
     k=0；             //计数（比 x 大的数）。
     la->next=null；//置空单链表表头结点。
     while（p && p->data<x）       //先解决比 x 小的数按递减次序排列
      {r=p->next；          //暂存后继
```

```
    p->next=la->next; //逆置
    la->next=p;
    p=r; //恢复当前指针。退出循环时，r 指向值>=x 的结点。
   }
  q->next=p; pre=q;    //pre 指向结点的前驱结点
     while (p->data==x) {pre=p; p=p->next;} //从小于 x 到大于 x 可能经过等于 x
       while (p)              //以下结点的数据域的值均大于 x
         {k++; x=p->data;     //下面仍用 x 表示数据域的值，计数
          if (x % 2==0)       //删偶数
             {while (p->data==x)
             {u=p; p=p->next; free(u); }
              pre->next=p;  //拉上链
              }
           else                //处理奇数
             while (p->data==x)//相同数只记一次
             {pre->next=p; pre=p; p=p->next; }
          }//while(p)
        printf ("比值%d 大的数有%d 个\n", x, k);
    }//算法 exam 结束
```

[算法讨论] 本题“要求用最少的时间和最小的空间”。本算法中“最少的时间”体现在链表指针不回溯，最小空间是利用了几个变量。在查比 x 大的数时，必须找到第一个比 x 大的数所在结点（因等于 x 的数可能有，也可能多个，也可能没有）。之后，计数据的第一次出现，同时删去偶数。

顺便指出，题目设有“按递增次序”的“有序单链表”，所给例子序列与题目的论述并不一致。

27. [题目分析] 单链表中查找任何结点，都必须从头指针开始。本题要求将指针 p 所指结点与其后继结点交换，这不仅要求知道 p 结点，还应知道 p 的前驱结点。这样才能在 p 与其后继结点交换后，由原 p 结点的前驱来指向原 p 结点的后继结点。

另外，若无特别说明，为了处理的方便统一，单链表均设头结点，链表的指针就是头结点的指针。并且由于链表指针具有标记链表的作用，也常用指针名冠以链表名称。如“链表 head”既指的是链表的名字是 head，也指出链表的头指针是 head。

```
LinkedList Exchange (LinkedList HEAD, p)
  //HEAD 是单链表头结点的指针，p 是链表中的一个结点。本算法将 p 所指结点与其后
          继结点交换。
  {q=head->next;  //q 是工作指针，指向链表中当前待处理结点。
   pre=head;      //pre 是前驱结点指针，指向 q 的前驱。
   while (q!=null && q!=p) {pre=q; q=q->next; }   //未找到 p 结点，后移指针。
   if (p->next==null) printf ("p 无后继结点\n") ;  //p 是链表中最后一个结点，
无后继。
   else//处理 p 和后继结点交换
     {q=p->next;       //暂存 p 的后继。
      pre->next=q;     //p 前驱结点的后继指向 p 的后继。
      p->next=q->next; //p 的后继指向原 p 后继的后继。
      q->next=p      ; //原 p 后继的后继指针指向 p。
```

```
        }
    }//算法结束。
```

类似本题的其它题目的解答：

（1）与上面第 27 题基本相同，只是明确说明“p 指向的不是链表最后那个结点。”

（2）与上面第 27 题基本相同，仅叙述不同，故不再作解答。

28.［题目分析］本题链表结点的数据域存放英文单词，可用字符数组表示，单词重复出现时，链表中只保留一个，单词是否相等的判断使用 strcmp 函数，结点中增设计数域，统计单词重复出现的次数。

```
typedef  struct  node
  {int  freg; //频域，记单词出现的次数。
   char  word[maxsize]; //maxsize 是单词中可能含有的最多字母个数。
   struct  node  *next;
  }node, *LinkedList;
(1)LinkedList creat ()
  //建立有 n（n>0）个单词的单向链表，若单词重复出现，则只在链表中保留一个。
  {LinkedList la;
   la=（LinkedList）malloc（sizeof（node)); //申请头结点。
   la->next=null;          //链表初始化。
   for（i=1; i<=n; i++）  //建立 n 个结点的链表
     {scanf（“%s”, a);    //a 是与链表中结点数据域同等长度的字符数组。
      p=la->next; pre=p; //p 是工作指针,pre 是前驱指针。
      while（p!=null）
        if（strcmp(p->data, a）==0) {p->freg++; break; } //单词重复出现,频度增 1。
        else {pre=p; p=p->next; }      //指针后移。
      if（p==null）        //该单词没出现过，应插入。
        {p=（LinkedList）malloc（sizeof（node));
         strcopy（p->data, a); p->freg=1; p->next=null; pre->next=p;
        }      //将新结点插入到链表最后。
  }//结束 for 循环。
  return（la);
 }//结束 creat 算法。
  (2) void CreatOut（ ）
     //建立有 n 个单词的单向链表，重复单词只在链表中保留一个,最后输出频度最高的 k 个单词。
  {LinkedList la;
   la=（LinkedList）malloc（sizeof（node)); //申请头结点。
   la->next=null;          //链表初始化。
   for（i=1; i<=n; i++）  //建立 n 个结点的链表
     {scanf（“%s”, a);    //a 是与链表中结点数据域同等长度的字符数组。
      p=la->next; pre=p; //p 是工作指针,pre 是前驱指针。
      while（p!=null）
        if（strcmp(p->data, a)==0)
          {p->freg++;          //单词重复出现,频度增 1。
```

```
            pre->next=p->next; //先将 p 结点从链表上摘下，再按频度域值插入到合适位置
            pre=la; q=la->next;
            while(q->freg>p->freg) (pre=q; q=q->next; )
            pre->next=p; p->next=q; //将 p 结点插入到合适位置
           }
         else {pre=p; p=p->next; }      //指针后移。
       if (p==null)        //该单词没出现过，应插入到链表最后。
         {p= (LinkedList) malloc (sizeof (node));
          strcopy (p->data, a); p->freg=1; p->next=null; pre->next=p;
         }//if 新结点插入。
     }//结束 for 循环建表。
    int k,i=0;
    scanf("输入要输出单词的个数%d",&k);
    p=la->next;
    while (p && i<k) //输出频度最高的 k 个单词
      {printf ("第%3d 个单词%s 出现%3d 次\n", ++i, p->data, p->freg);
       p=p->next;
      }
    if (!p)
       printf("给出的%d 值太大\n",k);
    }//结束算法
```

29.[题目分析] 双向循环链表自第二结点至表尾递增有序，要求将第一结点插入到链表中，使整个链表递增有序。由于已给条件（$a_1<x<a_n$），故应先将第一结点从链表上摘下来，再将其插入到链表中相应位置。由于是双向链表，不必象单链表那样必须知道插入结点的前驱。

```
    void DInsert (DLinkedList dl)
  //dl 是无头结点的双向循环链表，自第二结点起递增有序。本算法将第一结点(a1<x<an)插入到链表中，使整个链表递增有序。
  {s=la;          //s 暂存第一结点的指针。
   p=la->next; p->prior=la->prior; p->prior->next=p; //将第一结点从链表上摘下。
   while (p->data<x) p=p->next;    //查插入位置
   s->next=p; s->prior=p->prior; p->prior->next=s; p->prior=s; //插入原第一结点 s
  }//算法结束。
```

[算法讨论] 由于题目已给 $a_1<x<a_n$，所以在查找第一结点插入位置时用的循环条件是 p->data<x，即在 a_1 和 a_n 间肯定能找到第一结点的插入位置。若无此条件，应先看第一结点数据域值 x 是否小于等于 a_1，如是，则不作任何操作。否则，查找其插入位置，循环控制要至多查找完 a_1 到 a_n 结点。

```
    if (p->data<x) p=p->next; else  break;
```

30.[题目分析] 在顺序存储的线性表上删除元素，通常要涉及到一系列元素的移动(删第 i 个元素，第 i+1 至第 n 个元素要依次前移)。本题要求删除线性表中所有值为 item 的数据元素，并未要求元素间的相对位置不变。因此可以考虑设头尾两个指针（i=1, j=n），从两端向中间移动，凡遇到值 item 的数据元素时，直接将右端元素左移至值为 item 的数据元

素位置。

```
void  Delete (ElemType A[ ], int  n)
//A 是有 n 个元素的一维数组，本算法删除 A 中所有值为 item 的元素。
{i=1; j=n; //设置数组低、高端指针（下标）。
 while (i<j)
   {while (i<j && A[i]!=item) i++;          //若值不为 item，左移指针。
    if (i<j) while (i<j && A[j]==item) j--; //若右端元素值为 item，指针左移
    if (i<j) A[i++]=A[j--];
   }
```

[算法讨论] 因元素只扫描一趟，算法时间复杂度为 O（n）。删除元素未使用其它辅助空间，最后线性表中的元素个数是 j。若题目要求元素间相对顺序不变，请参见本章三、填空题 25 的算法。

31. [题目分析] 本题所用数据结构是静态双向链表，其结构定义为：

```
typedef  struct  node
   {char  data[maxsize]; //用户姓名，maxsize 是可能达到的用户名的最大长度。
    int  Llink，Rlink; //前向、后向链，其值为乘客数组下标值。
   }unode;
unode  user[max]; //max 是可能达到的最多客户数。
```

设 av 是可用数组空间的最小下标，当有客户要订票时，将其姓名写入该单元的 data 域，然后在静态链表中查找其插入位置。将该乘客姓名与链表中第一个乘客姓名比较，根据大于或小于第一个乘客姓名，而决定沿第一个乘客的右链或左链去继续查找，直到找到合适位置插入之。

```
 void Insert (unode  user[max],int av)
//user 是静态双向链表，表示飞机票订票系统，元素包含 data、Llink 和 Rlink 三个域，结点按来客姓名排序。本算法处理任一乘客订票申请。
{scanf ("%s", s);            //s 是字符数组，存放乘客姓名。
 strcopy (user[av].data, s);
 p=1;              //p 为工作指针（下标）
 if(strcmp (user[p].data, s) <0) //沿右链查找
   {while (p!=0 && strcmp (user[p].data, s) <0) {pre=p; p=user[p].Rlink; }
    user[av].Rlink=p; user[av].Llink=pre;   //将新乘客链入表中
    user[pre].Rlink=av; user[p].Llink=av;
   }
 else  //沿左右链查找
   {while (p!=0 && strcmp (user[p].data, s) >0) {pre=p; p=user[p].Llink; }
    user[av].Rlink=pre; user[av].Llink=p;   //将新乘客链入表中
    user[pre].Llink=av; user[p].Rlink=av;
   }
 }//算法结束
```

[算法讨论] 本算法只讨论了乘客订票情况，未考虑乘客退票。也未考虑从空开始建立链表。增加乘客时也未考虑姓名相同者（实际系统姓名不能做主关键字）。完整系统应有（1）初始化，把整个数组空间初始化成双向静态链表，全部空间均是可利用空间。（2）申请空间。当有乘客购票时，要申请空间，直到无空间可用为止。（3）释放空间。当乘客退票时，将其空间收回。由于空间使用无优先级，故可将退票释放的空间作为下个可利用空间，链入可利

用空间表中。

32.[题目分析]首先在双向链表中查找数据值为 x 的结点，查到后，将结点从链表上摘下，然后再顺结点的前驱链查找该结点的位置。

```
DLinkList  locate(DLinkList L, ElemType x)
// L 是带头结点的按访问频度递减的双向链表，本算法先查找数据 x,查找成功时结点的访问频度域增 1，最后将该结点按频度递减插入链表中适当位置。
{DLinkList p=L->next,q;     //p 为 L 表的工作指针，q 为 p 的前驱，用于查找插入位置。
 while (p && p->data !=x) p=p->next; // 查找值为 x 的结点。
 if (!p) {printf("不存在值为 x 的结点\n"); exit(0);}
 else { p->freq++;                  // 令元素值为 x 的结点的 freq 域加 1 。
        p->next->pred=p->pred;    // 将 p 结点从链表上摘下。
        p->pred->next=p->next;
        q=p->pred;                // 以下查找 p 结点的插入位置
        while (q !=L && q->freq<p->freq) q=q->pred;
        p->next=q->next; q->next->pred=p;// 将 p 结点插入
        p->pred=q; q->next=p;
      }
 return(p);    //返回值为 x 的结点的指针
}  // 算法结束
```

33.[题目分析] 题目要求按递增次序输出单链表中各结点的数据元素，并释放结点所占存储空间。应对链表进行遍历，在每趟遍历中查找出整个链表的最小值元素，输出并释放结点所占空间；再查次最小值元素，输出并释放空间，如此下去，直至链表为空，最后释放头结点所占存储空间。当然，删除结点一定要记住该结点的前驱结点的指针。

```
   void  MiniDelete (LinkedList  head)
//head 是带头结点的单链表的头指针，本算法按递增顺序输出单链表中各结点的数据元素，并释放结点所占的存储空间。
   {while (head->next!=null)  //循环到仅剩头结点。
      {pre=head;              //pre 为元素最小值结点的前驱结点的指针。
       p=pre->next;           //p 为工作指针
       while (p->next!=null)
         {if (p->next->data<pre->next->data) pre=p; //记住当前最小值结点的前驱
          p=p->next;
         }
       printf (pre->next->data); //输出元素最小值结点的数据。
       u=pre->next; pre->next=u->next; free (u); //删除元素值最小的结点，释放结点空间
      }// while (head->next!=null)
     free (head); }    //释放头结点。
```

[算法讨论] 算法中使用的指针变量只有 pre，p 和 u 三个，请读者细心体会。要注意没特别记最小值结点，而是记其前驱。

34.[题目分析] 留下三个链表中公共数据，首先查找两表 A 和 B 中公共数据，再去 C 中找有无该数据。要消除重复元素，应记住前驱，要求时间复杂度 O（m+n+p），在查找每个链表时，指针不能回溯。

```
LinkedList Common (LinkedList A, B, C)
```

//A，B 和 C 是三个带头结点且结点元素值非递减排列的有序表。本算法使 A 表仅留下三个表均包含的结点，且结点值不重复，释放所有结点。

```
{pa=A->next; pb=B->next; pc=C->next; //pa,pb 和 pc 分别是 A,B 和 C 三个表的工作指针。
 pre=A; //pre 是 A 表中当前结点的前驱结点的指针。
 while (pa && pb && pc) //当 A,B 和 C 表均不空时，查找三表共同元素
   { while (pa && pb)
    if (pa->data<pb->data) {u=pa; pa=pa->next; free (u); }//结点元素值小时，后移指针。
     else if (pa->data> pb->data) pb=pb->next;
      else if (pa && pb) //处理 A 和 B 表元素值相等的结点
           {while (pc && pc->data<pa->data) pc=pc->next;
            if(pc)
            {if (pc->data>pa->data) //pc 当前结点值与 pa 当前结点值不等,pa 后移指针。
              {u=pa; pa=pa->next; free (u); }
             else  //pc，pa 和 pb 对应结点元素值相等。
              {if(pre==A){ pre->next=pa; pre=pa; pa=pa->next} //结果表中第一个结点。
               else if(pre->data==pa->data)          //（处理）重复结点不链入 A 表
                       {u=pa; pa=pa->next; free (u); }
               else {pre->next=pa; pre=pa; pa=pa->next; }//将新结点链入 A 表。
               pb=pb->next; pc=pc->next; //链表的工作指针后移。
            } }//else pc，pa 和 pb 对应结点元素值相等
       if (pa==null) pre->next=null; //原 A 表已到尾,置新 A 表表尾
       else       //处理原 A 表未到尾而 B 或 C 到尾的情况
         {pre->next=null;       //置 A 表表尾标记
          while (pa!=null)     //删除原 A 表剩余元素。
            {u=pa; pa=pa->next; free (u); }
```

[算法讨论] 算法中 A 表、B 表和 C 表均从头到尾（严格说 B、C 中最多一个到尾）遍历一遍，算法时间复杂度符合 O（m+n+p）。算法主要由 while（pa && pb && pc）控制。三表有一个到尾则结束循环。算法中查到 A 表与 B 表和 C 表的公共元素后，又分三种情况处理：一是三表中第一个公共元素值相等的结点；第二种情况是，尽管不是第一结点，但与前驱结点元素值相同，不能成为结果表中的结点；第三种情况是新结点与前驱结点元素值不同，应链入结果表中，前驱指针也移至当前结点，以便与以后元素值相同的公共结点进行比较。算法最后要给新 A 表置结尾标记，同时若原 A 表没到尾，还应释放剩余结点所占的存储空间。

第 3 章　　栈和队列

一　选择题

1. 对于栈操作数据的原则是（　　）。【青岛大学 2001 五、2（2 分）】

A. 先进先出　　B. 后进先出　　C. 后进后出　　D. 不分顺序

2. 在作进栈运算时,应先判别栈是否（ ① ）,在作退栈运算时应先判别栈是否（ ② ）。当栈中元素为 n 个,作进栈运算时发生上溢,则说明该栈的最大容量为（ ③ ）。

为了增加内存空间的利用率和减少溢出的可能性,由两个栈共享一片连续的内存空间时,应将两栈的（ ④ ）分别设在这片内存空间的两端,这样,当（ ⑤ ）时，才产生上溢。

①, ②: A. 空　　B. 满　　C. 上溢　　D. 下溢

③: A. n-1　　B. n　　C. n+1　　D. n/2

④: A. 长度　　B. 深度　　C. 栈顶　　D. 栈底

⑤: A. 两个栈的栈顶同时到达栈空间的中心点.

B. 其中一个栈的栈顶到达栈空间的中心点.

C. 两个栈的栈顶在栈空间的某一位置相遇.

D. 两个栈均不空,且一个栈的栈顶到达另一个栈的栈底.

【上海海运学院 1997 二、1（5 分）】【上海海运学院 1999 二、1（5 分）】

3. 一个栈的输入序列为 123…n，若输出序列的第一个元素是 n，输出第 i（1<=i<=n）个元素是（　　）。

A. 不确定　　B. n-i+1　　C. i　　D. n-i

【中山大学 1999 一、9(1 分)】

4. 若一个栈的输入序列为 1,2,3,…,n，输出序列的第一个元素是 i，则第 j 个输出元素是（　　）。

A. i-j-1　　B. i-j　　C. j-i+1　　D. 不确定的

【武汉大学 2000 二、3】

5. 若已知一个栈的入栈序列是 1,2,3,…,n，其输出序列为 p_1, p_2, p_3，…，p_N,若 p_N 是 n，则 p_i 是（　　）。

A. i　　B. n-i　　C. n-i+1　　D. 不确定

【南京理工大学 2001 一、1（1.5 分）】

6. 有六个元素 6，5，4，3，2，1 的顺序进栈，问下列哪一个不是合法的出栈序列？（　　）

A. 5 4 3 6 1 2　　B. 4 5 3 1 2 6　　C. 3 4 6 5 2 1　　D. 2 3 4 1 5 6

【北方交通大学 2001 一、3（2 分）】

7. 设栈的输入序列是 1，2，3，4,则（　）不可能是其出栈序列。【中科院计算所 2000 一、10(2 分)】

A. 1，2，4，3,　　B. 2，1，3，4,　　C. 1，4，3，2,

D. 4，3，1，2,　　E. 3，2，1，4,

8. 一个栈的输入序列为 1 2 3 4 5，则下列序列中不可能是栈的输出序列的是（　　）。

A. 2 3 4 1 5　　B. 5 4 1 3 2　　C. 2 3 1 4 5　　D. 1 5 4 3 2

【南开大学 2000 一、1】【山东大学 2001 二、4 (1 分)】【北京理工大学 2000 一、2（2 分）】

9. 设一个栈的输入序列是 1，2，3，4，5,则下列序列中，是栈的合法输出序列的是（　　）。

A. 5 1 2 3 4　　B. 4 5 1 3 2　　C. 4 3 1 2 5　　D. 3 2 1 5 4

【合肥工业大学 2001 一、1（2 分）】

10. 某堆栈的输入序列为 a, b，c ，d,下面的四个序列中，不可能是它的输出序列的是

（　　）。

A. a，c，b，d　　B. b，c，d，a　　C. c，d，b，a　　D. d，c，a，b

【北京航空航天大学 2000 一、3（2 分）】【北京邮电大学 1999 一、3（2 分）】

11. 设 abcdef 以所给的次序进栈，若在进栈操作时，允许退栈操作,则下面得不到的序列为（　　）。

A. fedcba　　B. bcafed　　C. dcefba　　D. cabdef

【南京理工大学 1996 一、9（2 分）】

12. 设有三个元素 X，Y，Z 顺序进栈（进的过程中允许出栈），下列得不到的出栈排列是（　　）。

A. XYZ　　B. YZX　　C. ZXY　　D. ZYX

【南京理工大学 1997 一、5（2 分）】

13. 输入序列为 ABC，可以变为 CBA 时，经过的栈操作为（　　）【中山大学 1999 一、8(1 分)】

A. push,pop,push,pop,push,pop　　B. push,push,push,pop,pop,pop

C. push,push,pop,pop,push,pop　　D. push,pop,push,push,pop,pop

14. 若一个栈以向量 V[1..n]存储，初始栈顶指针 top 为 n+1，则下面 x 进栈的正确操作是（　　）。

A. top:=top+1; V [top]:=x　　B. V [top]:=x; top:=top+1

C. top:=top-1; V [top]:=x　　D. V [top]:=x; top:=top-1

【南京理工大学 1998 一、13（2 分）】

15. 若栈采用顺序存储方式存储，现两栈共享空间 V[1..m]，top[i]代表第 i 个栈(i =1,2)栈顶，栈 1 的底在 v[1]，栈 2 的底在 V[m]，则栈满的条件是（　　）。

A. |top[2]-top[1]|=0　　B. top[1]+1=top[2]　　C. top[1]+top[2]=m　　D. top[1]=top[2]

【南京理工大学 1999 一、14（1 分）】

16. 栈在（　　）中应用。【中山大学 1998 二、3（2 分）】

A. 递归调用　　B. 子程序调用　　C. 表达式求值　　D. A，B，C

17. 一个递归算法必须包括（　　）。【武汉大学 2000 二、2】

A. 递归部分　　B. 终止条件和递归部分　　C. 迭代部分　　D.终止条件和迭代部分

18. 执行完下列语句段后，i 值为：（　　）【浙江大学 2000 一 、6 (3 分)】

```
int   f(int x)
{ return  ((x>0) ? x* f(x-1):2);}
 int i  ;
 i =f(f(1));
```

A. 2　　B. 4　　C. 8　　D. 无限递归

19. 表达式 a*(b+c)-d 的后缀表达式是(　　)。【南京理工大学 2001 一、2（1.5 分）】

A. abcd*+-　　B. abc+*d-　　C. abc*+d-　　D. -+*abcd

20. 表达式 3* 2^(4+2*2-6*3)-5 求值过程中当扫描到 6 时，对象栈和算符栈为（　　），其中^为乘幂 。

A. 3,2,4,1,1；(*^(+*-　　B. 3,2,8；(*^-　　C. 3,2,4,2,2；(*^(-　　D. 3,2,8；(*^(-

【青岛大学 2000 五、5（2 分）】

21. 设计一个判别表达式中左，右括号是否配对出现的算法，采用（　　）数据结构最佳。

A．线性表的顺序存储结构　　B．队列　　C．线性表的链式存储结构　　D．栈

【西安电子科技大学 1996 一、6（2 分）】

22．用链接方式存储的队列，在进行删除运算时（　　）。【北方交通大学 2001 一、12（2 分）】

A．仅修改头指针　B．仅修改尾指针　C．头、尾指针都要修改　D．头、尾指针可能都要修改

23．用不带头结点的单链表存储队列时，其队头指针指向队头结点，其队尾指针指向队尾结点，则在进行删除操作时（　　）。【北京理工大学 2001 六、3（2 分）】

A．仅修改队头指针　　B．仅修改队尾指针

C．队头、队尾指针都要修改　D．队头，队尾指针都可能要修改

24．递归过程或函数调用时，处理参数及返回地址，要用一种称为（　　）的数据结构。

A．队列　　B．多维数组　　C．栈　　D．线性表

【福州大学 1998 一、1（2 分）】

25．假设以数组 A[m]存放循环队列的元素，其头尾指针分别为 front 和 rear，则当前队列中的元素个数为（　　）。【北京工商大学 2001 一、2（3 分）】

A．(rear-front+m)%m　　B．rear-front+1　　C．(front-rear+m)%m　　D．(rear-front)%m

26．循环队列 A[0..m-1]存放其元素值，用 front 和 rear 分别表示队头和队尾，则当前队列中的元素数是（　　）。【南京理工大学 2001 一、5（1.5 分）】

A．(rear-front+m)%m　　B．rear-front+1　　C．rear-front-1　　D．rear-front

27．循环队列存储在数组 A[0..m]中，则入队时的操作为（　　）。【中山大学 1999 一、6（1 分）】

A．rear=rear+1　　B．rear=(rear+1) mod (m-1)

C．rear=(rear+1) mod m　　D．rear=(rear+1)mod(m+1)

28．若用一个大小为 6 的数组来实现循环队列，且当前 rear 和 front 的值分别为 0 和 3，当从队列中删除一个元素，再加入两个元素后，rear 和 front 的值分别为多少？（ ）【浙江大学 1999 四、1(4 分)】

A．1 和 5　　B．2 和 4　　C．4 和 2　　D．5 和 1

29．已知输入序列为 abcd 经过输出受限的双向队列后能得到的输出序列有（　　）。

A．dacb　　B．cadb　　C．dbca　　D．bdac　　E．以上答案都不对

【西安交通大学 1996 三、3 (3 分)】

30．若以 1234 作为双端队列的输入序列，则既不能由输入受限的双端队列得到，也不能由输出受限的双端队列得到的输出序列是（　）。【西安电子科技大学 1996 一、5（2 分）】

A．1234　　B．4132　　C．4231　　D．4213

31．最大容量为 n 的循环队列，队尾指针是 rear，队头是 front，则队空的条件是　（　　）。

A．(rear+1) MOD n=front　　B．rear=front

C．rear+1=front　　D．(rear-1) MOD n=front

【南京理工大学 1999　一、16（2 分）】

32．栈和队列的共同点是（　　）。【燕山大学 2001 一、1（2 分）】

A．都是先进先出　　B．都是先进后出

C．只允许在端点处插入和删除元素　　D．没有共同点

33．栈的特点是（　①　），队列的特点是（　②　），栈和队列都是（　③　）。若进栈序

列为 1,2,3,4 则（ ④ ）不可能是一个出栈序列（不一定全部进栈后再出栈）；若进队列的序列为 1,2,3,4 则（ ⑤ ）是一个出队列序列。【北方交通大学 1999 一、1（5 分）】

①, ②: A. 先进先出 B. 后进先出 C. 进优于出 D. 出优于进

③: A.顺序存储的线性结构 B.链式存储的线性结构

C.限制存取点的线性结构 D.限制存取点的非线性结构

④, ⑤: A. 3,2,1,4 B. 3,2,4,1 C. 4,2,3,1 D. 4,3,2,1 F. 1,2,3,4 G. 1,3,2,4

34. 栈和队都是（ ）【南京理工大学 1997 一、3（2 分）】

A. 顺序存储的线性结构 B. 链式存储的非线性结构

C. 限制存取点的线性结构 D. 限制存取点的非线性结构

35. 设栈 S 和队列 Q 的初始状态为空，元素 e1，e2，e3，e4,e5 和 e6 依次通过栈 S，一个元素出栈后即进队列 Q，若 6 个元素出队的序列是 e2，e4，e3,e6,e5,e1 则栈 S 的容量至少应该是（ ）。

A. 6 B. 4 C. 3 D. 2

【南京理工大学 2000 一、6（1.5 分）】

36. 用单链表表示的链式队列的队头在链表的（ ）位置。【清华大学 1998 一、1（2 分）】

A. 链头 B. 链尾 C. 链中

37. 依次读入数据元素序列{a，b，c，d，e，f，g}进栈,每进一个元素，机器可要求下一个元素进栈或弹栈，如此进行，则栈空时弹出的元素构成的序列是以下哪些序列？【哈尔滨工业大学 2000 七（8 分）】

A. {d ，e，c，f，b，g，a} B. {f，e，g，d，a，c，b}

C. {e，f，d，g，b，c，a} D. {c，d，b，e，f，a，g}

二 判断题

1. 消除递归不一定需要使用栈，此说法（ ）

【中科院计算所 1998 二、2（2 分）】【中国科技大学 1998 二、2（2 分）】

2. 栈是实现过程和函数等子程序所必需的结构。（ ）【合肥工业大学 2000 二、2（1 分）】

3. 两个栈共用静态存储空间，对头使用也存在空间溢出问题。（ ）【青岛大学 2000 四、2（1 分）】

4. 两个栈共享一片连续内存空间时，为提高内存利用率，减少溢出机会，应把两个栈的栈底分别设在这片内存空间的两端。（ ）【上海海运学院 1998 一、4（1 分）】

5. 即使对不含相同元素的同一输入序列进行两组不同的合法的入栈和出栈组合操作，所得的输出序列也一定相同。（ ）【北京邮电大学 1999 二、4（2 分）】

6. 有 n 个数顺序（依次）进栈，出栈序列有 Cn 种，Cn=[1/（n+1）]*（2n）!/[(n!)*(n!)]。（ ）

【北京邮电大学 1998 一、3（2 分）】

7. 栈与队列是一种特殊操作的线性表。（ ）【青岛大学 2001 四、3 （1 分）】

8. 若输入序列为 1,2,3,4,5,6,则通过一个栈可以输出序列 3,2,5,6,4,1. （ ）

【上海海运学院 1995 一、2（1 分） 1997 一、3（1 分）】

9. 栈和队列都是限制存取点的线性结构。（ ）【中科院软件所 1999 六、(5)（2 分）】

10. 若输入序列为 1，2，3，4，5，6，则通过一个栈可以输出序列 1，5，4，6，2，3。（ ）

【上海海运学院 1999 一、3（1 分）】

11. 任何一个递归过程都可以转换成非递归过程。（ ）【上海交通大学 1998 一、3（1 分）】

12. 只有那种使用了局部变量的递归过程在转换成非递归过程时才必须使用栈。（　　）
【上海交通大学 1998 一、4（1 分）】

13. 队列是一种插入与删除操作分别在表的两端进行的线性表，是一种先进后出型结构。（　　）
【上海海运学院 1998 一、3（1 分）】

14. 通常使用队列来处理函数或过程的调用。（　　）【南京航空航天大学 1997 一、5（1 分）】

15. 队列逻辑上是一个下端和上端既能增加又能减少的线性表。（　　）【上海交通大学 1998 一、2】

16. 循环队列通常用指针来实现队列的头尾相接。（　　）【南京航空航天大学 1996 六、1（1 分）】

17. 循环队列也存在空间溢出问题。（　　）【青岛大学 2002 一、2 （1 分）】

18. 队列和栈都是运算受限的线性表，只允许在表的两端进行运算。（ ）【长沙铁道学院 1997 一、5(1 分)】

19. 栈和队列都是线性表，只是在插入和删除时受到了一些限制。（　　）【北京邮电大学 2002 一、3(1 分)】

20. 栈和队列的存储方式，既可以是顺序方式，又可以是链式方式。（　　）
【上海海运学院 1996 一、2（1 分） 1999 一、2（1 分）】

三　填空题

1. 栈是_____的线性表，其运算遵循_____的原则。【北京科技大学 1997 一、3】

2. _____是限定仅在表尾进行插入或删除操作的线性表。【燕山大学 1998 一、3 （1 分）】

3. 一个栈的输入序列是：1，2，3 则不可能的栈输出序列是_____。【中国人民大学 2001 一、1(2 分)】

4. 设有一个空栈，栈顶指针为 1000H(十六进制)，现有输入序列为 1，2，3，4，5，经过 PUSH,PUSH,POP,PUSH,POP,PUSH,PUSH 之后，输出序列是_____，而栈顶指针值是_____H。设栈为顺序栈，每个元素占 4 个字节。【西安电子科技大学 1998 二、1（4 分）】

5. 当两个栈共享一存储区时，栈利用一维数组 stack(1,n)表示，两栈顶指针为 top[1]与 top[2]，则当栈 1 空时，top[1]为_____，栈 2 空时 ，top[2]为_____，栈满时为_____。
【南京理工大学 1997 三、1（3 分）】

6. 两个栈共享空间时栈满的条件_____。【中山大学 1998 一、3（1 分）】

7. 在作进栈运算时应先判别栈是否_(1)_;在作退栈运算时应先判别栈是否_(2)_；当栈中元素为 n 个，作进栈运算时发生上溢，则说明该栈的最大容量为_(3)_。
为了增加内存空间的利用率和减少溢出的可能性，由两个栈共享一片连续的空间时，应将两栈的_(4)_分别设在内存空间的两端，这样只有当_(5)_时才产生溢出。【山东工业大学 1994 一、1（5 分）】

8. 多个栈共存时，最好用_____作为存储结构。【南京理工大学 2001 二、7（2 分）】

9. 用 S 表示入栈操作，X 表示出栈操作，若元素入栈的顺序为 1234，为了得到 1342 出栈顺序，相应的 S 和 X 的操作串为_____。【西南交通大学 2000 一、5】

10. 顺序栈用 data[1..n]存储数据，栈顶指针是 top,则值为 x 的元素入栈的操作是_____。
【合肥工业大学 2001 三、2 （2 分）】

11. 表达式 23+((12*3-2)/4+34*5/7)+108/9 的后缀表达式是_____。【中山大学 1998 一、4（1 分）】

12. 循环队列的引入，目的是为了克服_____。【厦门大学 2001 一、1 （14/8 分）】

13. 用下标 0 开始的 N 元数组实现循环队列时，为实现下标变量 M 加 1 后在数组有效下标范围内循环，可采用的表达式是：M:=_____（填 PASCAL 语言，C 语言的考生不填）；M= _____（填 C 语言，PASCAL 语言的考生不填）。【西南交通大学 2000 一、7】

14. _____又称作先进先出表。【重庆大学 2000 一、7】

15. 队列的特点是_____。【北京理工大学 2000 二、2（2 分）】

16. 队列是限制插入只能在表的一端，而删除在表的另一端进行的线性表，其特点是_____。【北方交通大学 2001 二、5】

17. 已知链队列的头尾指针分别是 f 和 r，则将值 x 入队的操作序列是_____。【合肥工业大学 2000 三、3（2 分）】

18. 区分循环队列的满与空，只有两种方法，它们是_____和_____。【北京邮电大学 2001 二、2（4 分）】

19. 设循环队列用数组 A[1..M]表示，队首、队尾指针分别是 FRONT 和 TAIL，判定队满的条件为_____。【山东工业大学 1995 一、1（1 分）】

20. 设循环队列存放在向量 sq.data[0:M]中，则队头指针 sq.front 在循环意义下的出队操作可表示为_____，若用牺牲一个单元的办法来区分队满和队空（设队尾指针 sq.rear），则队满的条件为_____。【长沙铁道学院 1997 二、4 (4 分)】

21. 表达式求值是_____应用的一个典型例子。【重庆大学 2000 一、10】

22. 循环队列用数组 A[0..m-1]存放其元素值，已知其头尾指针分别是 front 和 rear ，则当前队列的元素个数是_____。【厦门大学 2000 六、1（16%/3 分）】

23. 设 Q[0..N-1]为循环队列，其头、尾指针分别为 P 和 R，则队 Q 中当前所含元素个数为_____。【北京科技大学 1997 一、4】

24. 完善下面算法。【中山大学 1998 四、2（6 分）】

后缀表达式求值，表达式 13/25+61 的后缀表达式格式为： 13, 25/61, +

```
FUNC  compute(a):real;    后缀表达式存储在数组 a[1..m]中。
BEGIN
   setnull(s); i:=1; ch:= (1)_____;
   WHILE ch<>' @'  DO
     BEGIN
       CASE ch OF
         '0' .. '9' :  x:=0;
                       WHILE ch<>',' DO
                         BEGIN
                           x:=x*10+ord(ch)-ord('0');
                           i:=i+1; ch:= (2)_____;
                         END
                '+' : x:=pop(s)+pop(s);
                '-' : x:=pop(s);x:=pop(s)-x;
                '*' : x:=pop(s)*pop(s);
                '/' : x:=pop(s);x:=pop(s)/x;
       ENDCASE
       push(s,x); i:=i+1; ch:=a[i];
```

```
    END;
  comput:= (3)_______;
 END;
```

25. 算术表达式求值的流程，其中 OPTR 为算术符栈，OPND 为操作数栈，precede(oper1，oper2)是比较运算符优先级别的函数，operate(opnd1,oper,opnd2)为两操作数的运算结果函数。(#表示运算起始和终止符号)【西北工业大学 1999 六、2 (7 分)】

```
FUNCTION  exp_reduced:operandtype;
INITSTACK(OPTR);PUSH(OPTR"#"); INITSTACK(OPND);read(w);
WHILE  NOT((w='#' ) AND (GETTOP(OPTR)='#')) DO
   IF NOT w in op THEN PUSH(OPND,w);
   ELSE CASE precede(GETTOP(OPTR),w)OF
            '<':[(1)_______; read(w);]
            '=':[(2)_______; read(w);];
            '>':[theta:=POP(OPTR);b:=POP(OPND);a:=POP(OPND);(3)_______; ]
        ENDC;
 RETURN(GETTOP(OPND));
 ENDF;
```

26. 根据需要，用适当的语句填入下面算法的_____中：

问题：设有 n 件物品，重量分别为 $w_1,w_2,w_3,\cdots,w_n$和一个能装载总重量为 T 的背包。能否从 n 件物品中选择若干件恰好使它们的重量之和等于 T。若能，则背包问题有解，否则无解。解此问题的算法如下：

```
FUNCTION  kanp_stack(VAR  stack,w:ARRAY[1..n]  OF  real;  VAR  top:integer;
T:real):boolean;
```

{w[1：n] 存放 n 件物品的重量，依次从中取出物品放入背包中，检查背包重量，若不超过 T，则装入，否则弃之，取下一个物品试之。若有解则返回函数值 true,否则返回 false}

```
BEGIN
  top:=0;  i:=1;                { i 指示待选物品}
  WHILE (1)_______ AND(2)_______DO
     [IF (3)_____ OR (4)_______ AND (i<n)
        THEN  [top := (5)_______ ;stack[top] :=i;{第 i 件物品装入背包}
               T:=T-w[i]];
      IF T=0 THEN RETURN ((6)_______)  {背包问题有解}
             ELSE  [IF  (i=n )  AND  (top>0)
                    THEN  [i:=(7)_______;{取出栈顶物品}
                           top:= (8)_______ ; T:= (9)_______ ];  {恢复 T 值}
                     i:=i+1          {准备挑选下一件物品}
                   ];
      ];
  RETURN((10)_______) {背包无解}
END;
```

【北京邮电大学 1996 四（10 分）】

四　应用题

1. 名词解释：栈。【燕山大学 1999 一、1（2 分）】【吉林工业大学 1999 一、3（2 分）】

2. 名词解释：队列【大连海事大学 1996 一、6（1分）】

3. 什么是循环队列？【哈尔滨工业大学 2001 三、2（3 分）】【河南大学 1998 一、4（3分）】

4. 假设以S和X分别表示入栈和出栈操作，则对初态和终态均为空的栈操作可由S和X组成的序列表示（如SXSX）。

（1）试指出判别给定序列是否合法的一般规则。

（2）两个不同合法序列（对同一输入序列）能否得到相同的输出元素序列？如能得到，请举列说明。

【东南大学 1992 二（10分）】

5. 有5个元素，其入栈次序为：A，B，C，D，E，在各种可能的出栈次序中，以元素C，D最先出栈（即C第一个且D第二个出栈）的次序有哪几个？【西南交通大学 2000 二、1】

6. 如果输入序列为1 2 3 4 5 6,试问能否通过栈结构得到以下两个序列:4 3 5 6 1 2和1 3 5 4 2 6;请说明为什么不能或如何才能得到。【武汉交通科技大学 1996 二、3 (3分)】

7. 若元素的进栈序列为：A、B、C、D、E，运用栈操作，能否得到出栈序列B、C、A、E、D和D、B、A、C、E？为什么？【北京科技大学 1998 一、2】

8. 设输入序列为 a,b,c,d,试写出借助一个栈可得到的两个输出序列和两个不能得到的输出序列。

【北京科技大学 2001 一、4（2分）】

9. 设输入序列为2，3，4，5，6，利用一个栈能得到序列2，5，3，4，6吗？栈可以用单链表实现吗？

【山东师范大学 1996 五、4（2分）】

10. 试证明：若借助栈由输入序列1,2,…,n得到输出序列为$P_1,P_2,\cdots,P_n$（它是输入序列的一个排列），则在输出序列中不可能出现这样的情形：存在着$i<j<k$,使$P_j<P_k<P_i$。【上海交通大学 1998 二（15分）】

11. 设一数列的输入顺序为123456，若采用堆栈结构，并以A和D分别表示入栈和出栈操作，试问通过入出栈操作的合法序列。

（1） 能否得到输出顺序为325641的序列。（5分）

（2） 能否得到输出顺序为154623的序列。（5分） 【北方交通大学 1995 一（10分）】

12.（1） 什么是递归程序？

（2） 递归程序的优、缺点是什么？

（3） 递归程序在执行时，应借助于什么来完成？

（4） 递归程序的入口语句、出口语句一般用什么语句实现？【大连海事大学 1996 二、4(4分)】

13. 设有下列递归算法:

```
FUNCTION  vol(n:integer):integer;
VAR     x :integer:
BEGIN IF n=0 THEN  vol:=0
              ELSE   BEGIN read(x); vol:=vol(n-1)+x; END;
END;
```

如该函数被调用时，参数n值为4,读入的x值依次为5,3,4,2，函数调用结束时返回值vol为多少?用图示描述函数执行过程中，递归工作栈的变化过程。【北京工业大学 1998 四 (10分)】

14. 当过程P递归调用自身时,过程P内部定义的局部变量在P的2次调用期间是否占用同一数据区？为什么？【山东师范大学 1999 一、4 （4分）】

15. 试推导出当总盘数为 n 的 Hanoi 塔的移动次数。【北京邮电大学 2001 四、3 （5 分）】

16. 对下面过程写出调用 P(3)的运行结果。

```
PROCEDURE p（w：integer）；
   BEGIN
      IF w>0 THEN
               BEGIN
                  p(w-1);
                  writeln(w);{输出 W}
                  p(w-1)
               END;
   END;
```

【西北大学 2001 三、7】

17. 用一个数组 S（设大小为 MAX）作为两个堆栈的共享空间。请说明共享方法，栈满/栈空的判断条件，并用 C 或 PASCAL 设计公用的入栈操作 push（i，x），其中 i 为 0 或 1，用于表示栈号，x 为入栈值。

【浙江大学 1998 五、2 (7 分)】

18. 简述下列程序段的功能。

```
PROC    algo(VAR S : stack;  k:integer);
VAR   T: stack; temp: integer;
    WHILE NOT  empty(S) DO
      [temp:=POP(S); IF temp<>k  THEN PUSH(T,temp)];
    WHILE NOT  empty(T)  DO   [temp:=POP(T);PUSH(S,temp)];
```

【山东科技大学 2002 一、1（4 分）】

19. 用栈实现将中缀表达式 8-(3+5)*(5-6/2)转换成后缀表达式，画出栈的变化过程图。

【南京航空航天大学 2001 五 （10 分）】

20. 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**为例说明)【东南大学 1993 一、2(6 分) 1997 一、1(8 分)】

21. 有递归算法如下：

```
FUNCTION  sum (n:integer):intger;
BEGIN
 IF n=0 THEN sum:=0
        ELSE BEGIN read(x); sum:=sum(n-1)+x END;
END;
```

设初值 n=4，读入 x=4,9,6,2

问：(1) 若 x 为局部变量时；该函数递归结束后返回调用程序的 sum=? 并画出在递归过程中栈状态的变化过程；

(2) 若 x 为全程变量递归结束时返回调用程序的 sum=?【北京邮电大学 1997 一 （10 分）】

22. 画出对算术表达式 A-B*C/D-E↑F 求值时操作数栈和运算符栈的变化过程。

【东南大学 2000 一、3（6 分）】

23. 计算算术表达式的值时，可用两个栈作辅助工具。对于给出的一个表达式，从左向右扫描它的字符，并将操作数放入栈 S1 中，运算符放入栈 S2 中，但每次扫描到运算符时，要把它同 S2 的栈顶运算符进行优先级比较，当扫描到的运算符的优先级不高于栈顶运算符的优

先级时，取出栈 S1 的栈顶和次栈顶的两个元素，以及栈 S2 的栈顶运算符进行运算将结果放入栈 S1 中（得到的结果依次用 T1、T2 等表示）。为方便比较，假设栈 S2 的初始栈顶为®（®运算符的优先级低于加、减、乘、除中任何一种运算）。现假设要计算表达式： A-B*C/D+E/F。写出栈 S1 和 S2 的变化过程。【山东科技大学 2001 一、4 （7 分）】

24. 有字符串次序为 3*-y-a/yˆ2,利用栈，给出将次序改为 3y-*ay2ˆ/-的操作步骤。（可用 X 代表扫描该字符串过程中顺序取一个字符进栈的操作，用 S 代表从栈中取出一个字符加入到新字符串尾的出栈操作。例如，ABC 变为 BCA 的操作步骤为 XXSXSS)【东北大学 2001 一、4 （ 4 分)】

25. 内存中一片连续空间（不妨假设地址从 1 到 m）提供给两个栈 S1 和 S2 使用，怎样分配这部分存储空间,使得对任一个栈,仅当这部分空间全满时才发生上溢。【东北大学 2000 一、1 (3 分)】

26. 将两个栈存入数组 V[1..m]应如何安排最好？这时栈空、栈满的条件是什么？【东南大学 1998 一、5】

27. 在一个算法中需要建立多个堆栈时可以选用下列三种方案之一，试问：这三种方案之间相比较各有什么优缺点？

（1）分别用多个顺序存储空间建立多个独立的堆栈；

（2）多个堆栈共享一个顺序存储空间；

（3）分别建立多个独立的链接堆栈。【北京航空航天大学 1998 一、6（4 分)】

28. 在某程序中，有两个栈共享一个一维数组空间 SPACE[N]、SPACE[0]、SPACE[N-1] 分别是两个栈的栈底。

（1）对栈 1、栈 2，试分别写出（元素 x）入栈的主要语句和出栈的主要语句。

（2）对栈 1、栈 2，试分别写出栈满、栈空的条件。【北京理工大学 1999 二、2（8 分)】

29. 简述顺序存储队列的假溢出的避免方法及队列满和空的条件。【山东大学 2000 一、2 (4 分)】

30. 举例说明顺序队的“假溢出”现象，并给出解决方案。【福州大学 1998 三、5 (6 分)】

31. 怎样判定循环队列的空和满？【燕山大学 1999 二、3（4 分)】

32. 简要叙述循环队列的数据结构，并写出其初始状态、队列空、队列满时的队首指针与队尾指针的值。

【南京航空航天大学 1995 七（5 分)】

33. 利用两个栈 s1,s2 模拟一个队列时，如何用栈的运算实现队列的插入，删除以及判队空运算。请简述这些运算的算法思想。【北京邮电大学 1992 一、1】【东南大学 1999 一、1 （7 分)】

34. 一个循环队列的数据结构描述如下：

```
TYPE sequeuetp=RECORD
                  elem：ARRAY[1..MAXSIZE] OF elemtp;
                  front，rear：0..MAXSIZE;
                END;
```

给出循环队列的队空和队满的判断条件,并且分析一下该条件对队列实际存储空间大小的影响，如果为了不损失存储空间，你如何改进循环队列的队空和队满的判断条件？【西北工业大学 1999 三 (8 分)】

35. 如果用一个循环数组 q[0..m-1]表示队列时，该队列只有一个队列头指针 front，不设队列尾指针 rear，而改置计数器 count 用以记录队列中结点的个数。

（1）编写实现队列的三个基本运算：判空、入队、出队（3 分）

（2）队列中能容纳元素的最多个数是多少？（1 分）【东北大学 2002 一、1】

36. 给出循环队列中元素个数的计算式(设队最大长度为N,队首指针FRONT,队尾指针REAR)
【西北大学 2000 二、7 (5分)】

37. 顺序队列一般应该组织成为环状队列的形式,而且一般队列头或尾其中之一应该特殊处理。例如，队列为 listarray[0..n-1]，队列头指针为 front，队列尾指针为 rear， 则listarray [rear]表示下一个可以插入队列的位置。请解释其原因。【北京大学 1999 一、3（20/3 分)】

38. 设一个双端队列，元素进入该队列的次序为 a，b，c，d。求既不能由输入受限的双端队列得到,又不能由输出受限的双端队列得到的输出序列。【中山大学 1999 一、4 （3 分)】

39. 若以 1、2、3、4 作为双端队列的输入序列，试分别求出以下条件的输出序列：
（1）能由输入受限的双端队列得到，但不能由输出受限的双端队列得到的输出序列；
（2）能由输出受限的双端队列得到，但不能由输入受限的双端队列得到的输出序列；
（3）既不能由输入受限的双端队列得到，也不能由输出受限的双端队列得到的输出序列。
【山东科技大学 2001 一、3 （6 分)】

40. 假设以数组 sq[0..7]存放循环队列元素,变量 f 指向队头元素的前一位置,变量 r 指向队尾元素，如用 A 和 D 分别表示入队和出队操作，请给出：
（1） 队空的初始条件;
（2）执行操作序列 $A^3D^1A^5D^2A^1D^2A^4$ 时的状态,并作必要的说明。【北方交通大学 1993 四(12分)】

41、设输入元素为 1、2、3、P 和 A，输入次序为 123PA，如图（编者略)。元素经过栈后达输出序列，当所有元素均到达输出序列后，有哪些序列可以作为高级语言的变量名。【中山大学 1997】

五 算法设计题

1. 设有两个栈 S_1,S_2 都采用顺序栈方式，并且共享一个存储区[0..maxsize-1],为了尽量利用空间，减少溢出的可能，可采用栈顶相向，迎面增长的存储方式。试设计 S_1,S_2 有关入栈和出栈的操作算法。
【哈尔滨工业大学 2001 七 （12 分)】

2. 设从键盘输入一整数的序列：a_1, a_2, a_3，…，a_n,试编写算法实现：用栈结构存储输入的整数，当 $a_i \neq -1$ 时，将 a_i 进栈；当 $a_i=-1$ 时，输出栈顶整数并出栈。算法应对异常情况（入栈满等）给出相应的信息。
【南京航空航天大学 1998 六 （10 分)】

3. 设表达式以字符形式已存入数组 E[n]中，‘#’为表达式的结束符，试写出判断表达式中括号（‘(’ 和 ‘)’）是否配对的 C 语言描述算法：EXYX(E);(注：算法中可调用栈操作的基本算法。)
【北京科技大学 2001 九、1 （10 分)】

4. 从键盘上输入一个逆波兰表达式，用伪码写出其求值程序。规定：逆波兰表达式的长度不超过一行，以$符作为输入结束，操作数之间用空格分隔,操作符只可能有+、-、*、/四种运算。例如：234 34+2*$
【山东师范大学 1999 七 （10 分)】

5. 假设以 I 和 O 分别表示入栈和出栈操作。栈的初态和终态均为空，入栈和出栈的操作序列可表示为仅由 I 和 O 组成的序列，称可以操作的序列为合法序列，否则称为非法序列。
（1）下面所示的序列中哪些是合法的？
A. IOIIOIOO　　B. IOOIOIIO　　C. IIIOIOIO　　D. IIIOOIOO
（2）通过对（1）的分析，写出一个算法，判定所给的操作序列是否合法。若合法，返回

true，否则返回 false（假定被判定的操作序列已存入一维数组中）。【武汉大学 2000 五、2】

6.设计一个算法，判断一个算术表达式中的括号是否配对。算术表达式保存在带头结点的单循环链表中，每个结点有两个域：ch 和 link，其中 ch 域为字符类型。【南京邮电大学 2000 五】

7. 请利用两个栈 S1 和 S2 来模拟一个队列。已知栈的三个运算定义如下：PUSH(ST,x):元素 x 入 ST 栈；POP(ST,x)：ST 栈顶元素出栈，赋给变量 x；Sempty(ST)：判 ST 栈是否为空。那么如何利用栈的运算来实现该队列的三个运算：enqueue:插入一个元素入队列；dequeue:删除一个元素出队列；queue_empty：判队列为空。（请写明算法的思想及必要的注释）

【西安电子科技大学 2001 软件五(10 分)】【上海交通大学 1999 二(12 分)】【河海大学 1998 三(12 分)】

类似本题的另外叙述有：

（1）有两个长度相同的栈 S1,S2，已知以下入栈、出栈、判栈满和判栈空操作：

```
PROCEDURE   push(Stack:Stacktype;x:Datatype);
FUNCTION    Pop(Stack:Stacktype ):Datatype;
FUNCTION    Full (Stack:Stacktype):Boolean;
FUNCTION    Empty(Stack:Stacktype)Boolean;
```

现用此二栈构成一个队列，试写出下面入队列、出队列操作算法：

```
PROCEDURE  EnQueue(x:Datatype);
FUNCTION   DeQueue: Datatype;
```

【北京邮电大学 2000 六（10 分）】

8. 设结点结构为(data,link)，试用一个全局指针 p 和某种链接结构实现一个队列,画出示意图,并给出入队 addq 和出队 deleteq 过程,要求它们的时间复杂性都是 O(1)（不计 new 和 dispose 时间）

【东南大学 1996 二 （10 分)】

9. 假设以带头结点的循环链表表示队列，并且只设一个指针指向队尾结点，但不设头指针，如图所示（编者略），请写出相应的入队列和出队列算法。【西安电子科技大学 1999 计应用六 （10 分)】

10. 如果允许在循环队列的两端都可以进行插入和删除操作。要求：

（1）写出循环队列的类型定义；

（2）写出“从队尾删除”和“从队头插入”的算法。【北方交通大学 1994 三 （12 分)】

11. 在一个循环链队中只有尾指针（记为 rear，结点结构为数据域 data，指针域 next），请给出这种队列的入队和出队操作的实现过程。【山东科技大学 2002 一、2 （6 分)】

12. 双端队列（duque）是一个可以在任一端进行插入和删除的线性表。现采用一个一维数组作为双端队列的数据存储结构，使用类 PASCAL 语言描述如下：

```
CONST maxsize=32;              {数组中可容纳的元素个数}
TYPE duque=RECORD
          elem: ARRAY[0..MAXSIZE-1] OF datatype;     {环形队列的存放数组}
          end1,end2:0..MAXSIZE;                      {环形数组的两端}
        END;
```

试编写两个算法 add(Qu:duque;x:datatype;tag:0..1)和 delete（Qu:duque; var x:datatype; tag:0..1）用以在此双端队列的任一端进行插入和删除。当 tag=0 时在左端 end1 端操作，当 tag=1 时在右端 end2 端操作。【清华大学 1998 二（10 分)】

13. 一个双端队列 deque 是限定在两端 end1，end2 都可进行插入和删除的线性表。队空条件是 end1=end2。若用顺序方式来组织双端队列，试根据下列要求，定义双端队列的结构，

并给出在指定端 i（i=1，2）的插入 enq 和删除 deq 操作的实现。

（1） 当队满时，最多只能有一个元素空间可以是空的。

（2） 在做两端的插入和删除时，队列中其它元素一律不动。【清华大学 1999 六（12 分）】

14. 已知 Q 是一个非空队列，S 是一个空栈。仅用队列和栈的 ADT 函数和少量工作变量，使用 Pascal 或 C 语言编写一个算法，将队列 Q 中的所有元素逆置。栈的 ADT 函数有：

```
makeEmpty(s:stack);                  置空栈
push(s:stack;value:datatype);        新元素 value 进栈
pop(s:stack):datatype;               出栈，返回栈顶值
isEmpty(s:stack):Boolean;            判栈空否
```

队列的 ADT 函数有：

```
enqueue(q:queue:value:datatype);  元素 value 进队
deQueue(q:queue):datatype;        出队列，返回队头值
isEmpty(q:queue):boolean;         判队列空否 【清华大学 2000 六（12 分）】
```

15. 将 n 个队列顺序映射到数组 v[1..m]中，每一队列在 v 中表示为一循环队列。试画出其示意图并写出对应这种表示的 addq 和 deleteq 过程。【东南大学 1993 二（20 分）】

16. 设整数序列 $a_1, a_2, \cdots, a_n$，给出求解最大值的递归程序。【南京航空航天大学 2000 六】

17. 线性表中元素存放在向量 A（1,…,n）中，元素是整型数。试写出递归算法求出 A 中的最大和最小元素。【北京邮电大学 1994 八 （10 分）】

18. 已知求两个正整数 m 与 n 的最大公因子的过程用自然语言可以表述为反复执行如下动作：第一步：若 n 等于零，则返回 m；第二步：若 m 小于 n，则 m 与 n 相互交换；否则，保存 m，然后将 n 送 m，将保存的 m 除以 n 的余数送 n。

（1）将上述过程用递归函数表达出来（设求 x 除以 y 的余数可以用 x MOD y 形式表示）。

（2）写出求解该递归函数的非递归算法。【北京航空航天大学 2001 五（15 分）】

19. 写出和下列递归过程等价的非递归过程。

```
PROCEDURE   test(VAR  sum:integer);
VAR  a:integer,
BEGIN
  read(a);
  IF a=0  THEN  sum:=1
          ELSE  BEGIN test(sum); sum:=sum*a; END;
 write(sum);
END;                    【清华大学 1996 二】
```

20. 试将下列递归过程改写为非递归过程。

```
void  test(int  &sum)
{ int  x;
  scanf(x);
  if(x=0) sum=0 else {test(sum); sum+=x;}
  printf(sum);
}          【北京轻工业学院 2000 三 （15 分）】
```

21. 已知 Ackermann 函数定义如下：

$$Ack(m,n)=\begin{cases} n+1 & \text{当 } m=0 \text{ 时} \\ Ack(m-1,1) & \text{当 } m\neq 0, n=0 \text{ 时} \\ Ack(m-1,Ack(m,n-1)) & \text{当 } m\neq 0, n\neq 0 \text{ 时} \end{cases}$$

（1） 写出 Ack(2,1)的计算过程。

（2） 写出计算 Ack(m,n)的非递归算法。 【北京航空航天大学 1999 六 （15 分）】

22. 设计算法以求解从集合{1..n}中选取 k（k<=n）个元素的所有组合。例如，从集合{1..4}中选取 2 个元素的所有组合的输出结果为：1 2，1 3，1 4，2 3， 2 4，3 4。

【合肥工业大学 2000 五、5（8 分）】

第三章 栈和队列（答案）

一、选择题

1.B	2.1B	2.2A	2.3B	2.4D	2.5.C	3.B	4.D	5.D	6.C	7.D	8.B
9.D	10.D	11.D	12.C	13.B	14.C	15.B	16.D	17.B	18.B	19.B	20.D
21.D	22.D	23.D	24.C	25.A	26.A	27.D	28.B	29.BD	30.C	31.B	32.C
33.1B	33.2A	33.3C	33.4C	33.5F	34.C	35.C	36.A	37.AD			

二、判断题

1.√	2.√	3.√	4.√	5.×	6.√	7.√	8.√	9.√	10.×	11.√	12.×
13.×	14.×	15.√	16.×	17.√	18.×	19.√	20.√				

部分答案解释如下。

1、 尾递归的消除就不需用栈

2、 这个数是前序序列为 1,2,3,...,n，所能得到的不相似的二叉树的数目。

三、填空题

1、操作受限（或限定仅在表尾进行插入和删除操作） 后进先出

2、栈 3、3 1 2 4、23 100CH 5、0 n+1 top[1]+1=top[2]

6、两栈顶指针值相减的绝对值为 1（或两栈顶指针相邻）。

7、(1)满 (2)空 (3)n (4)栈底 (5)两栈顶指针相邻（即值之差的绝对值为 1）

8、链式存储结构 9、S×SS×S×× 10、data[++top]=x;

11、23.12.3*2-4/34.5*7/++108.9/+（注：表达式中的点(.)表示将数隔开，如 23.12.3 是三个数）

12、假溢出时大量移动数据元素。

13、(M+1) MOD N (M+1)% N; 14、队列 15、先进先出 16、先进先出

17、s=(LinkedList)malloc(**sizeof**(LNode)); s->data=x;s->next=r->next;r->next=s;r=s;

18、牺牲一个存储单元 设标记

19、(TAIL+1) MOD M=FRONT (数组下标 0 到 M-1，若一定使用 1 到 M，则取模为 0 者，值改取 M

20 、 sq.front=(sq.front+1)%(M+1) ; **return**(sq.data(sq.front)) ; (sq.rear+1)%(M+1)==sq.front;

21、栈 22、(rear-front+m) % m; 23、(R-P+N) % N;

24、(1) a[i]或 a[1] (2) a[i] (3) pop (s) 或 s[1];

25、(1) PUSH (OPTR, w) (2) POP (OPTR) (3) PUSH (OPND, operate (a, theta, b))

26、(1) T>0 (2) i<n (3) T>0 (4) top<n (5) top+1 (6) true (7) i-1 (8) top-1 (9) T+w[i] (10) false

四、应用题

1、栈是只准在一端进行插入和删除操作的线性表，允许插入和删除的一端叫栈顶，另一端叫栈底。最后插入的元素最先删除，故栈也称后进先出（LIFO）表。

2、队列是允许在一端插入而在另一端删除的线性表，允许插入的一端叫队尾，允许删除的一端叫队头。最先插入队的元素最先离开（删除），故队列也常称先进先出（FIFO）表。

3、用常规意义下顺序存储结构的一维数组表示队列，由于队列的性质（队尾插入和队头删除），容易造成“假溢出”现象，即队尾已到达一维数组的高下标，不能再插入，然而队中元素个数小于队列的长度（容量）。循环队列是解决“假溢出”的一种方法。通常把一维数组看成首尾相接。在循环队列下，通常采用“牺牲一个存储单元”或“作标记”的方法解决“队满”和“队空”的判定问题。

4、（1）通常有两条规则。第一是给定序列中 S 的个数和 X 的个数相等；第二是从给定序列的开始，到给定序列中的任一位置，S 的个数要大于或等于 X 的个数。

（2）可以得到相同的输出元素序列。例如，输入元素为 A，B，C，则两个输入的合法序列 ABC 和 BAC 均可得到输出元素序列 ABC。对于合法序列 ABC，我们使用本题约定的 S×S×S×操作序列；对于合法序列 BAC，我们使用 SS××S×操作序列。

5、三个：CDEBA，CDBEA，CDBAE

6、输入序列为 123456，不能得出 435612，其理由是，输出序列最后两元素是 12，前面 4 个元素（4356）得到后，栈中元素剩 12，且 2 在栈顶，不可能栈底元素 1 在栈顶元素 2 之前出栈。

得到 135426 的过程如下：1 入栈并出栈，得到部分输出序列 1；然后 2 和 3 入栈，3 出栈，部分输出序列变为：13；接着 4 和 5 入栈，5，4 和 2 依次出栈，部分输出序列变为 13542；最后 6 入栈并退栈，得最终结果 135426。

7、能得到出栈序列 B、C、A、E、D，不能得到出栈序列 D、B、A、C、E。其理由为：若出栈序列以 D 开头，说明在 D 之前的入栈元素是 A、B 和 C，三个元素中 C 是栈顶元素，B 和 A 不可能早于 C 出栈，故不可能得到 D、B、A、C、E 出栈序列。

8、借助栈结构，n 个入栈元素可得到 1/(n+1)((2n)! / (n!*n!))种出栈序列。本题 4 个元素，可有 14 种出栈序列，abcd 和 dcba 就是其中两种。但 dabc 和 adbc 是不可能得到的两种。

9、不能得到序列 2，5，3，4，6。栈可以用单链表实现，这就是链栈。由于栈只在栈顶操作，所以链栈通常不设头结点。

10、如果 i<j，则对于 $p_i<p_j$ 情况，说明 p_i 在 p_j 入栈前先出栈。而对于 $p_i>p_j$ 的情况，则说明要将 p_j 压到 p_i 之上，也就是在 p_j 出栈之后 p_i 才能出栈。这就说明，对于 i<j<k，不可能出现 $p_j<p_k<p_i$ 的输出序列。换句话说，对于输入序列 1，2，3，不可能出现 3，1，2 的输出序列。

11、（1）能得到 325641。在 123 依次进栈后，3 和 2 出栈，得部分输出序列 32；然后 4，5 入栈，5 出栈，得部分出栈序列 325；6 入栈并出栈，得部分输出序列 3256；最后退栈，直到栈空。得输出序列 325641。其操作序列为 AAADDAADADDD。

（2）不能得到输出顺序为 154623 的序列。部分合法操作序列为 ADAAAADDAD，得到部分输出序列 1546 后，栈中元素为 23，3 在栈顶，故不可能 2 先出栈，得不到输出序列 154623。

12、（1）一个函数在结束本函数之前，直接或间接调用函数自身，称为递归。例如，函数 f 在执行中，又调用函数 f 自身，这称为直接递归；若函数 f 在执行中，调用函数 g，而 g 在执行中，又调用函数 f，这称为间接递归。在实际应用中，多为直接递归，也常简称为递归。

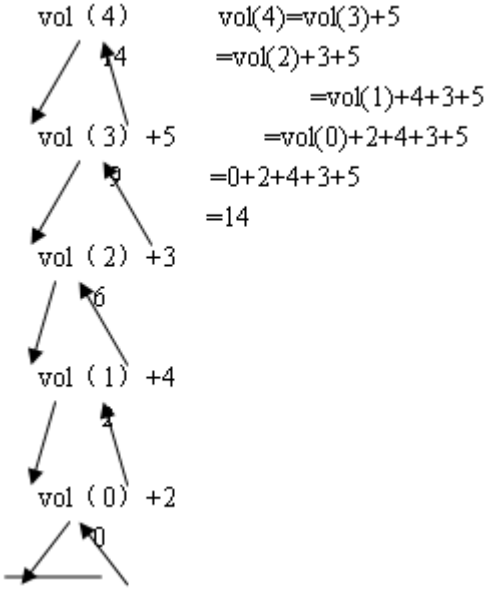
（2）递归程序的优点是程序结构简单、清晰，易证明其正确性。缺点是执行中占内

存空间较多，运行效率低。

（3）递归程序执行中需借助栈这种数据结构来实现。

（4）递归程序的入口语句和出口语句一般用条件判断语句来实现。递归程序由基本项和归纳项组成。基本项是递归程序出口，即不再递归即可求出结果的部分；归纳项是将原来问题化成简单的且与原来形式一样的问题，即向着“基本项”发展，最终“到达”基本项。

13、函数调用结束时 vol=14。执行过程图示如下：



14、过程 p 递归调用自身时，过程 p 由内部定义的局部变量在 p 的 2 次调用期间，不占同一数据区。每次调用都保留其数据区，这是递归定义所决定，用“递归工作栈”来实现。

15、设 H_n 为 n 个盘子的 Hanoi 塔的移动次数。(假定 n 个盘子从钢针 X 移到钢针 Z，可借助钢针 Y)

则 $H_n = 2H_{n-1}+1$ //先将 n-1 个盘子从 X 移到 Y，第 n 个盘子移到 Z，再将那 n-1 个移到 Z

$=2（2H_{n-2}+1）+1$

$=2^2 H_{n-2}+2+1$

$=2^2（2H_{n-3}+1）+2+1$

$=2^3 H_{n-3}+2^2+2+1$

•

•

•

$= 2^k H_{n-k}+2^{k-1}+2^{k-2}+\cdots+2^1+2^0$

$=2^{n-1} H_1+2^{n-2}+2^{n-3}+\cdots+2^1+2^0$

因为 $H_1=1$，所以原式 $H_n=2^{n-1}+2^{n-2}+\cdots+2^1+2^0=2^n-1$

故总盘数为 n 的 Hanoi 塔的移动次数是 2^n-1。

16、运行结果为：1 2 1 3 1 2 1（注：运行结果是每行一个数，为节省篇幅，放到一行。）

17、两栈共享一向量空间（一维数组），栈底设在数组的两端，两栈顶相邻时为栈满。设共享数组为 S[MAX]，则一个栈顶指针为-1，另一个栈顶指针为 MAX 时，栈为空。

用 C 写的入栈操作 push（i，x）如下：

```
const  MAX=共享栈可能达到的最大容量
typedef   struct  node
   {elemtype  s[MAX];
    int  top[2];
   }anode;
anode ds;
int push(int i,elemtype x)
```

//ds 为容量有 MAX 个类型为 elemtype 的元素的一维数组，由两个栈共享其空间。i 的值为 0 或 1，x 为类型为 elemtype 的元素。本算法将 x 压入栈中。如压栈成功，返回 1；否则，返回 0。

```
{if（ds.top[1]-ds.top[0]==1）{printf（“栈满\n”）；return（0）；}
 switch（i）
  {case  0：ds.s[++ds.top[i]]=x；break；
   case  1：ds.s[--ds.top[i]]=x；
   return（1）；}//入栈成功。
}
```

18、本程序段查找栈 S 中有无整数为 k 的元素，如有，则删除。采用的办法使用另一个栈 T。在 S 栈元素退栈时，若退栈元素不是整数 k，则压入 T 栈。遇整数 k，k 不入 T 栈，然后将 T 栈元素全部退栈，并依次压入栈 S 中，实现了在 S 中删除整数 k 的目的。若 S 中无整数 k，则在 S 退成空栈后，再将 T 栈元素退栈，并依次压入 S 栈。直至 T 栈空。这后一种情况下 S 栈内容操作前后不变。

19、中缀表达式 8-(3+5)*(5-6/2）的后缀表达式是： 8 3 5 + 5 6 2 / - * -

栈的变化过程图略（请参见 22 题），表达式生成过程为：

中缀表达式 exp1 转为后缀表达式 exp2 的规则如下：

设操作符栈 s，初始为空栈后，压入优先级最低的操作符‘#’。对中缀表达式从左向右扫描，遇操作数，直接写入 exp2；若是操作符（记为 w），分如下情况处理，直至表达式 exp1 扫描完毕。

（1）w 为一般操作符（’+’，’-‘，’*’，’/’等），要与栈顶操作符比较优先级，若 w 优先级高于栈顶操作符，则入栈；否则，栈顶运算符退栈到 exp2，w 再与新栈顶操作符作上述比较处理，直至 w 入栈。

（2）w 为左括号（’（’），w 入栈。

（3）w 为右括号（’）’），操作符栈退栈并进入 exp2，直到碰到左括号为止，左括号退栈（不能进入 exp2），右括号也丢掉，达到 exp2 中消除括号的目的。

（4）w 为‘#’，表示中缀表达式 exp1 结束，操作符栈退栈到 exp2，直至碰到‘#’，退栈，整个操作结束。

这里，再介绍一种简单方法。中缀表达式转为后缀表达式有三步：首先，将中缀表达式中所有的子表达式按计算规则用嵌套括号括起来；接着，顺序将每对括号中的运算符移到相应括号的后面；最后，删除所有括号。

例如，将中缀表达式 8-(3+5)*(5-6/2）转为后缀表达式。按如上步骤：

执行完上面第一步后为：(8-((3+5)*(5-(6/2))))；

执行完上面第二步后为：(8((35)+(5(62)/)-)*)- ；

执行完上面第三步后为：8 3 5 + 5 6 2 / - * - 。

可用类似方法将中缀表达式转为前缀表达式。

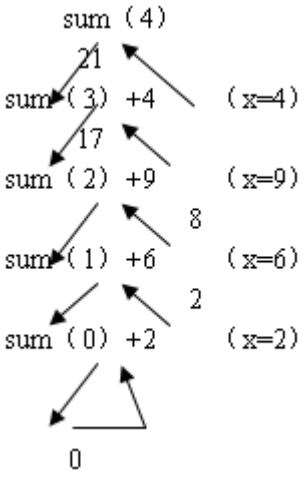
20、中缀表达式转为后缀表达式的规则基本上与上面 19 题相同，不同之处是对运算符

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*，这里规定在读入*后，不能立即当乘号处理，要看下一个符号，若下个符号不是*，则前个*是乘号。这里因为下一个待读的符号也是*，故认为**是一个运算符，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**，我们规定，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。

21、（1）sum=21。当 x 为局部变量时，每次递归调用，都要给局部变量分配存储单元，故 x 数值 4，9，6 和 2 均保留，其递归过程示意图如下：



(2) sum=8,当 x 为全局变量时，在程序的整个执行期间，x 只占一个存储单元，先后读入的 4 个数(4,9,6,2),仅最后一个起作用。当递归调用结束，逐层返回时 sum:=sum(n-1)+x 表达式中，x 就是 2，所以结果为 sum=8。

22、设操作数栈是 opnd，操作符栈是 optr，对算术表达式 A-B*C/D-E↑F 求值，过程如下：

步骤	opnd栈	optr栈	输入字符	主要操作
初始		#	A-B*C/D-E↑F#	PUSH(OPTR,'#')
1	A	#	A-B*C/D-E↑F#	PUSH(OPND,A)
2	A	# -	-B*C/D-E↑F#	PUSH(OPTR,'-')
3	AB	# -	B*C/D-E↑F#	PUSH(OPND,B)
4	AB	# - *	*C/D-E↑F#	PUSH(OPTR,'*')
5	ABC	# - *	C/D-E↑F#	PUSH(OPND,C)
6	AT(T=B*C)	# - /	/D-E↑F#	PUSH(OPND,POP(OPND)*POP(OPND)) PUSH(OPTR,'/')
7	ATD	# - /	D-E↑F#	PUSH(OPND,D)
8	AT(T=T/D) T(T=A-T)	# - # -	-E↑F#	x=POP(OPND);y=POP(OPND) PUSH(OPND,y/x); x=POP(OPND);y=POP(OPND); PUSH(OPND,y-x) PUSH(OPTR,'-')
9	TE	# -	E↑F#	PUSH(OPND,E)
10	TE	# -↑	↑F#	PUSH(OPTR, '↑')
11	TEF	# -↑	F#	PUSH(OPND,F)
12	TE TS(S=E↑F) R(R=T-S)	 #- #	#	X=POP(OPND) Y=POP(OPND) POP(OPTR) PUSH(OPND,y↑x) x=POP(OPND) y=POP(OPND) POP(OPTR) PUSH(OPND,y-x)

步骤	栈S1	栈S2	输入的算术表达式（按字符读入）
初始	_	®	A-B*C/D+E/F®
1	A	®	A-B*C/D+E/F®
2	A	®-	-B*C/D+E/F®
3	AB	®-	B*C/D+E/F®
4	AB	®-*	*C/D+E/F®
5	ABC	®-*	C/D+E/F®
6	AT_1（注：T_1=B*C）	®-/	/D+E/F®
7	AT_1D	®-/	D+E/F®
8	AT_2（注：T_2=T1/D） T_3（注：T_3=A-T_2）	®- ®+	+E/F®
9	T_3E	®+	E/F®
10	T_3E	®+/	/F®
11	T_3EF	®+/	F®
12	T_3T_4（注：T_4=E/F） T_5（注：T_5= T_3+ T_4）	®+ ®	®

24、XSXXXSSSXXSXXSXXSSSS

25、S1 和 S2 共享内存中一片连续空间（地址 1 到 m），可以将 S1 和 S2 的栈底设在两端，两栈顶向共享空间的中心延伸，仅当两栈顶指针相邻（两栈顶指针值之差的绝对值等于 1）时，判断为栈满，当一个栈顶指针为 0，另一个栈顶指针 m+1 时为两栈均空。

26、设栈 S1 和栈 S2 共享向量 V[1..m]，初始时，栈 S1 的栈顶指针 top[0]=0，栈 S2 的栈顶指针 top[1]=m+1，当 top[0]=0 为左栈空，top[1]=m+1 为右栈空；当 top[0]=0 并且 top[1]=m+1 时为全栈空。当 top[1]-top[0]=1 时为栈满。

27、（1）每个栈仅用一个顺序存储空间时，操作简便，但分配存储空间小了，容易产生溢出，分配空间大了，容易造成浪费，各栈不能共享空间。

（2）多个栈共享一个顺序存储空间，充分利用了存储空间，只有在整个存储空间都用完时才能产生溢出，其缺点是当一个栈满时要向左、右栈查询有无空闲单元。如果有，则要移动元素和修改相关的栈底和栈顶指针。当接近栈满时，查询空闲单元、移动元素和修改栈底栈顶指针的操作频繁，计算复杂并且耗费时间。

（3）多个链栈一般不考虑栈的溢出（仅受用户内存空间限制），缺点是栈中元素要以指针相链接，比顺序存储多占用了存储空间。

28、设 top1 和 top2 分别为栈 1 和 2 的栈顶指针

（1）入栈主要语句

```
if(top2-top1==1) {printf(“栈满\n”); exit(0);}
```

```
case1:top1++; SPACE[top1]=x;     //设 x 为入栈元素。
case2:top2--; SPACE[top2]=x;
出栈主要语句
case1: if (top1==-1)  {printf ("栈空\n"); exit (0); }
       top1--; return (SPACE[top1+1]);      //返回出栈元素。
case2: if (top2==N) {printf ("栈空\n"); exit (0); }
       top2++; return (SPACE[top2-1]);      //返回出栈元素。
```

（2）栈满条件：top2-top1=1

栈空条件：top1=-1 并且 top2=N　//top1=-1 为左栈空，top2=N 为右栈空

29、设顺序存储队列用一维数组 q[m]表示，其中 m 为队列中元素个数，队列中元素在向量中的下标从 0 到 m-1。设队头指针为 front，队尾指针是 rear，约定 front 指向队头元素的前一位置，rear 指向队尾元素。当 front 等于-1 时队空，rear 等于 m-1 时为队满。由于队列的性质（"删除"在队头而"插入"在队尾），所以当队尾指针 rear 等于 m-1 时，若 front 不等于-1，则队列中仍有空闲单元，所以队列并不是真满。这时若再有入队操作，会造成假"溢出"。其解决办法有二，一是将队列元素向前"平移"（占用 0 至 rear-front-1）；二是将队列看成首尾相连，即循环队列（0..m-1）。在循环队列下，仍定义 front=rear 时为队空，而判断队满则用两种办法，一是用"牺牲一个单元"，即 rear+1=front（准确记是（rear+1）%m=front，m 是队列容量）时为队满。另一种解法是"设标记"方法，如设标记 tag，tag 等于 0 情况下，若删除时导致 front=rear 为队空；tag=1 情况下，若因插入导致 front=rear 则为队满。

30、见上题 29 的解答。　31、参见上面 29 题。

32、
```
typedef struct node
{elemtype elemcq[m];  //m 为队列最大可能的容量。
 int front ,rear;     //front 和 rear 分别为队头和队尾指针。
}cqnode;
cqnode  cq;
```
(1) 初始状态

```
cq.front=cq.rear=0;
```

(2) 队列空

```
cq.front==cq.rear;
```

(3) 队列满

```
(cq.rear+1)%m==cq.front;
```

33、栈的特点是后进先出，队列的特点是先进先出。初始时设栈 s1 和栈 s2 均为空。

（1）用栈 s1 和 s2 模拟一个队列的输入：设 s1 和 s2 容量相等。分以下三种情况讨论：若 s1 未满，则元素入 s1 栈；若 s1 满，s2 空，则将 s1 全部元素退栈，再压栈入 s2，之后元素入 s1 栈；若 s1 满，s2 不空（已有出队列元素），则不能入队。

（2）用栈 s1 和 s2 模拟队列出队（删除）：若栈 s2 不空，退栈，即是队列的出队；若 s2 为空且 s1 不空，则将 s1 栈中全部元素退栈，并依次压入 s2 中，s2 栈顶元素退栈，这就是相当于队列的出队。若栈 s1 为空并且 s2 也为空，队列空，不能出队。

（3）判队空 若栈 s1 为空并且 s2 也为空，才是队列空。

讨论：s1 和 s2 容量之和是队列的最大容量。其操作是，s1 栈满后，全部退栈并压栈入 s2（设 s1 和 s2 容量相等）。再入栈 s1 直至 s1 满。这相当队列元素"入队"完毕。出队时，s2 退栈完毕后，s1 栈中元素依次退栈到 s2，s2 退栈完毕，相当于队列中全部元素出队。

在栈 s2 不空情况下，若要求入队操作，只要 s1 不满，就可压入 s1 中。若 s1 满和 s2

不空状态下要求队列的入队时，按出错处理。

34、（1）队空 s.front=s.rear;　　//设 s 是 sequeuetp 类型变量

（2）队满：（s.rear+1）MOD MAXSIZE=s.front　//数组下标为 0.. MAXSIZE-1

具体参见本章应用题第 29 题

35、typedef struct

{elemtp q[m];

int front,count;　//front 是队首指针，count 是队列中元素个数。

}cqnode;　　//定义类型标识符。

(1)判空：int Empty(cqnode　cq)　　//cq 是 cqnode 类型的变量

{if(cq.count==0) return(1); else return(0);　//空队列}

入队：int EnQueue(cqnode cq，elemtp x)

{if(count==m){printf(“队满\n”); exit(0); }

cq.q[(cq.front+count)%m]=x;　　//x 入队

count++; return(1);　　//队列中元素个数增加 1,入队成功。

}

出队：int DelQueue(cqnode cq)

{if (count==0){printf(“队空\n”); return(0);}

printf(“出队元素”, cq.q[cq.front]);

x=cq.q[cq.front];

cq.front=(cq.front+1)%m;　//计算新的队头指针。

return(x)

}

(2) 队列中能容纳的元素的个数为 m。队头指针 front 指向队头元素。

36、循环队列中元素个数为（REAR-FRONT+N）%N。其中 FRONT 是队首指针，指向队首元素的前一位置；REAR 是队尾指针，指向队尾元素；N 是队列最大长度。

37、循环队列解决了用向量表示队列所出现的“假溢出”问题，但同时又出现了如何判断队列的满与空问题。例如：在队列长 10 的循环队列中，若假定队头指针 front 指向队头元素的前一位置，而队尾指针指向队尾元素，则 front=3，rear=7 的情况下，连续出队 4 个元素，则 front==rear 为队空；如果连续入队 6 个元素，则 front==rear 为队满。如何判断这种情况下的队满与队空，一般采取牺牲一个单元的做法或设标记法。即假设 front==rear 为队空，而（rear+1）%表长==front 为队满，或通过设标记 tag。若 tag=0，front==rear 则为队空；若 tag=1，因入队而使得 front==rear，则为队满。

本题中队列尾指针 rear，指向队尾元素的下一位置,listarray[rear]表示下一个入队的元素。在这种情况下，我们可规定，队头指针 front 指向队首元素。当 front==rear 时为队空，当（rear+1）%n=front 时为队满。出队操作（在队列不空情况下）队头指针是 front=（front+1）%n，

38、既不能由输入受限的双端队列得到，也不能由输出受限的双端队列得到的输出序列是 dbca。

39、（1）4132　　（2）4213　　（3）4231

40、（1）队空的初始条件：f=r=0;

（2）执行操作 A^3 后，r=3; // A^3 表示三次入队操作

执行操作 D^1 后，f=1; //D^1 表示一次出队操作

执行操作 A^5 后，r=0;

执行操作 D^2 后，f=3;

执行操作 A^1后，r=1；
执行操作 D^2后，f=5；
执行操作 A^4后，按溢出处理。因为执行 A^3后，r=4，这时队满，若再执行 A 操作，则出错。

41．一般说，高级语言的变量名是以字母开头的字母数字序列。故本题答案是:
AP321,PA321,P3A21,P32A1,P321A。

五、算法设计题

1、[题目分析]两栈共享向量空间，将两栈栈底设在向量两端，初始时，s1 栈顶指针为-1，s2 栈顶为 maxsize。两栈顶指针相邻时为栈满。两栈顶相向，迎面增长，栈顶指针指向栈顶元素。

```
#define maxsize   两栈共享顺序存储空间所能达到的最多元素数
#define elemtp int        //假设元素类型为整型
typedef struct
{elemtp stack[maxsize];  //栈空间
 int top[2];             //top 为两个栈顶指针
}stk;
stk s;                   //s 是如上定义的结构类型变量，为全局变量。
```

(1)入栈操作：

```
int push(int i,int x)
//入栈操作。i 为栈号，i=0 表示左边的栈 s1，i=1 表示右边的栈 s2，x 是入栈元素。
入栈成功返回 1，否则返回 0。
{if(i<0||i>1){printf(“栈号输入不对”);exit(0);}
if(s.top[1]-s.top[0]==1) {printf(“栈已满\n”);return(0);}
switch(i)
 {case 0: s.stack[++s.top[0]]=x; return(1); break;
  case 1: s.stack[--s.top[1]]=x; return(1);
 }
}//push
```

（2） 退栈操作

```
elemtp pop(int i)
//退栈算法。i 代表栈号，i=0 时为 s1 栈，i=1 时为 s2 栈。退栈成功返回退栈元素，
 否则返回-1。
{if(i<0 || i>1){printf(“栈号输入错误\n”); exit(0);}
  switch(i)
   {case 0: if(s.top[0]==-1) {printf(“栈空\n”); return (-1); }
            else return(s.stack[s.top[0]--]);
    case 1: if(s.top[1]==maxsize {printf(“栈空\n”); return(-1);}
            else return(s.stack[s.top[1]++]);
   }
}//算法结束
```

[算法讨论] 请注意算法中两栈入栈和退栈时的栈顶指针的计算。两栈共享空间示意图略，s1 栈是通常意义下的栈，而 s2 栈入栈操作时，其栈顶指针左移（减 1），退栈时，栈顶指针右移（加 1）。

2、#define maxsize 栈空间容量

```
void InOutS(int s[maxsize])
   //s 是元素为整数的栈，本算法进行入栈和退栈操作。
{int top=0;              //top 为栈顶指针，定义 top=0 时为栈空。
 for(i=1; i<=n; i++)     //n 个整数序列作处理。
  {scanf(“%d”,&x);       //从键盘读入整数序列。
   if(x!=-1)             // 读入的整数不等于-1 时入栈。
    if(top==maxsize-1){printf(“栈满\n”);exit(0);}else s[++top]=x; //x 入栈。
    else   //读入的整数等于-1 时退栈。
    {if(top==0){printf( “ 栈 空 \n ” );exit(0);}  else  printf( “ 出 栈 元 素 是%d\n”,s[top--]); }}
  }//算法结束。
```

3、[题目分析]判断表达式中括号是否匹配，可通过栈，简单说是左括号时进栈，右括号时退栈。退栈时，若栈顶元素是左括号，则新读入的右括号与栈顶左括号就可消去。如此下去，输入表达式结束时，栈为空则正确，否则括号不匹配。

```
int EXYX(char E[],int n)
//E[]是有 n 字符的字符数组，存放字符串表达式，以‘#’结束。本算法判断表达式中圆括号是否匹配。
{char s[30];           //s 是一维数组，容量足够大，用作存放括号的栈。
 int top=0;            //top 用作栈顶指针。
 s[top]= ‘#’;          //‘#’先入栈，用于和表达式结束符号‘#’匹配。
 int i=0;              //字符数组 E 的工作指针。
 while(E[i]!= ‘#’)  //逐字符处理字符表达式的数组。
   switch(E[i])
   {case ‘(’ :  s[++top]= ‘(’ ; i++ ;  break ;
    case ‘)’ :  if(s[top]== ‘(’ {top--; i++; break;}
                  else{printf(“括号不配对”);exit(0);}
    case ‘#’ :  if(s[top]== ‘#’ ){printf(“括号配对\n”);return (1);}
                else {printf(“ 括号不配对\n”);return (0);} //括号不配对
    default :    i++;    //读入其它字符，不作处理。
   }
 }//算法结束。
```

[算法讨论]本题是用栈判断括号匹配的特例：只检查圆括号的配对。一般情况是检查花括号（‘{’, ‘}’）、方括号（‘[’, ‘]’）和圆括号（‘(’, ‘)’）的配对问题。编写算法中如遇左括号（‘{’, ‘[’, 或 ‘(’）就压入栈中，如遇右括号（‘}’, ‘]’, 或 ‘)’），则与栈顶元素比较，如是与其配对的括号（左花括号，左方括号或左圆括号），则弹出栈顶元素；否则，就结论括号不配对。在读入表达式结束符 ‘#’ 时，栈中若应只剩 ‘#’，表示括号全部配对成功；否则表示括号不匹配。

另外，由于本题只是检查括号是否匹配，故对从表达式中读入的不是括号的那些字符，一律未作处理。再有，假设栈容量足够大，因此入栈时未判断溢出。

4、[题目分析]逆波兰表达式(即后缀表达式)求值规则如下：设立运算数栈 OPND,对表达式从左到右扫描(读入)，当表达式中扫描到数时，压入 OPND 栈。当扫描到运算符时，从 OPND 退出两个数，进行相应运算，结果再压入 OPND 栈。这个过程一直进行到读出表达式结束符$，这时 OPND 栈中只有一个数，就是结果。

```
float expr( )
 //从键盘输入逆波兰表达式，以‘$’表示输入结束，本算法求逆波兰式表达式的值。
  {float OPND[30];   // OPND 是操作数栈。
   init(OPND);       //两栈初始化。
   float num=0.0;    //数字初始化。
   scanf (“%c”,&x);//x 是字符型变量。
   while(x!=’ $’ )
    {switch
       {case ‘0’ <=x<=’ 9’ :while((x>=’ 0’ &&x<=’ 9’ )||x==’ .’ )  //拼数
                            if(x!=’ .’ )   //处理整数
                             {num=num*10+(ord(x)-ord(‘0’)); scanf(“%c”,&x);}
                            else           //处理小数部分。
                             {scale=10.0; scanf(“%c”,&x);
                              while(x>=’ 0’ &&x<=’ 9’ )
                                {num=num+(ord(x)-ord( ‘0’ )/scale;
                                 scale=scale*10;  scanf(“%c”,&x); }
                             }//else
                            push(OPND,num); num=0.0;//数压入栈，下个数初始化
        case x= ‘ ’ :break;  //遇空格，继续读下一个字符。
        case x= ‘+’ :push(OPND,pop(OPND)+pop(OPND));break;
        case x= ‘-’ :x1=pop(OPND);x2=pop(OPND);push(OPND,x2-x1);break;
        case x= ‘*’ :push(OPND,pop(OPND)*pop(OPND));break;
        case x= ‘/’ :x1=pop(OPND);x2=pop(OPND);push(OPND,x2/x1);break;
        default:       //其它符号不作处理。
      }//结束 switch
      scanf(“%c”,&x);//读入表达式中下一个字符。
    }//结束 while（x! = ‘$’）
   printf(“后缀表达式的值为%f”,pop(OPND));
  }//算法结束。
```

[算法讨论]假设输入的后缀表达式是正确的，未作错误检查。算法中拼数部分是核心。若遇到大于等于‘0’且小于等于‘9’的字符，认为是数。这种字符的序号减去字符‘0’的序号得出数。对于整数，每读入一个数字字符，前面得到的部分数要乘上 10 再加新读入的数得到新的部分数。当读到小数点，认为数的整数部分已完，要接着处理小数部分。小数部分的数要除以 10（或 10 的幂数）变成十分位，百分位，千分位数等等，与前面部分数相加。在拼数过程中，若遇非数字字符，表示数已拼完，将数压入栈中，并且将变量 num 恢复为 0，准备下一个数。这时对新读入的字符进入‘+’、‘-’、‘*’、‘/’及空格的判断，因此在结束处理数字字符的 case 后，不能加入 break 语句。

5、（1）A 和 D 是合法序列，B 和 C 是非法序列。

（2）设被判定的操作序列已存入一维数组 A 中。

```
int Judge(char A[])
 //判断字符数组 A 中的输入输出序列是否是合法序列。如是，返回 true，否则返
    回 false。
 {i=0;                  //i 为下标。
  j=k=0;                //j 和 k 分别为 I 和字母 O 的的个数。
```

```
while(A[i]!='\0') //当未到字符数组尾就作。
  {switch(A[i])
    {case 'I': j++; break; //入栈次数增 1。
     case '0': k++; if(k>j){printf("序列非法\n"); exit(0);}
     }
   i++; //不论 A[i]是 'I' 或 '0', 指针 i 均后移。}
 if(j!=k) {printf("序列非法\n"); return(false);}
 else {printf("序列合法\n"); return(true);}
}//算法结束。
```

[算法讨论]在入栈出栈序列（即由 'I' 和 '0' 组成的字符串）的任一位置，入栈次数（'I' 的个数）都必须大于等于出栈次数（即 '0' 的个数），否则视作非法序列，立即给出信息，退出算法。整个序列（即读到字符数组中字符串的结束标记 '\0'），入栈次数必须等于出栈次数（题目中要求栈的初态和终态都为空），否则视为非法序列。

6、[题目分析]表达式中的括号有以下三对：'('、')'、'['、']'、'{'、'}'，使用栈，当为左括号时入栈，右括号时，若栈顶是其对应的左括号，则退栈，若不是其对应的左括号，则结论为括号不配对。当表达式结束，若栈为空，则结论表达式括号配对，否则，结论表达式括号不配对。

```
int Match(LinkedList la)
//算术表达式存储在以 la 为头结点的单循环链表中，本算法判断括号是否正确配对
 {char s[];            //s 为字符栈，容量足够大
  p=la->link;          //p 为工作指针，指向待处理结点
  StackInit(s);        //初始化栈 s
  while (p!=la)        //循环到头结点为止
   {switch (p->ch)
    {case '(' :push(s,p->ch); break;
     case ')' :if(StackEmpty(s)||StackGetTop(s)!='(')
                {printf("括号不配对\n"); return(0);} else pop(s);break;
     case '[' :push(s,p->ch); break;
     case ']' : if(StackEmpty(s)||StackGetTop(s)!='[')
                {printf("括号不配对\n"); return(0);} else pop(s);break;
     case '{' :push(s,p->ch); break;
     case '}' : if(StackEmpty(s)||StackGetTop(s)!='{')
                {printf("括号不配对\n"); return(0);} else pop(s);break;
    } p=p->link; 后移指针
    }//while
    if (StackEmpty(s)) {printf("括号配对\n"); return(1);}
    else{printf("括号不配对\n"); return(0);}
  }//算法 match 结束
```

[算法讨论]算法中对非括号的字符未加讨论。遇到右括号时，若栈空或栈顶元素不是其对应的左圆（方、花）括号，则结论括号不配对，退出运行。最后，若栈不空，仍结论括号不配对。

7、[题目分析]栈的特点是后进先出，队列的特点是先进先出。所以，用两个栈 s1 和 s2 模拟一个队列时，s1 作输入栈，逐个元素压栈，以此模拟队列元素的入队。当需要出队时，将栈 s1 退栈并逐个压入栈 s2 中，s1 中最先入栈的元素，在 s2 中处于栈顶。s2 退栈，相当

于队列的出队，实现了先进先出。显然，只有栈 s2 为空且 s1 也为空，才算是队列空。

(1) int enqueue(stack s1,elemtp x)

//s1 是容量为 n 的栈，栈中元素类型是 elemtp。本算法将 x 入栈，若入栈成功返回 1，否则返回 0。

```
{if(top1==n && !Sempty(s2))        //top1 是栈 s1 的栈顶指针，是全局变量。
  {printf("栈满");return(0);}  //s1 满 s2 非空,这时 s1 不能再入栈。
if(top1==n && Sempty(s2))          //若 s2 为空，先将 s1 退栈,元素再压栈到 s2。
  {while(!Sempty(s1)) {POP(s1,x);PUSH(s2,x);}
 PUSH(s1,x); return(1);  //x 入栈，实现了队列元素的入队。
}
```

(2) void dequeue(stack s2,s1)

```
//s2 是输出栈，本算法将 s2 栈顶元素退栈，实现队列元素的出队。
  {if(!Sempty(s2))          //栈 s2 不空，则直接出队。
    {POP(s2,x);   printf("出队元素为",x);  }
   else                     //处理 s2 空栈。
     if(Sempty(s1)) {printf("队列空");exit(0);}//若输入栈也为空，则判定队空。
     else                   //先将栈 s1 倒入 s2 中，再作出队操作。
       {while(!Sempty(s1)) {POP(s1,x);PUSH(s2,x);}
        POP(s2,x);          //s2 退栈相当队列出队。
        printf("出队元素", x);
       }
  }//结束算法 dequue。
```

(3) int queue_empty()

```
//本算法判用栈 s1 和 s2 模拟的队列是否为空。
{if(Sempty(s1)&&Sempty(s2))  return(1);//队列空。
 else return(0);                        //队列不空。
}
```

[算法讨论]算法中假定栈 s1 和栈 s2 容量相同。出队从栈 s2 出，当 s2 为空时，若 s1 不空，则将 s1 倒入 s2 再出栈。入队在 s1，当 s1 满后，若 s2 空，则将 s1 倒入 s2，之后再入队。因此队列的容量为两栈容量之和。元素从栈 s1 倒入 s2，必须在 s2 空的情况下才能进行，即在要求出队操作时，若 s2 空，则不论 s1 元素多少（只要不空），就要全部倒入 s2 中。

类似本题叙述的其它题的解答：

（1） 该题同上面题本质相同，只有叙述不同，请参考上题答案。

8、[题目分析]本题要求用链接结构实现一个队列，我们可用链表结构来实现。一般说，由于队列的先进先出性质，所以队列常设队头指针和队尾指针。但题目中仅给出一个“全局指针 p”，且要求入队和出队操作的时间复杂性是 O(1)，因此我们用只设尾指针的循环链表来实现队列。

（1） PROC addq(VAR p:linkedlist,x:elemtp);

```
//p 是数据域为 data、链域为 link 的用循环链表表示的队列的尾指针，本算法是入队操作。
new(s);       //申请新结点。假设有内存空间，否则系统给出出错信息。
s↑.data:=x; s↑.link:=p↑.link;//将 s 结点入队。
```

```
    p↑.link:=s; p:=s;                    //尾指针 p 移至新的队尾。
    ENDP;
```
（2）
```
    PROC deleq(VAR p:linkedlist,VAR x:elemtp);
    // p 是数据域为 data、链域为 link 的用循环链表表示的队列的尾指针，本算法实
      现队列元素的出队，若出队成功，返回出队元素，否则给出失败信息。
    IF （p↑.link=p）THEN[writeln(“空队列”);return(0);]//带头结点的循环队列。
      ELSE[s:=p↑.link↑.link;              //找到队头元素。
           p↑.link↑.link:=s↑.link;        //删队头元素。
           x:=s↑.data;                    //返回出队元素。
           IF (p=s) THEN p:=p↑.link;      //队列中只有一个结点，出队后成为空队
      列。
           dispose(s);                    //回收出队元素所占存储空间。
        ]
  ENDP;
```
[算法讨论]上述入队算法中，因链表结构，一般不必考虑空间溢出问题，算法简单。在出队算法中，首先要判断队列是否为空，另外，对出队元素，要判断是否因出队而成为空队列。否则，可能导致因删除出队结点而将尾指针删掉成为“悬挂变量”。

9、本题与上题本质上相同，现用类 C 语言编写入队和出队算法。

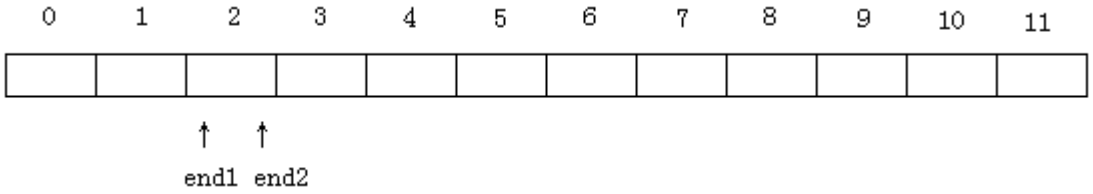
（1）
```
void EnQueue (LinkedList rear, ElemType x)
// rear 是带头结点的循环链队列的尾指针，本算法将元素 x 插入到队尾。
{ s= (LinkedList) malloc (sizeof(LNode)); //申请结点空间
  s->data=x;  s->next=rear->next;          //将 s 结点链入队尾
  rear->next=s;  rear=s;                   //rear 指向新队尾
}
```
（2）
```
void DeQueue (LinkedList  rear)
// rear 是带头结点的循环链队列的尾指针，本算法执行出队操作，操作成功输出队
  头元素；否则给出出错信息。
{ if (rear->next==rear)  { printf(“队空\n”); exit(0);}
  s=rear->next->next;                 //s 指向队头元素，
  rear->next->next=s->next;           //队头元素出队。
  printf (“出队元素是”, s->data);
  if (s==rear) rear=rear->next;   //空队列
 free(s);
}
```
10、[题目分析] 用一维数组 v[0..M-1]实现循环队列，其中 M 是队列长度。设队头指针 front 和队尾指针 rear，约定 front 指向队头元素的前一位置，rear 指向队尾元素。定义 front=rear 时为队空，(rear+1)%m=front 为队满。约定队头端入队向下标小的方向发展，队尾端入队向下标大的方向发展。

（1）
```
#define M  队列可能达到的最大长度
typedef struct
  { elemtp data[M];
    int front,rear;
  } cycqueue;
```
（2）
```
elemtp delqueue ( cycqueue Q)
```

```
//Q 是如上定义的循环队列，本算法实现从队尾删除，若删除成功，返回被删除元素，
 否则给出出错信息。
 { if (Q.front==Q.rear)  {printf(“队列空”); exit(0);}
   Q.rear=(Q.rear-1+M)%M;              //修改队尾指针。
   return(Q.data[(Q.rear+1+M)%M]); //返回出队元素。
 }//从队尾删除算法结束
void enqueue (cycqueue Q, elemtp x)
// Q 是顺序存储的循环队列，本算法实现“从队头插入”元素 x。
{if (Q.rear==(Q.front-1+M)%M)  {printf(“队满”; exit(0);)
 Q.data[Q.front]=x;          //x 入队列
 Q.front=(Q.front-1+M)%M;     //修改队头指针。
}// 结束从队头插入算法。
```

11、参见 9。

12、[题目分析] 双端队列示意图如下（设 maxsize =12）



用上述一维数组作存储结构，把它看作首尾相接的循环队列。可以在任一端（end1 或 end2）进行插入或删除。初始状态 end1+1=end2 被认为是队空状态；end1=end2 被认为是队满状态。（左端队列）end1 指向队尾元素的前一位置。end2 指向（右端队列）队尾元素的后一位置。入队时判队满，出队（删除）时判队空。删除一个元素时，首先查找该元素，然后，从队尾将该元素前的元素依次向后或向前（视 end1 端或 end2 端而异）移动。

```
FUNC add (Qu:deque; var x:datatype;tag 0..1):integer;
//在双端队列 Qu 中插入元素 x，若插入成功，返回插入元素在 Qu 中的下标；插入失败返回
-1。tag=0 表示在 end1 端插入；tag=1 表示在 end2 端插入。
  IF Qu.end1=Qu.end2 THEN [writeln(“队满”);return(-1);]
  CASE tag OF
     0: //在 end1 端插入
       [Qu.end1:=x;                          //插入 x
        Qu.end1:=(Qu.end1-1) MOD maxsize; //修改 end1
        RETURN(Qu.end1+1) MOD maxsize);   //返回插入元素的下标。
     1: //在 end2 端插入
       [Qu.end2:=x;
        Qu.end2:=(Qu.end2+1) MOD maxsize;
        RETURN(Qu.end2-1) MOD maxsize);
       ]
  ENDC; //结束 CASE 语句
ENDF; //结束算法 add
FUNC delete (Qu: deque; VAR x:datatype; tag:0..1):integer;
//本算法在双端队列 Qu 中删除元素 x，tag=0 时从 end1 端删除，tag=1 时从 end2 端删除。
删除成功返回 1，否则返回 0。
  IF (Qu.end1+1) MOD maxsize=Qu.end2 THEN [writeln(“队空”);return(0);]
```

```
  CASE tag OF
   0: //从 end1 端删除
    [i:=(Qu.end1+1) MOD maxsize; //i 是 end1 端最后插入的元素下标。
     WHILE(i<>Qu.end2) AND (Qu.elem[i]<>x) DO
            i=(i+1) MOD maxsize;//查找被删除元素 x 的位置
     IF (Qu.elem[i]=x) AND (i<>Qu.end2) THEN
       [ j:=i;
        WHILE((j-1+maxsize) MOD maxsize <>Qu.end1) DO
          [Qu.elem[j]:=Qu.elem[(j-1+maxsize) MOD maxsize];
         j:=(j-1+maxsize) MOD maxsize;
        ]//移动元素，覆盖达到删除
      Qu.end1:=(Qu.end1+1) MOD maxsize; //修改 end1 指针
      RETURN(1);
     ]
    ELSE RETURN(0);
  ]//结束从 end1 端删除。
  1: //从 end2 端删除
    [i:=(Qu.end2-1+maxsize) MOD maxsize; //i 是 end2 端最后插入的元素下标。
     WHILE(i<>Qu.end1) AND (Qu.elem[i]<>x) DO
            i=(i-1+maxsize) MOD maxsize;//查找被删除元素 x 的下标
     IF (Qu.elem[i]=x) AND (i<>Qu.end1) THEN //被删除元素找到
       [ j:=i;
        WHILE((j+1) MOD maxsize <>Qu.end2) DO
          [Qu.elem[j]:=Qu.elem[(j+1) MOD maxsize];
         j:=(j+1) MOD maxsize;
        ]//移动元素，覆盖达到删除
      Qu.end2:=(Qu.end2-1+maxsize) MOD maxsize; //修改 end2 指针
      RETURN(1);//返回删除成功的信息
     ]
    ELSE RETURN(0);//删除失败
    ]//结束在 end2 端删除。
  ENDC;//结束 CASE 语句
ENDF;//结束 delete
```

[算法讨论]请注意下标运算。(i+1) MOD maxsize 容易理解，考虑到 i-1 可能为负的情况，所以求下个 i 时用了(i-1+maxsize) MOD maxsize。

13、[题目分析] 本题与上面 12 题基本相同，现用类 C 语言给出该双端队列的定义。

```
#define maxsize  32
typedef struct
  {datatype elem[maxsize];
   int end1,end2; //end1 和 end2 取值范围是 0..maxsize-1
  } deque;
```

14、[题目分析] 根据队列先进先出和栈后进先出的性质，先将非空队列中的元素出队，并压入初始为空的栈中。这时栈顶元素是队列中最后出队的元素。然后将栈中元素出栈，依次插入到初始为空的队列中。栈中第一个退栈的元素成为队列中第一个元素，最后退栈的元素

（出队时第一个元素）成了最后入队的元素，从而实现了原队列的逆置。

```
void Invert(queue Q)
//Q 是一个非空队列，本算法利用空栈 S 和已给的几个栈和队列的 ADT 函数，将队列 Q 中的元素逆置。
  {makempty(S); //置空栈
   while (!isEmpty(Q))  // 队列 Q 中元素出队
   {value=deQueue(Q);  push(S,value);  }// 将出队元素压入栈中
  while(!isEmpty(S)) //栈中元素退栈
    {value=pop(S);   enQueue(Q,value); }//将出栈元素入队列 Q
   }//算法 invert 结束
```

15、为运算方便，设数组下标从 0 开始，即数组 v[0..m-1]。设每个循环队列长度（容量）为 L，则循环队列的个数为 n=⌈m/L⌉。为了指示每个循环队列的队头和队尾，设如下结构类型

```
typedef struct
   {int f,r;
    }scq;
scq q[n];
```

（1）初始化的核心语句

```
for(i=1;i<=n;i++) q[i].f=q[i].r=(i-1)*L;   //q[i]是全局变量
```

（2）入队 int addq(int i;elemtp x)

//n 个循环队列共享数组 v[0..m-1]和保存各循环队列首尾指针的 q[n]已经定义为全局变量，数组元素为 elemtp 类型，本过程将元素插入到第 i 个循环队列中。若入队成功，返回 1，否则返回队满标记 0（入队失败）。

```
{ if (i<1||i>n) {printf("队列号错误");exit(0);}
 if (q[i].r+1)%L+(i-1)*L==q[i].f) {printf("队满\n");exit(0);}
 q[i].r=(q[i].r+1)%L+(i-1)*L; // 计算入队位置
  v[q[i].r]=x; return(1);//元素 x 入队
}
```

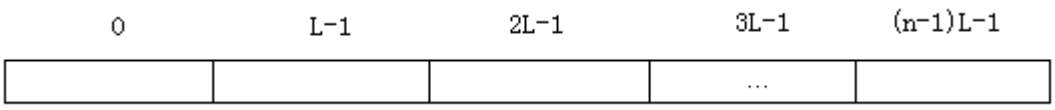
（3）出队 int deleteq (int i)

// n 个循环队列共享数组 v[0..m-1]和保存各循环队列首尾指针的 q[n]已经定义为全局变量，数组元素为 elemtp 类型，本过程将第 i 个循环队列出队。若出队成功，打印出队列元素，并返回 1 表示成功；若该循环队列为空，返回 0 表示出队失败。

```
{if (<1||>n) {printf("队列号错误\n");exit(0);}
 if (q[i].r==q[i].f) {printf("队空\n"); return(0);}
 q[i].f=(q[i].f+1)%L+(i-1)*L;
 printf("出队元素",q[i].f); return(1);
}
```

（4）讨论，上述算法假定最后一个循环队列的长度也是 L，否则要对最后一个循环队列作特殊处理。另外，未讨论一个循环队列满而相邻循环队列不满时，需修改个循环队列首尾指针的情况（即各循环队列长度不等）。

n 个循环队列共享数组 v[0..m-1]的示意图如下：



第 i 个循环队列从下标 (i-1)L 开始，到 iL-1 为止。设每个循环队列均用牺牲一个单元的办法来判断队满，即为 (q[i].r+1) %L+(i-1)*L=q[i].f 时，判定为队满。

16、
```
int MaxValue (int a[],int n)
//设整数序列存于数组 a 中，共有 n 个，本算法求解其最大值。
{if (n==1) max=a[1];
 else if a[n]>MaxValue(a,n-1) max=a[n];
      else max=MaxValue(a,n-1);
 return(max);
 }
```

17、本题与上题类似，只是这里是同时求 n 个数中的最大值和最小值的递归算法。

```
int MinMaxValue(int A[],int n,int *max,int *min)
//一维数组 A 中存放有 n 个整型数，本算法递归的求出其中的最小数。
{if (n>0)
 {if(*max<A[n]) *max=A[n];
  if(*min>A[n]) *min=A[n];
 MinMaxValue(A,n-1,max,min);
 }//算法结束
```

[算法讨论]调用本算法的格式是 MinMaxValue(arr,n,&max,&min);其中，arr 是具有 n 个整数的一维数组，max=-32768 是最大数的初值,min=32767 是最小数的初值。

18、[题目分析] 求两个正整数 m 和 n 的最大公因子，本题叙述的运算方法叫辗转相除法，也称欧几里德定理。其函数定义为：

gcd(m,n)=

```
int gcd (int m,n)
//求正整数 m 和 n 的最大公因子的递归算法
{if(m<n) return(gcd(n,m));       //若 m<n，则 m 和 n 互换
 if(n==0) return(m); else return(gcd(n,m%n));
}//算法结束
```

使用栈，消除递归的非递归算法如下：

```
int gcd(int m,n)
{int s[max][2];   //s 是栈，容量 max 足够大
 top=1; s[top][0]=m; s[top][1]=n;
 while (s[top][1]!=0)
    if (s[top][0]<s[top][1]) //若 m<n，则交换两数
       {t=s[top][0]; s[top][0]=s[top][1]; s[top][1]=t;}
    else{t=s[top][0]%s[top][1]; top++; s[top][0]=s[top-1][1]; s[top][1]=t; }
 return(s[top][0]);
}//算法结束
```

由于是尾递归，可以不使用栈，其非递归算法如下

```
int gcd (int m,n)
//求正整数 m 和 n 的最大公因子
{if (m<n){t=m;m=n;n=t;}// 若 m<n，则 m 和 n 互换
 while (n!=0) {t=m; m=n; n=t%n;}
```

```
    return(m);
  } //算法结束
```

19、[题目分析]这是以读入数据的顺序为相反顺序进行累乘问题，可将读入数据放入栈中，到输入结束，将栈中数据退出进行累乘。累乘的初值为 1。

```
PROC  test;
  CONST maxsize=32;
  VAR s:ARRAY[1..maxsize] OF integer, top,sum,a:integer;
  [top:=0; sum:=1;//
   read(a);
   WHILE a<>0 DO
      [top:=top+1; s[top]:=a; read(a); ]
   write(sum:5);
   WHILE top>0 DO
      [sum:=sum*s[top]; top:=top-1; write(sum:5);]
 ENDP;
```

20、[题目分析] 本题与第 19 题基本相同，不同之处就是求和，另外用 C 描述。

```
 int test;
  {int x,sum=0,top=0,s[];
   scanf(“%d”,&x)
   while (x<>0)
     {s[++top]:=a; scanf(“%d”,&x); }
   printf(sum:5);
   while (top)
     {sum+=s[top--]; printf(sum:5); }
   };
```

21、

```
int Ack(int m,n)
{if (m==0) return(n+1);
 else if(m!=0&&n==0) return(Ack(m-1,1));
     else return(Ack(m-1,Ack(m,m-1));
}//算法结束
```

（1）Ack(2,1)的计算过程

```
Ack(2,1)=Ack(1,Ack(2,0))              //因 m<>0,n<>0 而得
        =Ack(1,Ack(1,1))              //因 m<>0,n=0 而得
        =Ack(1,Ack(0,Ack(1,0)))       // 因 m<>0,n<>0 而得
        = Ack(1,Ack(0,Ack(0,1)))      // 因 m<>0,n=0 而得
        =Ack(1,Ack(0,2))              // 因 m=0 而得
        =Ack(1,3)                     // 因 m=0 而得
        =Ack(0,Ack(1,2))              //因 m<>0,n<>0 而得
        = Ack(0,Ack(0,Ack(1,1)))      //因 m<>0,n<>0 而得
        = Ack(0,Ack(0,Ack(0,Ack(1,0)))) //因 m<>0,n<>0 而得
        = Ack(0,Ack(0,Ack(0,Ack(0,1)))) //因 m<>0,n=0 而得
        = Ack(0,Ack(0,Ack(0,2)))      //因 m=0 而得
        = Ack(0,Ack(0,3))             //因 m=0 而得
        = Ack(0,4)                    //因 n=0 而得
```

=5 //因 n=0 而得

（2）
```
int Ackerman( int m, int n)
{int akm[M][N];int i,j;
 for(j=0;j<N;j++) akm[0][j];=j+1;
 for(i=1;i<m;i++)
   {akm[i][0]=akm[i-1][1];
    for(j=1;j<N;j++)
      akm[i][j]=akm[i-1][akm[i][j-1]];
    }
   return(akm[m][n]);
 }//算法结束
```

22、[题目分析]从集合（1..n）中选出 k(本题中 k=2)个元素，为了避免重复和漏选，可分别求出包括 1 和不包括 1 的所有组合。即包括 1 时，求出集合（2..n）中取出 k-1 个元素的所有组合；不包括 1 时，求出集合（2..n）中取出 k 个元素的所有组合。，将这两种情况合到一起，就是题目的解。

```
int A[],n; //设集合已存于数组 A 中。
 void comb(int P[],int i,int k)
  //从集合（1..n）中选取 k（k<=n）个元素的所有组合
 {if (k==0) printf(P);
  else if(k<=n) {P[i]=A[i]; comb(P,i+1,k-1); comb(P,i+1,k); }
 }//算法结束
```

第四章　　串

一、选择题

1．下面关于串的的叙述中，哪一个是不正确的？（　　）【北方交通大学 2001 一、5（2分）】

A．串是字符的有限序列　　B．空串是由空格构成的串

C．模式匹配是串的一种重要运算　D．串既可以采用顺序存储，也可以采用链式存储

2　若串 S_1=‘ABCDEFG’，S2=‘9898’，S3=‘###’，S4=‘012345’，执行

concat(replace(S1,substr(S1,length(S2),length(S3)),S3),substr(S4,index(S2,‘8’),length(S2)))

其结果为（　　）【北方交通大学　1999　一、5　（25/7 分）】

A．ABC###G0123　B．ABCD###2345　C．ABC###G2345　D．ABC###2345

E．ABC###G1234　F．ABCD###1234　G．ABC###01234

3．设有两个串 p 和 q，其中 q 是 p 的子串，求 q 在 p 中首次出现的位置的算法称为（　　）

A．求子串　　B．联接　　C．匹配　　D．求串长

【北京邮电大学 2000 二、4（20/8 分）】【西安电子科技大学 1996 一、1 （2 分）】

4．已知串 S=‘aaab’，其 Next 数组值为（　　）。【西安电子科技大学 1996 一、7 （2 分）】

A．0123　　B．1123　　C．1231　　D．1211

5．串 ‘ababaaababaa’ 的 next 数组为（　　）。【中山大学 1999 一、7】

A．012345678999　B．012121111212　C．011234223456　D．0123012322345

6．字符串‘ababaabab’ 的 nextval 为（　　）

A．(0,1,0,1,04,1,0,1)　　B．(0,1,0,1,0,2,1,0,1)

C．(0,1,0,1,0,0,0,1,1)　　D．(0,1,0,1,0,1,0,1,1)

【北京邮电大学 1999　一、1（2 分）】

7．模式串 t=‘abcaabbcabcaabdab’，该模式串的 next 数组的值为（　），nextval 数组的值为　（ ）。

A．0 1 1 1 2 2 1 1 1 2 3 4 5 6 7 1 2　　B．0 1 1 1 2 1 2 1 1 2 3 4 5 6 1 1 2

C．0 1 1 1 0 0 1 3 1 0 1 1 0 0 7 0 1　　D．0 1 1 1 2 2 3 1 1 2 3 4 5 6 7 1 2

E．0 1 1 0 0 1 1 1 0 1 1 0 0 1 7 0 1　　F．0 1 1 0 2 1 3 1 0 1 1 0 2 1 7 0 1

【北京邮电大学 1998 二、3 （2 分）】

8．若串 S=’software’，其子串的数目是（　　）。【西安电子科技大学 2001 应用 一、2（2 分）】

A．8　　B．37　　C．36　　D．9

9．设 S 为一个长度为 n 的字符串，其中的字符各不相同，则 S 中的互异的非平凡子串（非空且不同于 S 本身）的个数为（　　）。【中科院计算所 1997 】

A．2n-1　　B．n2　　C．(n2/2)+(n/2)　　D．(n2/2)+(n/2)-1　　E．(n2/2)-(n/2)-1

F.其他情况

10．串的长度是指（　　）【北京工商大学 2001　一、6 （3 分）】

A．串中所含不同字母的个数　　B．串中所含字符的个数

C．串中所含不同字符的个数　　D．串中所含非空格字符的个数

二、判断题

1．KMP 算法的特点是在模式匹配时指示主串的指针不会变小。（　　）【北京邮电大学 2002 一、4 （1 分）】

2．设模式串的长度为 m,目标串的长度为 n，当 n≈m 且处理只匹配一次的模式时，朴素的匹

配(即子串定位函数)算法所花的时间代价可能会更为节省。(　　)【长沙铁道学院 1998 一、1 (1 分)】

3. 串是一种数据对象和操作都特殊的线性表。(　　)【大连海事大学 2001 1、L (1 分)】

二、填空题

1. 空格串是指__(1)__，其长度等于__(2)__。【西安电子科技大学 2001 软件 一、4（2 分)】

2. 组成串的数据元素只能是_____。【中山大学 1998 一、5 （1 分)】

3. 一个字符串中_____称为该串的子串 。【华中理工大学 2000 一、3（1 分)】

4. INDEX（'DATASTRUCTURE'， 'STR'）=_____。【福州大学 1998 二、4 (2 分)】

5. 设正文串长度为 n，模式串长度为 m，则串匹配的 KMP 算法的时间复杂度为_____。【重庆大学 2000 一、4】

6.模式串 P='abaabcac'的 next 函数值序列为_____。【西安电子科技大学 2001 软件 一、6（2 分)】

7. 字符串' ababaaab' 的 nextval 函数值为_____。【北京邮电大学 2001 二、4 （2 分)】

8.设 T 和 P 是两个给定的串,在 T 中寻找等于 P 的子串的过程称为__(1)__,又称 P 为__(2)__。【西安电子科技大学 1998 二、5 （16/6 分)】

9.串是一种特殊的线性表,其特殊性表现在__(1)__;串的两种最基本的存储方式是__(2)__、__(3)__；两个串相等的充分必要条件是__(4)__。【中国矿业大学 2000 一、3 （4 分)】

10. 两个字符串相等的充分必要条件是_____。【西安电子科技大学 1999 软件 一、1 （2 分)】

11. 知 U= 'xyxyxyxxyxy'; t= 'xxy';

```
ASSIGN（S, U);
ASSIGN（V, SUBSTR（S, INDEX（s, t), LEN（t）+1));
ASSIGN（m, 'ww'）
```

求 REPLACE（S, V, m）= _____。【东北大学 1997 一、1 (5 分)】

12. 实现字符串拷贝的函数 strcpy 为：

```
 void strcpy(char *s , char *t) /*copy t to s*/
{ while  (_____)
}
```

【浙江大学 1999 一、5 (3 分)】

13. 下列程序判断字符串 s 是否对称，对称则返回 1，否则返回 0；如 f("abba")返回 1，f("abab")返回 0；

```
int f((1)_____)
    {int   i=0,j=0;
     while (s[j])(2)_____;
     for(j--; i<j  && s[i]==s[j]; i++,j--);
     return((3)_____)
    }
```

【浙江大学 1999 一、6 (3 分)】

14. 下列算法实现求采用顺序结构存储的串 s 和串 t 的一个最长公共子串。

程序（a）

```
PROCEDURE  maxcomstr(VAR s,t : orderstring; VAR index,length : integer);
VAR i,j,k,length1:integer;  con:boolean;
BEGIN
```

```
  index :=0; length :=0;  i :=1;
  WHILE(i<=s.len) DO
    [j:=1;
     WHILE (j<=t.len) DO
        [ IF (s[i]=t[j])  THEN
            [ k:=1;  length1:=1;  con:=true;
              WHILE  con  DO
                IF (1)__THEN [length1:=length1+1;k:=k+1;] ELSE(2) _;
              IF (length1>length) THEN [index:=i; length:=length1; ]
              (3)    ;
             ]
           ELSE (4)___;
         ]
      (5) ___;
    ]
END;
```

程序(b)

```
void  maxcomstr(orderstring *s,*t; int index, length)
 {int i,j,k,length1,con;
  index=0;length=0;i=1;
  while (i<=s.len)
   {j=1;
    while(j<=t.len)
     { if (s[i]= =t[j])
        { k=1;length1=1;con=1;
          while(con)
             if (1) _ { length1=length1+1;k=k+1; }  else (2) __;
          if (length1>length) { index=i;  length=length1; }
          (3)    ;
         }
       else (4)    ;
      }
     (5) __
   } }
```

【上海大学 2000 一、2 （10 分)】

15. 完善算法：求 KMP 算法中 next 数组。

```
  PROC get _next(t:string,VAR next:ARRAY[1..t.len] OF integer);
  BEGIN
   j:=1; k:=(1)__;  next[1]:=0;
   WHILE j<t.len DO
   IF k=0 OR t.ch[j]=t.ch[k] THEN BEGIN j:=j+1; k:=k+1; next[j]:=k;END
                             ELSE k:=(2)___;
  END;
```

【中山大学 1998 四、1 （4 分)】

16. 下面函数 index 用于求 t 是否为 s 的子串，若是返回 t 第一次出现在 s 中的序号(从 1

开始计)，否则返回0。

例如:s=‘abcdefcdek’，t=‘cde’,则indse(s,t)=3, index(s,’aaa’)=0 。已知t，s的串长分别是mt,ms

```
FUNC index(s,t,ms,mt);
  i:=1;j:=1;
  WHILE  (i<ms) AND (j<mt) DO
    IF s[i]=t[j] THEN  [ (1)__; (2)__]
                 ELSE [ (3)___; (4)_ ]
  IF j>mt THEN  return (5)____; ELSE  return (6)__
ENDF;
```

【南京理工大学 1999 三、2 （6分)】

17. 阅读下列程序说明和pascal程序,把应填入其中的（ ）处的字句写在答题纸上。

程序说明：

本程序用于判别输入的字符串是否为如下形式的字符串:

W&M$ 其中,子字符串M是子字符串W的字符反向排列,在此假定W不含有字符&和字符$,字符&用作W与M的分隔符,字符$用作字符串的输入结束符。

例如,对输入字符串ab&ba$、11&12$、ab&dd$、&$,程序将分别输出Ok.(是),No.(不是)。

程序

```
PROGRAM  accept(input,output);
CONST  midch=’&’;   endch=’$’;
VAR   an:boolean;    ch:char;
PROCEDURE  match(VAR  answer: boolean);
   VAR  ch1,ch2:char;   f:boolean;
BEGIN
  read(ch1);
  IF  ch1<>endch
     THEN IF  (1)__
          THEN BEGIN match(f);
                 IF  f  THEN  BEGIN  read(ch2);  answer:=(2)_  END  ELSE
answer:=false
               END
           ELSE (3)___
      ELSE (4)___
END;
BEGIN
   writeln(‘Enter  String:’);
   match(an);
   IF  an  THEN BEGIN
                (5)__  IF  (6)_  THEN    writeln(  ‘ Ok. ’  )  ELSE
writeln(‘No.’)
                END
   ELSE   writeln(‘No.’)
END.
```

【上海海运学院 1998 七 （15分)】

18. 试利用下列栈和串的基本操作完成下述填空题。

```
initstack(s)          置 s 为空栈;
push(s,x)             元素 x 入栈;
pop(s)                出栈操作;
gettop(s)             返回栈顶元素;
sempty(s)             判栈空函数;
setnull(st)           置串 st 为空串;
length(st)            返回串 st 的长度;
equal(s1,s2)          判串 s1 和 s2 是否相等的函数;
concat(s1,s2)         返回联接 s1 和 s2 之后的串;
sub(s,i,1)            返回 s 中第 i 个字符;
empty(st)             判串空函数
```

```
FUNC   invert(pre:string; VAR  exp:string):boolean;
```

{若给定的表达式的前缀式pre正确，本过程求得和它相应的表达式exp并返回“true”，否则exp为空串，并返回“false”。已知原表达式中不包含括弧，opset为运算符的集合。}

```
VAR  s:stack;   i,n:integer;   succ:boolean;   ch: char;
BEGIN
  i:=1;  n:=length(pre);   succ:=true;
  (1)__;  (2)__;
  WHILE  (i<n)  AND  succ  DO
    BEGIN ch:=sub (pre,i,1) ;
          IF (3)__THEN (4)__
          ELSE IF (5)__THEN (6)__
               ELSE  BEGIN
                       exp:=concat((7)___,(8)____);
                       exp:=concat((9)____,(10)___);
                        (11)__;
                     END;
        i:=i+1
    END;
  IF (12)___THEN
     BEGIN exp:=concat(exp,sub(pre,n,1)); invert:=true END
  ELSE  BEGIN setnull(exp); invert:=false  END
END;
```

注意：每个空格只填一个语句。 【清华大学 1996 八】

四、应用题

1. 名词解释：串 【大连海事 1996 一、10 (1分)】【河海大学 1998 二、5（3分)】

2. 描述以下概念的区别：空格串与空串。【大连海事大学 1996 三、2、(1) (2分)】

3. 两个字符串S1和S2的长度分别为m和n。求这两个字符串最大共同子串算法的时间复杂度为T(m,n)。估算最优的T(m,n)，并简要说明理由。 【北京工业大学 1996 一、5 (6分)】

4. 设主串S=‘xxyxxxyxxxxyxyx’，模式串T=‘xxyxy’。请问：如何用最少的比较次数找到T在S中出现的位置？相应的比较次数是多少？ 【大连海事大学 2001 四 (8分)】

5．KMP 算法(字符串匹配算法)较 Brute(朴素的字符串匹配)算法有哪些改进?【大连海事大学 1996 三、1((2 分)】

6．已知模式串 t=‘abcaabbabcab’写出用 KMP 法求得的每个字符对应的 next 和 nextval 函数值。

【北京邮电大学 1997 三 （10 分)】

7．给出字符串‘abacabaaad’在 KMP 算法中的 next 和 nextval 数组。【北京邮电大学 2000 三、1（5 分)】

8．令 t=‘abcabaa’,求其 next 函数值和 nextval 函数值。【北方交通大学 1994 一 （6 分)】

9．已知字符串‘cddcdececdea’，计算每个字符的 next 和 nextval 函数的值。【南京邮电大学 2000 一 2】

10．试利用 KMP 算法和改进算法分别求 p1=‘abaabaa’和 p2=‘aabbaab’的 next 函数和 nextval 函数。

【东南大学 1999 一、6（8 分)】

11.已知 KMP 串匹配算法中子串为 bababababa,写出 next 数组改进后的 next 数组信息值(要求写出数组下标起点)。【西南交通大学 2000 二、2】

12．求模式串 T=‘abcaabbac’ 的失败函数 Next(j)值。【西安交通大学 1996 四、4 (5 分)】

13．字符串的模式匹配 KMP 算法中,失败函数(NEXT)是如何定义的?计算模式串 p=‘aabaabaaabc’中各字符的失败函数值.【石油大学 1998 一、2 (10 分)】

14．设字符串 S=‘aabaabaabaac’，P=‘aabaac’

（1）给出 S 和 P 的 next 值和 nextval 值;

（2）若 S 作主串，P 作模式串，试给出利用 BF 算法和 KMP 算法的匹配过程。

【北方交通大学 1998 二（15 分)】

15．设目标为 t=‘abcaabbabcabaacbacba’,模式为 p=‘abcabaa’

（1）计算模式 p 的 naxtval 函数值;（5 分）

（2）不写出算法,只画出利用 KMP 算法进行模式匹配时每一趟的匹配过程。(5 分）

【清华大学 1998 八（10 分)】

16．模式匹配算法是在主串中快速寻找模式的一种有效的方法，如果设主串的长度为 m，模式的长度为 n，则在主串中寻找模式的 KMP 算法的时间复杂性是多少？如果，某一模式 P=’abcaacabaca’，请给出它的 NEXT 函数值及 NEXT 函数的修正值 NEXTVAL 之值。【上海交通大学 2000 一 （5 分)】

17．设目标为 S=‘abcaabbcaaababababaabca’，模式为 P=‘babab’，

（1）手工计算模式 P 的 nextval 数组的值；(5 分)

（2）写出利用求得的 nextval 数组，按 KMP 算法对目标 S 进行模式匹配的过程。 (5 分)

【清华大学 1997 四（10 分)】

18．用无回溯的模式匹配法（KMP 法）及快速的无回溯的模式匹配法求模式串 T 的 next[j]值，添入下面表中：

j	1	2	3	4	5	6	7
t	a	a	b	b	a	a	b
kmp法求得的next[j]值							
快速无回溯法求得的next[j]值							

【北京邮电大学 1992 三、1（25/4 分）】

19. 在改进了的（无回溯）字符串模式匹配中，要先求 next 数组的值。下面是求 nextval 值的算法。

```
TYPE SAR=ARRAY[1..m] OF INTEGER;
     PTY=ARRAY[1..m] OF CHAR;
PROCEDURE next2(P:PTY;VAR NEXTVAL:SAR);
{在模式 P 中求 nextval 数组的值}
1   BEGIN
2   J:=1;NEXTVAL[1]:=0;K:=0
3   REPEAT
4     IF (K=0) OR (P[J]=P[K])
5       THEN [ J:=J+1;K:=K+1;
6              IF P[J]=P[K]
7                THEN NEXTVAL[J]:=NEXTVAL[K]
8                ELSE NEXTVAL[J]:=K ]
9       ELSE K:=NEXTVAL[K]
10  UNTIL J=m
11  END;
```

算法中第 4 行有 P[J]=P[K],第六行中也有 P[J]=P[K]。两处比较语句相同。请分析说明此两处比较语句的含义是什么？分析此算法在最坏情况下的时间复杂度是多少？【北京邮电大学 1993 二、2（6 分）】

20. 在字符串模式匹配的 KMP 算法中，求模式的 next 数组值的定义如下：

$$next[j]=\begin{cases}0 & \text{当 } j=1 \text{ 时}\\ \max\{k \mid 1<k<j \text{ 且 } `p_1\cdots p_{k-1}`=`p_{j-k+1}\cdots p_{j-1}`\} & \\ 1 & \text{其它情况}\end{cases}$$

请问：

（1）当 j=1 时，为什么要取 next[1]=0？

（2）为什么要取 max{K},K 最大是多少？

（3）其它情况是什么情况，为什么取 next[j]=1？【北京邮电大学 1994 二（8 分）】

21. 给出 KMP 算法中失败函数 f 的定义，并说明利用 f 进行串模式匹配的规则，该算法的技术特点是什么？

【东南大学 1993 一、3 （9 分） 1997 一、2 （8 分） 2001 一、6 （6 分）】

22. 在模试匹配 KMP 算法中所用失败函数 f 的定义中，为何要求 $p_1p_2\cdots\cdots p_{f(j)}$为 $p_1p_2\cdots\cdots p_j$ 两头匹配的真子串？且为最大真子串？ 【东南大学 1996 一、3（7 分）】

23. 如果两个串含有相等的字符，能否说它们相等？【西安电子科技大学 2000 软件 一、3（5 分）】

24. 设 S1,S2 为串，请给出使 S1//S2=S2//S1 成立的所有可能的条件（//为连接符）。

【长沙铁道学院 1997 三、5 (3 分)】【国防科技大学 1999 一 】

25. 已知：s ＝’（xyz）＋＊’，t ＝’（x＋z）＊y’。试利用联结、求子串和置换等基本运算，将 s 转化为 t 。

【北方交通大学 1996 一、3（5 分）】【山东科技大学 2002 一、6 （5 分）】

第五部分、算法设计

1．设 s、t 为两个字符串，分别放在两个一维数组中，m、n 分别为其长度，判断 t 是否为 s 的子串。如果是，输出子串所在位置（第一个字符），否则输出 0。（注：用程序实现）【南京航空航天大学 1997 九(10 分)】

2．输入一个字符串，内有数字和非数字字符，如：ak123x456 17960?302gef4563，将其中连续的数字作为一个整体，依次存放到一数组 a 中，例如 123 放入 a［0］,456 放入 a［1］,… … 。编程统计其共有多少个整数，并输出这些数。【上海大学 1998 一 （13 分）】

3． 以顺序存储结构表示串，设计算法。求串 S 中出现的第一个最长重复子串及其位置并分析算法的时间复杂度。【东南大学 2000 五 （15 分）】

类似本题的另外叙述有：

（1）如果字符串的一个子串（其长度大于 1）的各个字符均相同，则称之为等值子串。试设计一算法，输入字符串 S，以“!”作为结束标志。如果串 S 中不存在等值子串，则输出信息“无等值子串”，否则求出（输出）一个长度最大的等值子串。

例如：若 S=“abc123abc123!”，则输出“无等值子串”；若 S=“abceebccaddddddaaadd!”，则输出“ddddd”。

【华中科技大学 2001】

4．假设串的存储结构如下所示，编写算法实现串的置换操作。【清华大学 1995 五（15 分）】

```
TYPE strtp =RECORD
              ch: ARRAY[1..maxlen]  OF  char;
              curlen:0..maxlen
            END;
```

5．函数 void insert(char*s,char*t,int pos)将字符串 t 插入到字符串 s 中，插入位置为 pos。请用 c 语言实现该函数。假设分配给字符串 s 的空间足够让字符串 t 插入。（说明：不得使用任何库函数）

【北京航空航天大学 2001 六 （10 分）】

6．设计一个二分检索的算法,在一组字符串中找出给定的字符串,假设所有字符串的长度为 4。

（1）简述算法的主要思想；(3 分)

（2）用 PASCAL 语言分别对算法中用到的类型和变量作出说明；(3 分)

（3）用类 PASCAL 语言或自然语言写算法的非递归过程； (8 分)

（4）分析该算法的最大检索长度；(3 分)

（5）必要处加上中文注释。(3 分)

【山东工业大学 1995 八 （20 分）】

7．设计一 PASCAL 或 C 语言的函数 atoi(x).其中 X 为字符串，由 0--9 十个数字符和表示正负数的‘-’组成，返回值为整型数值 。【浙江大学 1994 二 (7 分)】

8．已知字符串 S1 中存放一段英文，写出算法 format(s1,s2,s3,n),将其按给定的长度 n 格式化成两端对齐的字符串 S2, 其多余的字符送 S3。 【首都经贸大学 1998 三、8（15 分）】

9．串以静态存储结构存储，结构如下所述，试实现串操作 equal 算法.

```
CONST  maxlen=串被确认的最大长度
TYPE  strtp=RECORD
            ch:ARRAY[1..maxlen] OF char;
            curlen:0..maxlen
          END;
```

(以一维数组存放串值，并设指示器 curlen 指示当前串长)【北京轻工业大学 1998 一

（12 分）】

10．编写程序，统计在输入字符串中各个不同字符出现的频度并将结果存入文件（字符串中的合法字符为 A-Z 这 26 个字母和 0-9 这 10 个数字）。【西北大学 2000 四 (10 分)】

11．写一个递归算法来实现字符串逆序存储，要求不另设串存储空间。【西南交通大学 2000 三、2】

12．已知三个字符串分别为 s=’ ab…abcaabcbca…a’，s’ =’ caab’， s’’ =’ bcb’。利用所学字符串基本运算的函数得到结果串为：s’’’ =’ caabcbca…aca…a’，要求写出得到上结果串 S’’’ 所用的函数及执行算法。【东北大学 1998 一、1 (10 分)】

13．S=“S1S2…Sn”是一个长为 N 的字符串，存放在一个数组中，编程序将 S 改造之后输出：

（1）将 S 的所有第偶数个字符按照其原来的下标从大到小的次序放在 S 的后半部分；

（2）将 S 的所有第奇数个字符按照其原来的下标从小到大的次序放在 S 的前半部分；

例如：

S= ‘ABCDEFGHIJKL’

则改造后的 S 为 ‘ACEGIKLJHFDB’。【中科院计算所 1995】

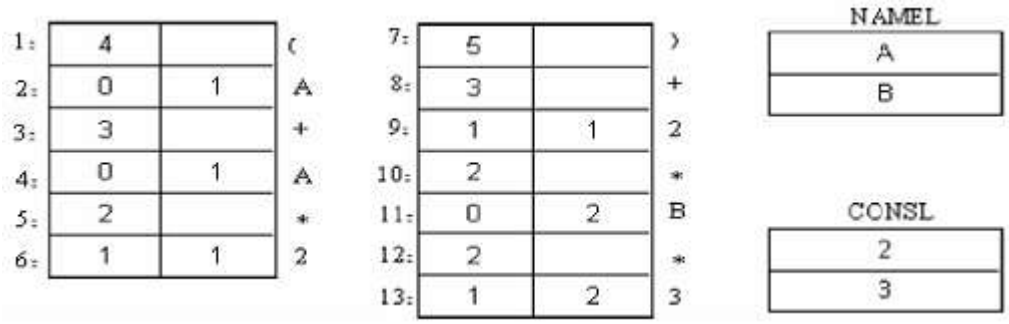
14．编一程序，对输入的一表达式（字符串），输出其 TOKEN 表示。表达式由变量 A，B，C，常数（数字）0，1，…，9，运算符+，*和括号“(”，“)”组成。首先定义符号的类码：

符号	变量	常量	*	+	(	)
类码	0	1	2	3	4	5

其次定义符号的 TOKEN 表示：

其中 NAMEL 是变量名表（不允许有相同名），CONST 是常量表（不允许有相同数）。

例如，假设有表达式（A+A*2）+2*B*3#，则将生成如下 TOKENL：



【吉林大学 1995 一 (20 分)】

第四章 串 （答案）

一、选择题

1.B	2.E	3.C	4.A	5.C	6.A	7.1D	7.2F	8.B注	9.D	10.B

注：子串的定义是：串中任意个连续的字符组成的子序列，并规定空串是任意串的子串，任意串是其自身的子串。若字符串长度为 n（n>0），长为 n 的子串有 1 个，长为 n-1 的子串有 2 个，长为 n-2 的子串有 3 个，……，长为 1 的子串有 n 个。由于空串是任何串的子串，所以本题的答案为：8*（8+1）/2+1=37。故选 B。但某些教科书上认为“空串是任意串的子串”无意义，所以认为选 C。为避免考试中的二意性，编者认为第 9 题出得好。

二、判断题

1. √	2. √	3. √

三．填空题

1．(1) 由空格字符（ASCII 值 32）所组成的字符串　(2)空格个数　2．字符

3．任意个连续的字符组成的子序列　4．5　5.0(m+n)

6．01122312　7．01010421　8．(1)模式匹配　(2)模式串

9．(1)其数据元素都是字符(2)顺序存储(3)和链式存储(4)串的长度相等且两串中对应位置的字符也相等

10．两串的长度相等且两串中对应位置的字符也相等。

11.’xyxyxywwy’　12．*s++=*t++ 或（*s++=*t++）!=‘\0’

13.（1）**char** s[ ]　(2) j++　(3) i >= j

14．[题目分析]本题算法采用顺序存储结构求串 s 和串 t 的最大公共子串。串 s 用 i 指针（1<=i<=s.len)。t 串用 j 指针（1<=j<=t.len)。算法思想是对每个 i（1<=i<=s.len，即程序中第一个 **WHILE** 循环），来求从 i 开始的连续字符串与从 j（1<=j<=t.len，即程序中第二个WHILE 循环）开始的连续字符串的最大匹配。程序中第三个（即最内层）的 **WHILE** 循环，是当 s 中某字符（s [i]）与 t 中某字符（t [j]）相等时，求出局部公共子串。若该子串长度大于已求出的最长公共子串（初始为 0 ），则最长公共子串的长度要修改。

程序（a)：(1)（i+k<=s.len）AND(j+k<=t.len) AND(s[i+k]=t[j+k])
　　//如果在 s 和 t 的长度内，对应字符相等，则指针 k 后移（加 1)。
　　(2) con:=false　//s 和 t 对应字符不等时置标记退出
　　(3) j:=j+k　//在 t 串中，从第 j+k 字符再与 s[i]比较
　　(4) j:=j+1　//t 串取下一字符
　　(5) i：=i+1　//s 串指针 i 后移（加 1)。

程序（b)：(1) i+k<=s.len && j+k<=t.len && s[i+k]==t[j+k]　//所有注释同上（a)
　　(2) con=0　(3) j+=k　(4) j++　(5) i++

15.（1）0　（2）next[k]

16.(1)i：=i+1　(2)j:=j+1　(3)i:=i-j+2　(4)j:=1;　(5)i-mt（或 i:=i-j+1）　(6)0

17.程序中递归调用

（1）ch1<>midch　//当读入不是分隔符&和输入结束符$时，继续读入字符

（2）ch1=ch2　//读入分隔符&后，判 ch1 是否等于 ch2，得出真假结论。

（3）answer：=true

（4）answer：=false

（5）read（ch）

（6）ch=endch

18.（1）initstack（s）　//栈 s 初始化为空栈。

(2) setnull (exp)　　//串 exp 初始化为空串。

(3) ch in opset　　//判取出字符是否是操作符。

(4) push (s,ch)　　//如 ch 是运算符，则入运算符栈 s。

(5) sempty (s)　　//判栈 s 是否为空。

(6) succ := false　　//若读出 ch 是操作数且栈为空，则按出错处理。

(7) exp　　(8)ch　//若 ch 是操作数且栈非空，则形成部分中缀表达式。

(9) exp　　(10) gettop(s) //取栈顶操作符。

(11) pop(s)　　//操作符取出后，退栈。

(12) sempty(s)　　//将 pre 的最后一个字符（操作数）加入到中缀式 exp 的最后。

四．应用题

1．串是零个至多个字符组成的有限序列。从数据结构角度讲，串属于线性结构。与线性表的特殊性在于串的元素是字符。

2．空格是一个字符，其 ASCII 码值是 32。空格串是由空格组成的串，其长度等于空格的个数。空串是不含任何字符的串，即空串的长度是零。

3．最优的 T(m,n)是 O（n）。串 S2 是串 S1 的子串，且在 S1 中的位置是 1。开始求出最大公共子串的长度恰是串 S2 的长度，一般情况下，T(m,n) =O(m*n)。

4．朴素的模式匹配（Brute－Force）时间复杂度是 O（m＊n），KMP 算法有一定改进，时间复杂度达到 O（m＋n）。本题也可采用从后面匹配的方法，即从右向左扫描，比较 6 次成功。另一种匹配方式是从左往右扫描，但是先比较模式串的最后一个字符，若不等，则模式串后移；若相等，再比较模式串的第一个字符，若第一个字符也相等，则从模式串的第二个字符开始，向右比较，直至相等或失败。若失败，模式串后移，再重复以上过程。按这种方法，本题比较 18 次成功。

5．KMP 算法主要优点是主串指针不回溯。当主串很大不能一次读入内存且经常发生部分匹配时，KMP 算法的优点更为突出.

6．模式串的 next 函数定义如下：

next [j]

=

根据此定义，可求解模式串 t 的 next 和 nextval 值如下：

j	1 2 3 4 5 6 7 8 9 10 11 12
t串	a b c a a b b a b c a b
next[j]	0 1 1 1 2 2 3 1 2 3 4 5
nextval[j]	0 1 1 0 2 1 3 0 1 1 0 5

7．解法同上题 6，其 next 和 nextval 值分别为 0112123422 和 0102010422。
8．解法同题 6，t 串的 next 和 nextval 函数值分别为 0111232 和 0110132。
9．解法同题 6，其 next 和 nextval 值分别为 011123121231 和 011013020131。
10．p1 的 next 和 nextval 值分别为：0112234 和 0102102；p2 的 next 和 nextval 值分别为：0121123 和 0021002。
11．next 数组值为 011234567　改进后的 next 数组信息值为 010101017。
12．011122312。
13．next 定义见题上面 6 和下面题 20。串 p 的 next 函数值为：01212345634。
14.(1)S 的 next 与 nextval 值分别为 012123456789 和 002002002009,p 的 next 与 nextval 值分别为 012123 和 002003。

（2）利用 BF 算法的匹配过程：

第一趟匹配：aabaabaabaac
　　aabaac(i=6,j=6)
第二趟匹配：aabaabaabaac
　　aa(i=3,j=2)
第三趟匹配：aabaabaabaac
　　a(i=3,j=1)
第四趟匹配：aabaabaabaac
　　aabaac(i=9,j=6)
第五趟匹配：aabaabaabaac
　　aa(i=6,j=2)
第六趟匹配：aabaabaabaac
　　a(i=6,j=1)
第七趟匹配：aabaabaabaac
（成功）　　aabaac(i=13,j=7)

利用 KMP 算法的匹配过程：

第一趟匹配：aabaabaabaac
　　aabaac(i=6,j=6)
第二趟匹配：aabaabaabaac
　　(aa)baac
第三趟匹配：aabaabaabaac
　　（成功）(aa)baac

15．（1）p 的 nextval 函数值为 0110132。（p 的 next 函数值为 0111232）。
（2）利用 KMP(改进的 nextval)算法，每趟匹配过程如下：

第一趟匹配：abcaabbabcabaacbacba
　　abcab(i=5,j=5)
第二趟匹配：abcaabbabcabaacbacba
　　abc(i=7,j=3)
第三趟匹配：abcaabbabcabaacbacba
　　a(i=7,j=1)
第四趟匹配：abcaabbabcabaac bacba
（成功）　　abcabaa(i=15,j=8)

16．KMP 算法的时间复杂性是 O（m+n）。
p 的 next 和 nextval 值分别为 01112212321 和 01102201320。
17．（1）p 的 nextval 函数值为 01010。（next 函数值为 01123）
（2）利用所得 nextval 数值，手工模拟对 s 的匹配过程，与上面 16 题类似，为节省篇幅，故略去。
18．模式串 T 的 next 和 nextval 值分别为 0121123 和 0021002。
19．第 4 行的 p[J]=p[K]语句是测试模式串的第 J 个字符是否等于第 K 个字符，如是，则指针 J 和 K 均增加 1，继续比较。第 6 行的 p[J]=p[K]语句的意义是，当第 J 个字符在模式匹配中失配时，若第 K 个字符和第 J 个字符不等，则下个与主串匹配的字符是第 K 个字符；否

则，若第 K 个字符和第 J 个字符相等，则下个与主串匹配的字符是第 K 个字符失配时的下一个（即 NEXTVAL[K]）。

该算法在最坏情况下的时间复杂度 O（m^2）。

20.（1）当模式串中第一个字符与主串中某字符比较不等（失配）时，next[1]=0 表示模式串中已没有字符可与主串中当前字符 s[i]比较，主串当前指针应后移至下一字符，再和模式串中第一字符进行比较。

（2）当主串第 i 个字符与模式串中第 j 个字符失配时，若主串 i 不回溯，则假定模式串第 k 个字符与主串第 i 个字符比较，k 值应满足条件 1<k<j 并且‘$p_1 \cdots p_{k-1}$’=‘$p_{j-k+1} \cdots p_{j-1}$’，即 k 为模式串向后移动的距离，k 值有多个，为了不使向右移动丢失可能的匹配，k 要取大，由于 max{k}表示移动的最大距离，所以取 max{k}，k 的最大值为 j-1。

（3）在上面两种情况外，发生失配时，主串指针 i 不回溯，在最坏情况下，模式串从第 1 个字符开始与主串第 i 个字符比较，以便不致丢失可能的匹配。

21. 这里失败函数 f，即是通常讲的模式串的 next 函数，其定义见本章应用题的第 6 题。

进行模式匹配时，若主串第 i 个字符与模式串第 j 个字符发生失配，主串指针 i 不回溯，和主串第 i 个字符进行比较的是模式串的第 next[j]个字符。模式串的 next 函数值，只依赖于模式串，和主串无关，可以预先求出。

该算法的技术特点是主串指针 i 不回溯。在经常发生“部分匹配”和主串很大不能一次调入内存时，优点特别突出。

22. 失败函数（即 next）的值只取决于模式串自身，若第 j 个字符与主串第 i 个字符失配时，假定主串不回溯，模式串用第 k（即 next[j]）个字符与第 i 个相比，有‘ $p_1 \cdots p_{k-1}$’=‘$p_{j-k+1} \cdots p_{j-1}$’，为了不因模式串右移与主串第 i 个字符比较而丢失可能的匹配，对于上式中存在的多个 k 值，应取其中最大的一个。这样，因 j-k 最小，即模式串向右滑动的位数最小，避免因右移造成的可能匹配的丢失。

23. 仅从两串含有相等的字符，不能判定两串是否相等，两串相等的充分必要条件是两串长度相等且对应位置上的字符相同（即两串串值相等）。

24.（1）s1 和 s2 均为空串；（2）两串之一为空串；（3）两串串值相等（即两串长度相等且对应位置上的字符相同）。（4）两串中一个串长是另一个串长（包括串长为 1 仅有一个字符的情况）的数倍，而且长串就好象是由数个短串经过连接操作得到的。

25、题中所给操作的含义如下：

//：连接函数，将两个串连接成一个串

substr（s,i,j）：取子串函数，从串 s 的第 i 个字符开始，取连续 j 个字符形成子串

replace（s1,i,j,s2）：置换函数，用 s2 串替换 s1 串中从第 i 个字符开始的连续 j 个字符

本题有多种解法，下面是其中的一种：

```
（1） s1=substr（s,3,1）      //取出字符：‘y’
（2） s2=substr（s,6,1）      //取出字符：‘+’
（3） s3=substr（s,1,5）      //取出子串：‘(xyz)’
（4） s4=substr（s,7,1）      //取出字符：‘*’
（5） s5=replace（s3,3,1,s2）//形成部分串：‘(x+z)’
（6） s=s5//s4//s1             //形成串 t 即‘(x+z）*y’
```

五、算法设计

1、[题目分析]判断字符串 t 是否是字符串 s 的子串，称为串的模式匹配，其基本思想是对串 s 和 t 各设一个指针 i 和 j，i 的值域是 0..m-n，j 的值域是 0..n-1。初始值 i 和 j

均为 0。模式匹配从 s_0 和 t_0 开始，若 $s_0=t_0$，则 i 和 j 指针增加 1，若在某个位置 $s_i!=t_j$，则主串指针 i 回溯到 i=i-j+1，j 仍从 0 开始，进行下一轮的比较，直到匹配成功（j>n-1），返回子串在主串的位置（i-j）。否则，当 i>m-n 则为匹配失败。

```
int index(char s[],t[],int m,n)
//字符串 s 和 t 用一维数组存储，其长度分别为 m 和 n。本算法求字符串 t 在字符串 s 中的第一次出现，如是，输出子串在 s 中的位置，否则输出 0。
{int i=0,j=0;
 while (i<=m-n && j<=n-1)
   if (s[i]==t[j]){i++;j++;}  //对应字符相等，指针后移。
      else {i=i-j+1;j=0;}      //对应字符不相等，I 回溯，j 仍为 0。
   if(i<=m-n && j==n) {printf("t 在 s 串中位置是%d",i-n+1);return(i-n+1);}//匹配成功
   else  return(0); //匹配失败
}//算法 index 结束
main ()//主函数
{char s[],t[]; int m,n,i;
 scanf("%d%d",&m,&n);  //输入两字符串的长度
 scanf("%s",s);        //输入主串
 scanf("%s",t);        //输入子串
 i=index(s,t,m,n);
}//程序结束
```

[程序讨论]因用 C 语言实现，一维数组的下标从 0 开始，m-1 是主串最后一个字符的下标，n-1 是 t 串的最后一个字符的下标。若匹配成功，最佳情况是 s 串的第 0 到第 n-1 个字符与 t 匹配，时间复杂度为 o（n）；匹配成功的最差情况是，每次均在 t 的最后一个字符才失败，直到 s 串的第 m-n 个字符成功，其时间复杂度为 o（(m-n）*n），即 o（m*n）。失败的情况是 s 串的第 m-n 个字符比 t 串某字符比较失败，时间复杂度为 o（m*n）。之所以串 s 的指针 i 最大到 m-n，是因为在 m-n 之后，所剩子串长度已经小于子串长度 n，故不必再去比较。算法中未讨论输入错误（如 s 串长小于 t 串长）。

另外，根据子串的定义，返回值 i-n+1 是子串在主串中的位置，子串在主串中的下标是 i-n。

2.［问题分析］在一个字符串内，统计含多少整数的问题，核心是如何将数从字符串中分离出来。从左到右扫描字符串，初次碰到数字字符时，作为一个整数的开始。然后进行拼数，即将连续出现的数字字符拼成一个整数，直到碰到非数字字符为止，一个整数拼完，存入数组，再准备下一整数，如此下去，直至整个字符串扫描到结束。

```
int CountInt（）
// 从键盘输入字符串，连续的数字字符算作一个整数，统计其中整数的个数。
 {int i=0，a[]；      // 整数存储到数组 a，i 记整数个数
  scanf（"%c"，&ch)；// 从左到右读入字符串
  while（ch!='#'） // '#' 是字符串结束标记
    if（isdigit（ch））// 是数字字符
     {num=0；        // 数初始化
      while（isdigit（ch）&& ch!='#'）// 拼数
       {num=num*10+'ch'-'0'；
        scanf（"%c"，&ch)；
```

```
        }
       a[i]=num; i++;
       if (ch!= '#') scanf ("%c", &ch);  // 若拼数中输入了 '#', 则不再输入
      } // 结束 while (ch! = '#')
  printf ("共有%d 个整数, 它们是:" i);
  for ( j= 0 ; j<i; j++)
   {printf ("%6d", a[j]);
    if ((j+1) %10==0) printf ("\n"); } // 每 10 个数输出在一行上
} // 算法结束
```

[算法讨论] 假定字符串中的数均不超过 32767, 否则, 需用长整型数组及变量。

3、[题目分析]设以字符数组 s 表示串, 重复子串的含义是由一个或多个连续相等的字符组成的子串, 其长度用 max 表示, 初始长度为 0, 将每个局部重复子串的长度与 max 相比, 若比 max 大, 则需要更新 max, 并用 index 记住其开始位置。

```
int LongestString(char s[],int n)
//串用一维数组 s 存储, 长度为 n, 本算法求最长重复子串, 返回其长度。
{int index=0,max=0;  //index 记最长的串在 s 串中的开始位置, max 记其长度
 int length=1,i=0,start=0;   //length 记局部重复子串长度, i 为字符数组下标
 while(i<n-1)
   if(s[i]==s[i+1]) {i++; length++;}
   else //上一个重复子串结束
      {if(max<length) {max=length; index=start; } //当前重复子串长度大, 则更新 max
       i++;start=i;length=1;                    //初始化下一重复子串的起始位置和长度
      }
 printf("最长重复子串的长度为%d, 在串中的位置%d\n", max,index);
 return(max);
}//算法结束
```

[算法讨论]算法中用 i<n-1 来控制循环次数, 因 C 数组下标从 0 开始, 故长度为 n 的串, 其最后一个字符下标是 n-1, 当 i 最大为 n-2 时, 条件语句中 s[i+1]正好是 s[n-1], 即最后一个字符。子串长度的初值数为 1, 表示一个字符自然等于其身。

算法的时间复杂度为 O(n), 每个字符与其后继比较一次。

4、[题目分析]教材中介绍的串置换有两种形式: 第一种形式是 replace (s,i,j,t), 含义是将 s 串中从第 i 个字符开始的 j 个字符用 t 串替换,第二种形式是 replace(s,t,v), 含义是将 s 串中所有非重叠的 t 串用 v 代替。我们先讨论第一种形式的替换。因为已经给定顺序存储结构, 我们可将 s 串从第 (i+j-1) 到串尾 (即 s.curlen) 移动 t.curlen-j 绝对值个位置 (以便将 t 串插入): 若 j>t.curlen, 则向左移; 若 j<t.curlen, 则向右移动; 若 j=t.curlen, 则不必移动。最后将 t 串复制到 s 串的合适位置上。当然, 应考虑置换后的溢出问题。

```
int replace(strtp s,t,int i,j)
//s 和 t 是用一维数组存储的串, 本算法将 s 串从第 i 个字符开始的连续 j 个字符用 t 串置换, 操作成功返回 1, 否则返回 0 表示失败。
{if(i<1 || j<0 || t.curlen+s.curlen-j>maxlen)
   {printf("参数错误\n");exit(0);}       //检查参数及置换后的长度的合法性。
```

```
if(j<t.curlen) //若 s 串被替换的子串长度小于 t 串长度，则 s 串部分右移,
    for(k=s.curlen-1;k>=i+j-1;k--) s.ch[k+t.curlen-j]=s.ch[k];
 else if (j>t.curlen)      //s 串中被替换子串的长度小于 t 串的长度。
        for(k=i-1+j;k<=s.curlen-1;k++) s.ch[k-(j-t.curlen)]=s.ch[k];
 for(k=0;k<t.curlen;k++) s.ch[i-1+k]=t.ch[k];  //将 t 串复制到 s 串的适当位置
if(j>t.curlen)                              s.curlen=s.curlen-(j-t.curlen);else
s.curlen=s.curlen+(t.curlen-j);
}//算法结束
```

[算法讨论]若允许使用另一数组，在检查合法性后，可将 s 的第 i 个（不包括 i）之前的子串复制到另一子串如 s1 中，再将 t 串接到 s1 串后面,然后将 s 的第 i+j 直到尾的部分加到 s1 之后。最后将 s1 串复制到 s。主要语句有：

```
for(k=0;k<i;k++) s1.ch[k]=s.ch[k]; //将 s1 第 i 个字符前的子串复制到 s1，这时 k=i-1
for(k=0;k<t.curlen;k++) s1.ch[i+k]=t.ch[k]//将 t 串接到 s1 的尾部
l=s.curlen+t.curlen-j-1;
for(k=s.curlen-1;k>i-1+j;k--);//将子串第 i+j-1 个字符以后的子串复制到 s1
     s1.ch[l--]=s.ch[k]
for(k=0;k<s.curlen+t.curlen-j;k++) s.ch[k]=s1.ch[k];//将结果串放入 s。
```

下面讨论 replace（s,t,v）的算法。该操作的意义是用串 v 替换所有在串 s 中出现的和非空串 t 相等的不重叠的子串。本算法不指定存储结构，只使用串的基本运算。

```
void replace(string s,t,v)
//本算法是串的置换操作，将串 s 中所有非空串 t 相等且不重复的子串用 v 代替。
{i=index(s,t);//判断 s 是否有和 t 相等的子串
 if(i!=0)//串 s 中包含和 t 相等的子串
  {creat(temp,”);            //creat 操作是将串常量（此处为空串）赋值给 temp。
   m=length(t);n=length(s);  //求串 t 和 s 的长度
   while(i!=0)
      {assign(temp,concat(temp,substr(s,1,i-1),v));//用串 v 替换 t 形成部分结果
       assign(s,substr(s,i+m,n-i-m+1));          //将串 s 中串后的部分形成新的 s 串
       n=n-(i-1)-m;                              //求串 s 的长度
       i=index(s,t);                             //在新 s 串中再找串 t 的位置
      }
      assign(s,contact(temp,s));  //将串 temp 和剩余的串 s 连接后再赋值给 s
    }//if 结束
}//算法结束
```

5、[题目分析]本题是字符串的插入问题，要求在字符串 s 的 pos 位置，插入字符串 t。首先应查找字符串 s 的 pos 位置，将第 pos 个字符到字符串 s 尾的子串向后移动字符串 t 的长度，然后将字符串 t 复制到字符串 s 的第 pos 位置后。

对插入位置 pos 要验证其合法性，小于 1 或大于串 s 的长度均为非法，因题目假设给字符串 s 的空间足够大，故对插入不必判溢出。

```
void  insert(char *s,char *t,int pos)
```

```
  //将字符串 t 插入字符串 s 的第 pos 个位置。
{int i=1,x=0;  char *p=s,*q=t;  //p，q 分别为字符串 s 和 t 的工作指针
 if(pos<1) {printf("pos 参数位置非法\n");exit(0);}
 while(*p!=' \0' &&i<pos) {p++;i++;} //查 pos 位置
    //若 pos 小于串 s 长度，则查到 pos 位置时，i=pos。
 if(*p == '/0') {printf("%d 位置大于字符串 s 的长度",pos);exit(0);}
 else       //查找字符串的尾
   while(*p!='/0') {p++; i++;}  //查到尾时，i 为字符'\0'的下标，p 也指向'\0'。
 while(*q!='\0') {q++; x++;}    //查找字符串 t 的长度 x，循环结束时 q 指向'\0'。
 for(j=i;j>=pos ;j--){*(p+x)=*p; p--;}//串 s 的 pos 后的子串右移，空出串 t 的位
置。
 q--;  //指针 q 回退到串 t 的最后一个字符
 for(j=1;j<=x;j++) *p--=*q--;  //将 t 串插入到 s 的 pos 位置上
[算法讨论] 串 s 的结束标记('\0')也后移了，而串 t 的结尾标记不应插入到 s 中。
```

6.[题目分析]本题属于查找，待查找元素是字符串（长 4），将查找元素存放在一维数组中。二分检索（即折半查找或对分查找），是首先用一维数组的“中间”元素与被检索元素比较，若相等,则检索成功，否则，根据被检索元素大于或小于中间元素，而在中间元素的右方或左方继续查找，直到检索成功或失败（被检索区间的低端指针大于高端指针）。下面给出类 C 语言的解法

```
typedef struct node
{char data[4];//字符串长 4
}node;
非递归过程如下：
int binsearch(node string [];int n;char name[4])
//在有 n 个字符串的数组 string 中，二分检索字符串 name。若检索成功，返回 name
在 string 中的下标，否则返回-1。
{int low = 0,high = n - 1;//low 和 high 分别是检索区间的下界和上界
 while(low <= high)
  {mid = (low + high) /2; //取中间位置
   if(strcmp(string[mid],name) ==0) return (mid); //检索成功
   else if(strcmp(string[mid],name)<0) low=mid+1; //到右半部分检索
        else high=mid-1;                          //到左半部分检索
  }
 return 0; //检索失败
}//算法结束
```

最大检索长度为 $\log_2 n$。

7. [题目分析]设字符串存于字符数组 X 中，若转换后的数是负数，字符串的第一个字符必为'-'，取出的数字字符，通过减去字符零（'0'）的 ASCII 值，变成数，先前取出的数乘上 10 加上本次转换的数形成部分数，直到字符串结束，得到结果。

```
long atoi(char X[])
 //一数字字符串存于字符数组 X 中，本算法将其转换成数
  {long num=0;
   int i=1; //i 为数组下标
   while (X[i]!= '\0') num=10*num+(X[i++]-'0');//当字符串未到尾，进行数的转
```

换

```
    if(X[0]=='-') return (-num);      //返回负数
    else return ((X[0]-'0')*10+num); //返回正数，第一位若不是负号，则是数字
}//算法 atoi 结束
```

[算法讨论]如是负数，其符号位必在前面，即字符数组的 x[0]，所以在作转换成数时下标 i 从 1 开始，数字字符转换成数使用 X[i]-'0',即字符与'0'的 ASCII 值相减。请注意对返回正整数的处理。

8.[题目分析]本题要求字符串 s1 拆分成字符串 s2 和字符串 s3，要求字符串 s2“按给定长度 n 格式化成两端对齐的字符串”，即长度为 n 且首尾字符不得为空格字符。算法从左到右扫描字符串 s1，找到第一个非空格字符，计数到 n，第 n 个拷入字符串 s2 的字符不得为空格，然后将余下字符复制到字符串 s3 中。

```
void format (char *s1,*s2,*s3)
 //将字符串 s1 拆分成字符串 s2 和字符串 s3，要求字符串 s2 是长 n 且两端对齐
{char *p=s1, *q=s2;
 int i=0;
 while(*p!= '\0' && *p== ' ') p++;//滤掉 s1 左端空格
 if(*p== '\0') {printf("字符串 s1 为空串或空格串\n");exit(0); }
 while( *p!='\0' && i<n){*q=*p; q++; p++; i++;}//字符串 s1 向字符串 s2 中复制
 if(*p =='\0'){ printf("字符串 s1 没有%d 个有效字符\n",n); exit(0);}
 if(*(--q)==' ' ) //若最后一个字符为空格，则需向后找到第一个非空格字符
  {p-- ;           //p 指针也后退
   while(*p==' '&&*p!='\0') p++;//往后查找一个非空格字符作串 s2 的尾字符
     if(*p=='\0') {printf("s1 串没有%d 个两端对齐的字符串\n",n); exit(0);}
   *q=*p;          //字符串 s2 最后一个非空字符
   *(++q)='\0';    //置 s2 字符串结束标记
   }
 *q=s3;p++;       //将 s1 串其余部分送字符串 s3。
 while (*p!= '\0') {*q=*p; q++; p++;}
 *q='\0';         //置串 s3 结束标记
}
```

9.[题目分析]两个串的相等，其定义为两个串的值相等，即串长相等，且对应字符相等是两个串相等的充分必要条件。因此，首先比较串长，在串长相等的前提下，再比较对应字符是否相等。

```
int equal(strtp s,strtp t)
//本算法判断字符串 s 和字符串 t 是否相等，如相等返回 1，否则返回 0
{if (s.curlen!=t.curlen) return (0);
for (i=0; i<s.curlen;i++)      //在类 C 中，一维数组下标从零开始
   if (s.ch[i]!= t.ch[i])return (0);
return (1);   //两串相等
}//算法结束
```

10.［问题分析］由于字母共 26 个，加上数字符号 10 个共 36 个，所以设一长 36 的整型数组，前 10 个分量存放数字字符出现的次数，余下存放字母出现的次数。从字符串中读出数字字符时,字符的 ASCII 代码值减去数字字符‘0’的 ASCII 代码值,得出其数值(0..9)，字母的 ASCII 代码值减去字符‘A’的 ASCII 代码值加上 10，存入其数组的对应下标分量中。

遇其它符号不作处理，直至输入字符串结束。

```
void Count ()
//统计输入字符串中数字字符和字母字符的个数。
{int i, num[36];
 char ch;
 for (i=0; i<36; i++) num[i]=0 ; // 初始化
 while ((ch=getchar ()) != '#')   // '#' 表示输入字符串结束。
     if ('0' <=ch<= '9') {i=ch-48;num[i]++;}          // 数字字符
     else  if ('A' <=ch<= 'Z') {i=ch-65+10;num[i]++;} // 字母字符

 for (i=0; i<10; i++)   // 输出数字字符的个数
       printf ("数字%d 的个数=%d\n", i, num[i]);
 for (i=10; i<36; i++) // 求出字母字符的个数
       printf ("字母字符%c 的个数=%d\n", i+55, num[i]);
} // 算法结束。
```

11.[题目分析]实现字符串的逆置并不难，但本题“要求不另设串存储空间”来实现字符串逆序存储，即第一个输入的字符最后存储，最后输入的字符先存储，使用递归可容易做到。

```
 void InvertStore(char A[])
//字符串逆序存储的递归算法。
{ char ch;
  static int i = 0;//需要使用静态变量
  scanf ("%c",&ch);
  if (ch!= '.')    //规定'.'是字符串输入结束标志
    {InvertStore(A);
     A[i++] = ch;//字符串逆序存储
    }
  A[i] = '\0';  //字符串结尾标记
}//结束算法 InvertStore。
```

12. 串 s'''可以看作由以下两部分组成：'caabcbca...a'和 'ca...a'，设这两部分分别叫串 s1 和串 s2，要设法从 s,s' 和 s''中得到这两部分，然后使用联接操作联接 s1 和 s2 得到 s''' 。

```
i=index(s,s');   //利用串 s'求串 s1 在串 s 中的起始位置
s1=substr(s,i,length(s) - i + 1); //取出串 s1
j=index(s,s''); //求串 s''在串 s 中的起始位置，s 串中'bcb'后是'ca...a')
s2=substr(s,j+3,length(s) - j - 2); //形成串 s2
s3=concat(s1,s2);
```

13.[题目分析]对读入的字符串的第奇数个字符，直接放在数组前面，对第偶数个字符，先入栈，到读字符串结束，再将栈中字符出栈，送入数组中。限于篇幅，这里编写算法，未编程序。

```
void RearrangeString()
 //对字符串改造，将第偶数个字符放在串的后半部分，第奇数个字符前半部分。
{char ch,s[],stk[];            //s 和 stk 是字符数组（表示字符串）和字符栈
 int i=1,j;                    //i 和 j 字符串和字符栈指针
```

```
 while((ch=getchar())!=' #' )// ' #' 是字符串结束标志
   s[i++]=ch;                  //读入字符串
 s[i]=' \0' ;                  //字符数组中字符串结束标志
 i=1;j=1;
 while(s[i])                   //改造字符串
   {if(i%2==0) stk[i/2]=s[i]; else s[j++]=s[i];
    i++;  }//while
 i--; i=i/2;                          //i 先从' \0' 后退，是第偶数字符的个数
 while(i>0) s[j++]=stk[i--]    //将第偶数个字符逆序填入原字符数组
}
```

14.[题目分析]本题是对字符串表达式的处理问题，首先定义 4 种数据结构：符号的类码，符号的 TOKEN 表示，变量名表 NAMEL 和常量表 CONSL。这四种数据结构均定义成结构体形式，数据部分用一维数组存储，同时用指针指出数据的个数。算法思想是从左到右扫描表达式，对读出的字符，先查出其符号类码：若是变量或常量，就到变量名表和常量表中去查是否已有，若无，则在相应表中增加之，并返回该字符在变量名表或常量表中的下标；若是操作符，则去查其符号类码。对读出的每个符号，均填写其 TOKEN 表。如此下去，直到表达式处理完毕。先定义各数据结构如下。

```
struct // 定义符号类别数据结构
   {char  data[7];   //符号
    char  code[7];   //符号类码
   }TYPL;
typedef  struct      //定义 TOKEN 的元素
   {int typ;         //符号码
    int addr;        //变量、常量在名字表中的地址
    }cmp;
struct {cmp data[50]; //定义 TOKEN 表长度<50
        int last;     //表达式元素个数
        }TOKEN;
struct {char data[15];   //设变量个数小于 15 个
        int last;        //名字表变量个数
       }NAMEL;
struct {char data[15];   //设常量个数小于 15 个
        int  last;       //常量个数
       }CONSL;
int operator (char cr)
 //查符号在类码表中的序号
 {for (i=3; i<=6; i++)
  if (TYPL.data[i]==cr)  return (i);
 }
void PROCeString ()
 //从键盘读入字符串表达式（以 '#' 结束），输出其 TOKEN 表示。
 {NAMEL.last=CONSL.last=TOKEN.last=0;   //各表元素个数初始化为 0
  TYPL.data[3]= '*'; TYPL.data[4]= '+'; TYPL.data[5]= '(';
  TYPL.data[6]= ')';      //将操作符存入数组
```

```
  TYPL.code[3]= '3'; TYPL.code[4]= '4'; TYPL.code[5]= '5';
  TYPL.code[6]= '6';  //将符号的类码存入数组
  scanf ("%c", &ch);   //从左到右扫描（读入）表达式。
  while (ch! = '#')    // '#' 是表达式结束符
   {switch (ch) of
     {case 'A' : case  'B' : case  'C' : //ch 是变量
                TY=0;        //变量类码为 0
                 for (i=1; i<=NAMEL.last; i++)
                   if (NAMEL.data[i]==ch) break; //已有该变量，i 记住其位置
                 if (i>NAMEL.last) {NAMEL.data[i]=ch; NAMEL.last++; }//变量
加入
      case '0' : case '1' : case '2' : case '3' : case '4' : case '5' ://处
理常量
     case '6' : case  '7' :case '8' : case '9' : TY=1; //常量类码为 1
                for (i=1; i<=CONSL.last; i++)
                   if (CONSL.data[i]==ch) break; ////已有该常量，i 记住其位
置
                if (i>CONSL.last) {CONSL.data[i]=ch; CONSL.last++;} //将新
常量加入
      default:    //处理运算符
                 TY＝operator (ch); //类码序号
                 i='\0';          //填入 TOKEN 的 addr 域（期望输出空白）
     }//结束 switch，下面将 ch 填入 TOKEN 表
    TOKEN.data [++TOKEN.last] .typ=TY; TOKEN.data [TOKEN.last] .addr=i;
    scanf ("%c", &ch);   //读入表达式的下一符号。
   }//while
 }//算法结束
```

[程序讨论]为便于讨论，各一维数组下标均以 1 开始，在字符为变量或常量的情况下，将其类码用 TY 记下，用 i 记下其 NAMEL 表或 CONSL 表中的位置，以便在填 TOKEN 表时用。在运算符（'+'，'*'，'('，')'）填入 TOKEN 表时，TOKEN 表的 addr 域没意义，为了程序统一，这里填入了'\0'。本题是表达式处理的简化情况（只有 3 个单字母变量，常量只有 0..9，操作符只 4 个），若是真实情况，所用数据结构要相应变化。

第 5 章 数组和广义表

一、选择题

1.设有一个 10 阶的对称矩阵 A，采用压缩存储方式，以行序为主存储，a_{11}为第一元素，其存储地址为 1，每个元素占一个地址空间，则 a_{85}的地址为（　　）。【燕山大学 2001 一、2（2 分）】

A. 13　　B. 33　　C. 18　　D. 40

2. 有一个二维数组 A[1:6，0:7] 每个数组元素用相邻的 6 个字节存储，存储器按字节编址，那么这个数组的体积是（①）个字节。假设存储数组元素 A[1，0]的第一个字节的地址是 0，则存储数组 A 的最后一个元素的第一个字节的地址是（②）。若按行存储，则 A[2，4]的第一个字节的地址是（③）。若按列存储，则 A[5，7]的第一个字节的地址是（④）。就一般情况而言，当（⑤）时，按行存储的 A[I，J]地址与按列存储的 A[J，I]地址相等。供选择的答案：【上海海运学院 1998 二、2 （5 分）】

①-④：A. 12　　B. 66　　C. 72　　D. 96　　E. 114　　F. 120

G. 156　　H. 234　　I. 276　　J. 282　　K. 283　　L. 288

⑤：A. 行与列的上界相同　　B. 行与列的下界相同

C. 行与列的上、下界都相同　　D. 行的元素个数与列的元素个数相同

3. 设有数组 A[i,j]，数组的每个元素长度为 3 字节，i 的值为 1 到 8 ，j 的值为 1 到 10，数组从内存首地址 BA 开始顺序存放，当用以列为主存放时，元素 A[5，8]的存储首地址为（　　）。

A. BA+141　　B. BA+180　　C. BA+222　　D. BA+225

【南京理工大学 1997 一、8 （2 分）】

4. 假设以行序为主序存储二维数组 A=array[1..100，1..100]，设每个数据元素占 2 个存储单元，基地址为 10，则 LOC[5，5]=（　　）。【福州大学 1998 一、10 （2 分）】

A. 808　　B. 818　　C. 1010　　D. 1020

5. 数组 A[0..5,0..6]的每个元素占五个字节，将其按列优先次序存储在起始地址为 1000 的内存单元中，则元素 A[5，5]的地址是（　　）。【南京理工大学 2001 一、13 （1.5 分）】

A. 1175　　B. 1180　　C. 1205　　D. 1210

6. 有一个二维数组 A[0:8,1:5],每个数组元素用相邻的 4 个字节存储，存储器按字节编址，假设存储数组元素 A[0,1]的第一个字节的地址是 0，存储数组 A 的最后一个元素的第一个字节的地址是（　①　）。若按行存储，则 A[3,5]和 A[5,3]的第一个字节的地址是（　②　）和（　③　）。若按列存储，则 A[7,1]和 A[2,4]的第一个字节的地址是（　④　）和（　⑤　）。【上海海运学院 1996 二、1 （5 分）】

①-⑤:A. 28　　B. 44　　C. 76　　D. 92　　E. 108　　F. 116　　G. 132　　H. 176　　I. 184　　J. 188

7. 将一个 A[1..100，1..100]的三对角矩阵，按行优先存入一维数组 B[1‥298]中，A 中元素 A_{6665}（即该元素下标 i=66，j=65），在 B 数组中的位置 K 为（　　）。供选择的答案：

A. 198　　B. 195　　C. 197　　【北京邮电大学 1998 二、5 （2 分）】

8. 二维数组 A 的元素都是 6 个字符组成的串，行下标 i 的范围从 0 到 8，列下标 j 的范圈从 1 到 10。从供选择的答案中选出应填入下列关于数组存储叙述中（　　）内的正确答案。

（1）存放 A 至少需要（　　）个字节；

（2）A 的第 8 列和第 5 行共占（　　）个字节；

（3）若 A 按行存放，元素 A[8，5]的起始地址与 A 按列存放时的元素（　　）的起始地

址一致。

供选择的答案：

（1）A. 90　　B. 180　　C. 240　　D. 270　　E. 540

（2）A. 108　　B. 114　　C. 54　　D. 60　　E. 150

（3）A. A[8,5]　　B. A[3,10]　　C. A[5,8]　　D. A[0,9]

【山东工业大学 2000 三、1 （4 分）】【山东大学 1998 三、1 (4 分)】

9. 二维数组 A 的每个元素是由 6 个字符组成的串，其行下标 i=0,1,…,8,列下标 j=1,2,…,10。若 A 按行先存储，元素 A[8,5]的起始地址与当 A 按列先存储时的元素（　　）的起始地址相同。设每个字符占一个字节。【西安电子科技大学 1998 一、2 （2 分）】

A. A[8,5]　　B. A[3,10]　　C. A[5,8]　　D. A[0,9]

10. 若对 n 阶对称矩阵 A 以行序为主序方式将其下三角形的元素(包括主对角线上所有元素)依次存放于一维数组 B[1..(n(n+1))/2]中，则在 B 中确定 a_{ij}(i<j)的位置 k 的关系为(　　)。

A. i*(i-1)/2+j　　B. j*(j-1)/2+i　　C. i*(i+1)/2+j　　D. j*(j+1)/2+i

【北京航空航天大学 2000 一、2 （2 分）】

11. 设 A 是 n*n 的对称矩阵，将 A 的对角线及对角线上方的元素以列为主的次序存放在一维数组 B[1..n(n+1)/2]中，对上述任一元素 a_{ij}(1≤i，j≤n，且 i≤j)在 B 中的位置为(　　)。

A. i(i-1)/2+j　　B. j(j-1)/2+i　　C. j(j-1)/2+i-1　　D. i(i-1)/2+j-1

【南京理工大学 1999 一、9（2 分）】

12. A[N，N]是对称矩阵，将下面三角（包括对角线）以行序存储到一维数组 T[N（N+1）/2]中，则对任一上三角元素 a[i][j]对应 T[k]的下标 k 是（　　）。【青岛大学 2002 二、6 （2 分）】

A. i（i-1）/2+j　　B. j（j-1）/2+i　　C. i（j-i）/2+1　　D. j（i-1）/2+1

13. 设二维数组 A[1.. m，1.. n]（即 m 行 n 列）按行存储在数组 B[1.. m*n]中，则二维数组元素 A[i，j]在一维数组 B 中的下标为(　　)。【南京理工大学 1998 一、2 （2 分）】

A.（i-1）*n+j　　B.（i-1）*n+j-1　　C. i*（j-1）　　D. j*m+i-1

14. 有一个 100*90 的稀疏矩阵，非 0 元素有 10 个，设每个整型数占 2 字节，则用三元组表示该矩阵时，所需的字节数是（　　）。【南京理工大学 1999 二、8 （2 分）】

A. 60　　B. 66　　C. 18000　　D. 33

15. 数组 A[0..4,-1..-3,5..7]中含有元素的个数（　　）。【中山大学 1998 二、5（2 分）】

A. 55　　B. 45　　C. 36　　D. 16

16. 用数组 r 存储静态链表，结点的 next 域指向后继，工作指针 j 指向链中结点，使 j 沿链移动的操作为(　　)。【南京理工大学 2001 一、16（1.5 分）】

A. j=r[j].next　　B. j=j+1　　C. j=j->next　　D. j=r[j]-> next

17. 对稀疏矩阵进行压缩存储目的是（　　）。【北京工商大学 2001 一、1 (3 分)】

A. 便于进行矩阵运算　B. 便于输入和输出　C. 节省存储空间　D. 降低运算的时间复杂度

18. 已知广义表 L=((x,y,z)，a，(u，t，w))，从 L 表中取出原子项 t 的运算是（　　）。

A. head（tail（tail（L）））　　B. tail（head（head（tail（L））））

C. head（tail（head（tail（L））））　　D. head（tail(head（tail（tail（L））））

【北京邮电大学 1998 二、4（2 分）】

19. 已知广义表 LS＝((a,b,c),(d,e,f)),运用 head 和 tail 函数取出 LS 中原子 e 的运算是(　　)。

A. head(tail(LS))　　B. tail(head(LS))

C. head(tail(head(tail(LS)))　　D. head(tail(tail(head(LS))))

【西安电子科技大学 2001 应用一、3（2 分）】

20. 广义表 A=(a,b,(c,d),(e,(f,g))),则下面式子的值为(　　)。【北京邮电大学 1999 一、2（2 分）】

Head(Tail(Head(Tail(Tail(A)))))

A. (g)　　B. (d)　　C. c　　D. d

21. 已知广义表: A=(a,b), B=(A,A), C=(a,(b,A),B), 求下列运算的结果:

tail(head(tail(C))) =(　　)。【长沙铁道学院 1998 三、4 (2 分)】

A.（a）　　B. A　　C. a　　D. (b)　　E. b　　F. (A)

22. 广义表运算式 Tail(((a,b),(c,d)))的操作结果是（　　)。【西安电子科技大学 1998 一、4（2 分)】

A. (c,d)　　B. c,d　　C. ((c,d))　　D. d

23. 广义表 L=（a，(b，c))，进行 Tail（L）操作后的结果为（　　)。【中山大学 1999 一、10】

A. c　　B. b，c　　C.（b，c）　　D.（(b，c))

24. 广义表 ((a,b,c,d)) 的表头是（　　)，表尾是（　　)。【青岛大学 2002 二、7 （2 分)】

A. a　　B.（）　　C.（a,b,c,d）　　D.（b,c,d）

25. 广义表（a,(b,c),d,e）的表头为（　　)。【中山大学 1998 二、6（2 分)】

A. a　　B. a,(b,c)　　C. (a,(b,c))　　D. (a)

26. 设广义表 L=((a,b,c))，则 L 的长度和深度分别为（　　)。【武汉大学 2000 二、9】

A. 1 和 1　　B. 1 和 3　　C. 1 和 2　　D. 2 和 3

27. 下面说法不正确的是(　　)。【南京理工大学 2001 一、3 （1.5 分)】

A. 广义表的表头总是一个广义表　　B. 广义表的表尾总是一个广义表

C. 广义表难以用顺序存储结构　　D. 广义表可以是一个多层次的结构

二、判断题

1. 数组不适合作为任何二叉树的存储结构。（　　）【南京航空航天大学 1995 五、2 （1 分)】

2. 从逻辑结构上看，n 维数组的每个元素均属于 n 个向量。（　　）

【东南大学 2001 一、2 （1 分)】【中山大学 1994　　一、2　　（2 分)】

3. 稀疏矩阵压缩存储后，必会失去随机存取功能。(　　)【中科院软件所 1997 一、1（1 分)】

4. 数组是同类型值的集合。(　)【上海海运学院 1996 一、3（1 分）1999 一、4（1 分)】

5. 数组可看成线性结构的一种推广，因此与线性表一样，可以对它进行插入，删除等操作。（　　）

【上海交通大学 1998 一、5】

6. 一个稀疏矩阵 A_{m*n} 采用三元组形式表示，　若把三元组中有关行下标与列下标的值互换，并把 m 和 n 的值互换，则就完成了 A_{m*n} 的转置运算。（　　）【西安交通大学 1996 二、8 (3 分)】

7. 二维以上的数组其实是一种特殊的广义表。（　　）【北京邮电大学 2002 一、5 （1 分)】

8. 广义表的取表尾运算，其结果通常是个表，但有时也可是个单元素值。（　　）

【南京航空航天大学 1996 六、2 （1 分)】

9. 若一个广义表的表头为空表，则此广义表亦为空表。（　　）

【中科院软件所 1997 一、8（1 分）】【长沙铁道学院 1998 一、8 (1 分)】

10. 广义表中的元素或者是一个不可分割的原子，或者是一个非空的广义表。（ ）

【合肥工业大学 2000 二、3 （1 分）】

11. 所谓取广义表的表尾就是返回广义表中最后一个元素。（ ）【合肥工业大学 2001 二、3 （1 分）】

12. 广义表的同级元素（直属于同一个表中的各元素）具有线性关系。（ ）

【华南理工大学 2002 一、9（1 分）】

13. 对长度为无穷大的广义表，由于存储空间的限制，不能在计算机中实现。（ ）

【华南理工大学 2002 一、10（1 分）】

14. 一个广义表可以为其它广义表所共享。（ ） 【山东大学 2001 一、2（1 分）】

三、 填空题

1. 数组的存储结构采用_____存储方式。【中山大学 1998 一、6（1 分）】

2. 设二维数组 A[-20..30,-30..20]，每个元素占有 4 个存储单元，存储起始地址为 200. 如按行优先顺序存储,则元素 A[25,18]的存储地址为__（1）__；如按列优先顺序存储,则元素 A[-18,-25]的存储地址为__（2）__。【北方交通大学 1999 二、3（4 分）】

3. 设数组 a[1..50,1..80]的基地址为 2000，每个元素占 2 个存储单元，若以行序为主序顺序存储，则元素 a[45,68]的存储地址为__（1）__;若以列序为主序顺序存储，则元素 a[45,68]的存储地址为__（2）__。

【华中理工大学 2000 一、5（2 分）】

4. 将整型数组 A[1..8，1..8]按行优先次序存储在起始地址为 1000 的连续的内存单元中，则元素 A[7，3]的地址是：_____。【合肥工业大学 1999 三、4（2 分）】

5. 二维数组 a[4][5][6]（下标从 0 开始计，a 有 4*5*6 个元素），每个元素的长度是 2，则 a[2][3][4]的地址是____。(设 a[0][0][0]的地址是 1000,数据以行为主方式存储)

【南京理工大学 2000 二、11（1.5 分）】

6. 设有二维数组 A[0..9,0..19],其每个元素占两个字节，第一个元素的存储地址为 100，若按列优先顺序存储，则元素 A[6,6]存储地址为_____。【北京工商大学 2001 二、5 （4 分）】

7. 已知数组 A[0..9,0..9]的每个元素占 5 个存储单元，将其按行优先次序存储在起始地址为 1000 的连续的内存单元中，则元素 A[6，8]的地址为_____。【合肥工业大学 2001 三、4（2 分）】

8. 已知二维数组 A[1..10，0..9]中每个元素占 4 个单元，在按行优先方式将其存储到起始地址为 1000 的连续存储区域时，A[5，9]的地址是：_____。【厦门大学 2002 六、5 （4 分）】

9. 用一维数组 B 与列优先存放带状矩阵 A 中的非零元素 A[i,j] (1≤i≤n,i-2≤j≤i+2),B 中的第 8 个元素是 A 中的第__（1）__行，第__（2）__列的元素。【北京邮电大学 2001 二、3（4 分）】

10. 设数组 A[0..8,1..10],数组中任一元素 A[i,j]均占内存 48 个二进制位,从首地址 2000 开始连续存放在主内存里，主内存字长为 16 位，那么

（1） 存放该数组至少需要的单元数是_____;

（2） 存放数组的第 8 列的所有元素至少需要的单元数是_____;

（3） 数组按列存储时，元素 A[5,8]的起始地址是_____。【中国矿业大学 2000 一、4（4 分）】

11. 设 n 行 n 列的下三角矩阵 A 已压缩到一维数组 B[1..n*（n+1）/2]中，若按行为主序存

储，则 A[i,j]对应的 B 中存储位置为_____。 【武汉大学 2000 一、1】

12. n 阶对称矩阵 a 满足 a[i][j]=a[j][i],i,j=1..n,，用一维数组 t 存储时，t 的长度为__(1)____，当 i=j，a[i][j]=t[(2)],i>j,a[i][j]=t[(3)],i<j,a[i][j]=t[(4)]。【青岛大学 2001 六、1（3 分）】

13. 已知三对角矩阵 A【1..9,1..9】的每个元素占 2 个单元，现将其三条对角线上的元素逐行存储在起始地址为 1000 的连续的内存单元中，则元素 A[7,8]的地址为_____。【合肥工业大学 2000 三、4（2 分）】

14. 设有一个 10 阶对称矩阵 A 采用压缩存储方式（以行为主序存储：a_{11}=1），则 a_{85} 的地址为_____。

【西安电子科技大学 1999 软件 一、3 （2 分）】

15. 所谓稀疏矩阵指的是_____。【厦门大学 2001 一、2 （14%/5 分）】

16. 对矩阵压缩是为了_____。 【北京理工大学 2000 二、3（2 分）】

17. 上三角矩阵压缩的下标对应关系为：_____。【福州大学 1998 二、6 (2 分)】【南京大学 1999】

18. 假设一个 15 阶的上三角矩阵 A 按行优先顺序压缩存储在一维数组 B 中，则非零元素 $A_{9,9}$ 在 B 中的存储位置 k=_____。(注：矩阵元素下标从 1 开始)【北京工商大学 2001 二、1（4 分）】

19. 设下三角矩阵

$$A=\begin{bmatrix} a11 & & & & \\ a21 & a22 & & & \\ a31 & a32 & a33 & & \\ .. & .. & .. & .. & .. \\ an1 & an2 & .. & .. & ann \end{bmatrix}$$

如果按行序为主序将下三角元素 A_{ij}(i,j)存储在一个一维数组 B[1..n(n+1)/2]中，对任一个三角矩阵元素 A_{ij} ,它在数组 B 中的下标为_____。【北方交通大学 2001 二、3】

20. 当广义表中的每个元素都是原子时，广义表便成了_____。【长沙铁道学院 1998 二、8 (2 分)】

21. 广义表的表尾是指除第一个元素之外，_____。 【中山大学 1998 一、7 （1 分)】

22. 广义表简称表，是由零个或多个原子或子表组成的有限序列，原子与表的差别仅在于__(1)___。为了区分原子和表，一般用___(2)____表示表，用___(3)_____表示原子。一个表的长度是指___(4)__，而表的深度是指__(5)__【山东工业大学 2000 一、3（3 分)】 【山东大学 1998 一、2 (3 分)】

23. 广义表的_____ 定义为广义表中括弧的重数。【重庆大学 2000 一、5】

24. 设广义表 L=((),()), 则 head(L)是__(1)___；tail(L)是__(2)____；L 的长度是__(3)___；深度是___(4)__。

【中科院计算所 1998 一、2（4 分)】【中国科技大学 1998 一、2（4 分)】

25. 已知广义表 A=(9,7,(8,10,(99)),12)，试用求表头和表尾的操作 Head( )和 Tail( )将原子元素 99 从 A 中取出来。 【西安交通大学 1996 四、5 (5 分)】

26. 广义表的深度是_____。【北京轻工业学院 2000 一、1（2 分)】

27. 广义表(a,(a,b),d,e,((i,j),k))的长度是__(1)__，深度是__(2)__。【山东大学 2001 三、9 (2 分)】

【西安电子科技大学 2001 软件 一、5 （2 分)】 【哈尔滨工业大学 2001 一、2 （2

分)】

28. 已知广义表 LS=(a，(b，c，d)，e),运用 head 和 tail 函数取出 LS 中原子 b 的运算是_____。

【燕山大学 2001 二、2 (3 分)】

29. 广义表 A=(((a，b)，(c，d，e)))，取出 A 中的原子 e 的操作是: _____。

【合肥工业大学 1999 三、5 (2 分)】

30. 设某广义表 H=(A，(a,b,c)) ，运用 head 函数和 tail 函数求出广义表 H 中某元素 b 的运算式_____。

【北京科技大学 1997 一、5】

31. 广义表 A((( ),(a,(b),c)))，head(tail(head(tail(head(A))))等于_____。

【合肥工业大学 2000 三、5 (2 分)】

32. 广义表运算式 HEAD(TAIL(((a,b,c),(x,y,z))))的结果是_____。

【西安电子科技大学 1999 软件 一、9 (2 分)】

33. 已知广义表 A=(((a，b)，(c)，(d，e)))，head (tail (tail (head (A)))) 的结果是_____。

【合肥工业大学 2001 三、5 (2 分)】

34. 利用广义表的 GetHead 和 GetTail 操作,从广义表 L=((apple,pear),(banana,orange)) 中分离出原子 banana 的函数表达式是_____。 【山东大学 2001 三、6 (2 分)】

35. 已知 a 数组元素共 5 个，依次为 12，10，5，3，1；b 数组元素共 4 个，依次为 4,6,8,15，则执行如下所示的过程语句 sort 后得到 c 数组各元素依次为 15,12,10,8,6,5,4,3,1；数组 a,b,c 的长度分别为 l=5,m=4,n=9 请在程序中方框内填入正确的成分，完成上述要求。

```
PROCEDURE   sort;
  VAR   i, j, k, x: integer;  d: ARRAY[1..m] OF  integer;
  BEGIN
    FOR i:=1 TO  m DO   d[i]:=(1)          ;
    i:=1; j:=1; k:=1;
    WHILE (i<=l) AND (j<=m) DO
       BEGIN
        IF a[i]>d[j] THEN BEGIN(2)  ; (3)  END
                     ELSE BEGIN (4)  ; (5)  END;
        c[k]:=x; (6)
       END;
    WHILE(7)   DO
       BEGIN c[k]:=a[i]; k:=k+1; i:=i+1; END;
    WHILE(8)   DO
       BEGIN c[k]:=d[j]; k:=k+1; j:=j+1; END;
  END. {sort}
```

【上海交通大学 1998 七 (12 分)】

36. 下列程序段 search(a,n,k)在数组 a 的前 n(n>=1)个元素中找出第 k(1<=k<=n)小的值。这里假设数组 a 中各元素的值都不相同。

```
#define  MAXN  100
int  a[MAXN],n,k;
int  search_c(int a[],  int  n,  int  k)
{int  low,  high,  i,  j,  m,  t;
 k--,;low=0 ;high=n-1;
```

```
do {i=low;  j=high ;  t=a[low];
    do{while  (i<j  && t<a[j])  j--;
        if  (i<j)  a[i++]=a[j];
        while  (i<j && t>=a[i])  i++
        if  (i<j)  a[j--]=a[i];
       }  while (i<j);
     a[i]=t;
     if (1)    ;
      if (i<k) low= (2)    ; else high= (3)    ;
 }while(4)    ;
 return(a[k]);
}
```

【上海大学 1999 一、1（8 分）】

37. 完善下列程序，每小题在 PASCAL 语言（a）和 C 语言（b）中任选一题。下面是一个将广义表逆置的过程。例如原来广义表为((a,b),c,(d,e))，经逆置后为:((e,d),c,(b,a))。

（a）算法的 PASCAL 语言过程描述（编者略）:（b）算法的 C 语言过程描述：

```
typedef  struct  glistnode
{int  tag;
 struct  glistnode *next;
 union{char data;
       struct{struct glistnode *hp,*tp;}ptr;
      }val;
}*glist,gnode;
glist  reverse(p)
glist p;
{glist  q,h,t,s;
 if(p==NULL)  q=NULL;
 else
  {if(1)   { q=(glist)malloc(sizeof(gnode));  q->tag=0;
             q->val.data=p->val.data;  }
   else {(2)
         if (3)
            {t=reverse(p->val.ptr.tp); s=t;
             while(s->val.ptr.tp!=NULL)   s=s->val.ptr.tp;
             s->val.ptr.tp=(glist)malloc(sizeof(gnode));
             s=s->val.ptr.tp;s->tag=1;s->val.ptr.tp=NULL;
             s->val.ptr.hp=h; (4)    }
          else {q=(glist)malloc(sizeof(gnode));q->tag=1;
                q->val.ptr.tp=NULL; (5)   ; }
         }
    }
 return(q);
}
```

【上海大学 2002 六、3 （10 分）】

38. 完善下列程序，每小题在 PASCAL 语言（a）和 C 语言（b）中任选一题。下面的程序将

数列 1,2,3,…,n*n,依次按蛇型方式存放在二维数组 A[1..n,1..n]中。即（示意圖编者略）。

（a）算法的 PASCAL 语言程序描述（编者略）：（b）算法的 C 语言程序描述：

```
#define  NMAX  10
#include  “stdio.h”
main()
{ int  i,j,n,k,p,q,m;
  int  a [NMAX][NMAX];
  scanf(“%d”,&n);
  m=1;
  for(k=1; (1)      ; k++)
     {if (k<n)  q=k;     else(2)      ;
      for(p=1;p<=q;p++)
       {if(3)     {i=q-p+1; j=p; }
        else{i=p; j=q-p+1; }
        if(4)     {i=i+n-q; j=j+n-q; }
        a[i][j]=m; (5)     ;
       }
      for (i=1; i<=n;i++)
        { for (j=1; j<=n; j++)
          printf (“%4d”,a[i][j]) ;printf(“\n”);
       }
      }
   }
```

【上海大学 2002 六、1 （10 分）】

39. 约瑟夫环问题：设有 n 个人围坐一圈，并按顺时针方向 1—n 编号。从第 s 个人开始进行报数，报数到第 m 个人，此人出圈，再从他的下一个人重新开始从 1 到 m 的报数进行下去，直到所有的人都出圈为止。

```
PROCEDURE Josef (A:ARRAY [1..n]  OF  integer;  s,m:integer);
  BEGIN
    FOR i:= 1 TO n  DO  A[i]:=i;
    sl:=s;
    FOR  i:=n  DOWNTO 2  DO
       BEGIN sl:= (1)      ; //计算出圈人 sl
             IF  sl=0  THEN  (2)      ;
             w:=A[sl];       //A[s1]出圈
             FOR  j:= (3)      DO  A[j]:=A[j+1];
             A[i]:=w;
       END;
    write('出圈序列为：'); //输出出圈序列
    FOR i :=n  DOWNTO 1 DO write(A[i]); writeln ;
END;
```

【华南师范大学 2000 五、2 (9 分)】

40. 设有一个背包可以放入的物品重量为 S，现有 n 件物品，重量分别为 W_1，W_2，...，W_n。问能否从这 n 件物品中选择若干件放入背包，使得放入的重量之和正好是 S。设布尔函数 Knap(S，n)表示背包问题的解，W_i(i=1,2,...，n)均为正整数，并已顺序存储地在数组 W 中。请在下列算法的下划线处填空，使其正确求解背包问题。

```
Knap(S，n)
若 S=0
  则 Knap←true
  否则若（S<0）或(S>0 且 n<1)
        则 Knap←false
        否则若 Knap(1)  ，    _=true
              则 print(W[n])；Knap ←true
             否则 Knap←Knap(2)  _ ，    _
```

【山东工业大学 1996 五（10 分）1998 二、1 （4 分)】

四　应用题

1. 数组 A[1..8，-2..6，0..6]以行为主序存储，设第一个元素的首地址是 78，每个元素的长度为 4，试求元素 A[4,2,3]的存储首地址。【厦门大学 1998　五、1 （5 分）】

2. 已知 b 对角矩阵 $(a_{ij})_{n*n}$，以行主序将 b 条对角线上的非零元存储在一维数组中，每个数据元素占 L 个存储单元，存储基地址为 S，请用 i，j 表示出 a_{ij} 的存储位置。【北方交通大学 1996 三（10 分)】

3. 数组 A 中，每个元素 A[i,j]的长度均为 32 个二进位，行下标从-1 到 9，列下标从 1 到 11，从首地址 S 开始连续存放主存储器中，主存储器字长为 16 位。求：

（1）存放该数组所需多少单元？

（2）存放数组第 4 列所有元素至少需多少单元？

（3）数组按行存放时，元素 A[7,4]的起始地址是多少？

（4）数组按列存放时，元素 A[4,7]的起始地址是多少？ 【大连海事大学 1996 四、1 (6 分)】

4. 假设按低下标优先存储整型数组 A（-3:8,3:5,-4:0,0:7）时，第一个元素的字节存储地址是 100，每个整数占 4 个字节，问 A（0，4，-2，5）的存储地址是什么？【清华大学 1996 三】

5. 设有三维数组 A[-2:4,0:3,-5:1]按列序存放，数组的起始地址为 1210，试求 A(1,3,-2）所在的地址。【长沙铁道学院 1997　三、1 (3 分)】

6. 三维数组 A[1..10,-2..6,2..8]的每个元素的长度为 4 个字节，试问该数组要占多少个字节的存储空间？如果数组元素以行优先的顺序存贮，设第一个元素的首地址是 100，试求元素 A[5,0,7] 的存贮首地址。

【上海海运学院 1995 三（6 分） 1997 三（8 分)】

7. 设有五对角矩阵 $A=(a_{ij})_{20*20}$，按特殊矩阵压缩存储的方式将其五条对角线上的元素存于数组 A[-10：m]中，计算元素 A[15,16]的存储位置。【东北大学 1999 一、2（4 分)】

8. 数组 A[0..8, 1..10] 的元素是 6 个字符组成的串，则存放 A 至少需要多少个字节？ A 的第 8 列和第 5 行共占多少个字节？若 A 按行优先方式存储，元素 A[8,5]的起始地址与当 A 按列优先方式存储时的哪个元素的起始地址一致？ 【厦门大学 2000 五、3（14%/3 分）】

9. 若按照压缩存储的思想将 n×n 阶的对称矩阵 A 的下三角部分（包括主对角线元素）以行序为主序方式存放于一维数组 B[1..n(n+1)/2]中，那么，A 中任一个下三角元素 $a_{ij}(i \geqslant j)$，在数组 B 中的下标位置 k 是什么？【北京航空航天大学 1998　一、4（4 分)】

10. 设 m×n 阶稀疏矩阵 A 有 t 个非零元素，其三元组表表示为 LTMA[1..（t+1）,1..3]，试问：非零元素的个数 t 达到什么程度时用 LTMA 表示 A 才有意义？【北京航空航天大学 1998 一、5（4 分)】

11. 利用三元组存储任意稀疏数组时，在什么条件下才能节省存储空间。【西北工业大学

1998 三、2(5 分)】

12. 对一个有 t 个非零元素的 A_{mn} 矩阵，用 B[0..t][1..3]的数组来表示，其中第 0 行的三个元素分别为 m,n,t，从第一行开始到最后一行，每行表示一个非零元素；第一列为矩阵元素的行号，第二列为其列号，第三列为其值。对这样的表示法，如果需要经常进行该操作---确定任意一个元素 A[i][j]在 B 中的位置并修改其值，应如何设计算法可以使时间得到改善？【长沙铁道学院 1998 四、4 (6 分)】

13. 有一个二维数组 A[0:8,1:5],每个数组元素用相邻的 4 个字节存储,存储器按字节编址,假设存储数组元素 A[0,1]的第一个字节的地址是 0,那么存储数组的最后一个元素的第一个字节的地址是多少?若按行存储,则 A[3,5]和 A[5,3]的第一个字节的地址是多少?若按列存储,则 A[7,1]和 A[2,4]的第一个字节的地址是多少?【上海海运学院 1999 三（10 分)】

14. 设有三对角矩阵$(a_{i,j})_{m\times n}$,将其三条对角线上的元素逐行的存于数组 B(1:3n-2)中，使得 $B[k]=a_{i,j}$，求：

（1）用 i,j 表示 k 的下标变换公式；

（2）若 $n=10^3$,每个元素占用 L 个单元，则用 B[K]方式比常规存储节省多少单元。

【西安电子科技大学 1996 二、4 （5 分)】

15. 已知 A 为稀疏矩阵，试从空间和时间角度，比较采用两种不同的存储结构（二维数组和三元组表）完成求　　运算的优缺点。【西安电子科技大学 1996 二、6（5 分)】

16. 特殊矩阵和稀疏矩阵哪一种压缩存储后失去随机存取的功能？为什么？

【北京邮电大学 2001 三、1（5 分)】

17. 试叙述一维数组与有序表的异同。【西安电子科技大学 1999 计应用一、2（5 分)】

18. 一个 n╳n 的对称矩阵，如果以行或列为主序存入内存，则其容量为多少？

【西安电子科技大学 1999 计应用 一、3（5 分)】

19. 给出数组 A：ARRAY[3..8,2..6] OF INTEGER;当它在内存中按行存放和按列存放时，分别写出数组元素 A[i,j]地址计算公式（设每个元素占两个存储单元)。【南开大学 1998 一(8 分)】

20. 已知 n 阶下三角矩阵 A（即当 i<j 时，有 $a_{ij}=0$)，按照压缩存储的思想，可以将其主对角线以下所有元素(包括主对角线上元素)依次存放于一维数组 B 中，请写出从第一列开始采用列序为主序分配方式时在 B 中确定元素 a_{ij} 的存放位置的公式。【北京航空航天大学 1999 二(10 分)】 【中山大学 1999 三 2】

21. 设有三对角矩阵(a_{ij})n*n，将其三条对角线上的元素逐行地存于数组 B(1:3n-2)中，使得 $B[k]=a_{ij}$，求：（1）用 i,j 表示 k 的下标变换公式；

（2）用 k 表示 i,j 的下标变化公式。

【山东科技大学 2001 一、6 （6 分)】【长沙铁道学院 1997 五、1 (10 分)】

【东北大学 2002 一 (4 分)】【北京工业大学 2000 二、1 (9 分)】【南京航空航天大学 2000 四】

22. 算法 Print 及所引用的数组 A 的值如下，写出调用 Print(1)的运行结果（其中 n=15)。

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	B	C	D	E	F	G	0	0	H	0	I	J	K	L

```
PROCEDURE print (i: integer);
        BEGIN
```

```
        IF (i<=n) AND (A[i] <> '0') THEN
          BEGIN Print (2*i); write (A[i]); Print (2*i+1); END;
      END;
```

【合肥工业大学 1999 四、1（5 分）】

23. 设数组 A 的长度为 2N，前 N 个元素 A[1..N]递减有序，后 N 个元素 A[N+1.. 2N]递增有序，且 2N 是 2 的整数次幂，即 $k=\log_2 2N$ 为整数。例如 A[1..8]=[90,85,50,10,30,65,80,100]满足上述要求，这里 N=4,k=3,A 的前 4 个元素和后 4 个元素分别递减和递增有序。用此例调用如下的 Demo 过程，并要求：

（1）给出 for 循环中每次执行 PerfectShuffle（A,N）和 CompareExchange（A,N）的结果；（2）解释 Demo 的功能；（3）给出 Demo 的时间复杂度。

```
PROCEDURE PerfectShuffle(VAR A:arraytype; N:integer)
 [ i:=1; j:=1;
   WHILE i<=N DO
    [ B[j]:=A[i]; B[j+1]:=A[i+N]; i:=i+1; j:=j+2; ]
   A[1..2N]:=B[1..2N];        //B copy to A
 ]
PROCEDURE CompareExchange(VAR A:arraytype; N:integer)
 [ j:=1;
   WHILE  j<2N  DO
     [ IF  A[j]>A[j+1] THEN  A[j]←→A[j+1]; //交换 A[j]和 A[j+1]
       j:=j+2;    ]
  ]
PROCEDURE Demo (VAR A:arraytype;N:integer)
      //A 的长度为 2N,k=log2N 为整数
   [ FOR  i:=1 TO  log2N  DO
       [  PerfectShuffle(A,N);   CompareExchange(A,N);   ]
   ]
```

【中科院计算所 1998 四 （15 分）】【中国科技大学 1998 4（15 分）】

24. 现有算法及正整数 n 和数组 A 如下，求数组 C 的值。

```
VAR  A,B,C:Array[1..100] of  integer;
FUNC AAA(s,t:integer):integer;
 IF s=t  THEN  IF  B[s]=0 THEN AAA:=S  ELSE  AAA:=-s ELSE
  BEGIN
    l:=AAA(s,(s+t) DIV 2);
    r:=AAA((s+t) DIV 2+1,t);
    IF l>0 THEN  AAA:=l ELSE AAA:=r;
    IF (r>0) AND (A[l]>A[r]) THEN AAA:=r
  END
ENDF;
PROC BBB;
 FOR i:=1 TO  n  DO  B[i]:=0;
 FOR i:=1 TO  n  DO  B[AAA(1,n)]:=i;
 FOR i:=1 TO  n  DO  C[B[i]]:=A[i];
ENDP;
```

初始值：n=10,A={121,22,323,212,636,939,828,424,55,262};

【北京邮电大学 2002 五、1（10 分）】

25. 设有矩阵 a 且 $a=\begin{bmatrix}2 & 1 & 3\\ 3 & 3 & 1\\ 1 & 2 & 1\end{bmatrix}$ 执行下列语句后，矩阵 c 和 a 的结果分别是什么？

（1） FOR i:=1 TO 3 DO
FOR j:=1 TO 3 DO c[i,j]:=a[a[i,j],a[j,i]]

（2） FOR i:=1 TO 3 DO
FOR j:=1 TO 3 DO a[i,j]:=a[a[i,j],a[j,i]]

【浙江大学 1997 三 (8 分)】

26. 设矩阵 A 为 $\begin{bmatrix}2 & 1 & 3\\ 3 & 3 & 1\\ 1 & 2 & 1\end{bmatrix}$

（1）执行语句

```
FOR   i:=1  TO 3 DO
FOR   j:=1  TO 3 DO                ①
      C[i,j]:=A[A[i,j],A[j,i]]
```

结果 C 矩阵的值是什么？

（2）所选择的下标 i,j 的次序有关系吗？

（3）在语句①中，用 A 代替 C，A 的结果值是什么？

（4）对 i,j 这对下标取反序，即

（3，3），（3，2），（3，1），……，（1，3），（1，2），（1，1）

重复执行（3），把所得结果与（3）中所得结果作比较。【吉林大学 1995 二 (15 分)】

27. 指出下列算法中错误、低效之处，并将其改成一个正确且高效的算法。

```
PROCEDURE  delk(A, m , last,i, k) ;
  {从数组 A[1..last]中删除第 i 个元素起的 k 个元素，m 为 A 上限}
    BEGIN
      IF(i<1) OR (i>last) OR(k<0) OR(last>m)
        THEN   write  ('error')
      ELSE  FOR count: = 1  TO  k  TO
              [FOR  j:=last  DOWNTO   i+1  DO
                A[j-1]:=A[j];
              last:=last-1]
    ENDP;{delk}
```

【北方交通大学 1997 一 (10 分)】

28. 设输入为整数数组 A[1..n]，其中 1<=A[i]<=k(1<=i<=n)；输出数组为 b[1..n]；c[1..k] 是临时工作空间；阅读下述算法后，回答下列问题：

```
PROC Demo(A,B,k){
(1)FOR i:=1 TO k  DO  C[i]:=0;
(2)FOR j:=1 TO n  DO   C[A[j]]:= C[A[j]]+1;
(3)FOR i:=2 TO k  DO   C[i]:= C[i]+ C[i-1]
(4)FOR j:=n DOWNTO 1 DO {
```

(5) B[C[A[j]]]:=A[j];

(6)C[A[j]]:=C[A[j]]-1 } }

(a).当标号（2）行的循环执行完后，C[i](1<=i<=n)的值有何意义？

(b).当标号（3）行的循环执行完后，C[i](1<=i<=n)的值有何意义？

(c).算法执行后，B 的内容有何特点？

(d).当 k=O(n)时，算法的时间复杂度是多少？ 【中科院软件所 1997 二、2（9 分）】

29. 上三角阵 A（N*N）按行主序压缩存放在数组 B 中，其中 A[i,j]=B[k].写出用 i，j 表示的 k。

【北京工业大学 2001 二、1 (5 分)】

30. 设有上三角矩阵$(a_{ij})_{n*n}$，将其上三角中的元素按先行后列的顺序存于数组 B（1:m）中，使得 B[k]= a_{ij}且 k=f1(i)+f2(j)+c，请推导出函数 f1,f2 和常数 c，要求 f1 和 f2 中不含常数项。

【中科院自动化所 1999】 【山东科技大学 2002 一、5 （6 分)】

31. 设矩阵 $A=\begin{bmatrix}2&0&0&4\\0&0&3&0\\0&3&0&0\\4&0&0&0\end{bmatrix}$

(1) 若将 A 视为对称矩阵，画出对其压缩存储的存储表，并讨论如何存取 A 中元素 a_{ij} (0<=i,j<4)；

(2) 若将 A 视为稀疏矩阵，画出 A 的十字链表结构。【北京科技大学 1997 三 （10 分)】

32. 设对称矩阵 $A=\begin{bmatrix}1&0&0&2\\0&3&0&0\\0&0&0&5\\2&0&5&0\end{bmatrix}$

（1）若将 A 中包括主对角线的下三角元素按列的顺序压缩到数组 S 中，即 S:

1	0	0	2	3	0	0	0	5	0
下标：1	2	3	4	5	6	7	8	9	10

试求出 A 中任一元素的行列下标[i,j](1<=i,j<=4)与 S 中元素的下标 K 之间的关系.

（2）若将 A 视为稀疏矩阵时,画出其三元组表形式压缩存储表。【北京科技大学 1999 三（10 分)】

33. 设对角线矩阵 $A=\begin{bmatrix}1&2&0&0&0\\1&0&1&0&0\\0&2&1&0&0\\0&0&0&0&1\\0&0&0&3&5\end{bmatrix}$（行列下标 i ,j 满足：1≤i,j≤5）

（1）若将矩阵 A 压缩存储到数组 S 中：

1	2	1	0	1	2	1	0	0	0	1	3	5

下标： 1 2 3 4 5 6 7 8 9 10 11 12 13

试求出 A 中已存储之元素的行列下标（i， j）与 S 中元素的下标 K 之间的关系

（2）若将 A 视为稀疏距阵时，请画出其行逻辑链接顺序表。【北京科技大学 2000 三(10 分)】

34. 设有一个三维数组 a[c1:d1,c2:d2,c3:d3]，其中 ci:di 是第 i 维的界偶，如该数组在内存中按行排列，且 a[c1,c2,c3]的地址为 a0，那么请导出下标变量 a[i1,i2,i3]的地址，假设每个元素占 L 个单元。

【山东师范大学 1999 四、1 （5 分)】

35.假定有下列 n x n 矩阵(n 为奇数)

$$A=\begin{bmatrix} a_{11} & 0 & \cdots & \cdots & \cdots & 0 & a_{1n} \\ 0 & a_{22} & \cdots & \cdots & \cdots & a_{2,n-1} & 0 \\ & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & \cdots & \cdots & a_{\frac{n+1}{2},\frac{n+1}{2}} & \cdots & & 0 \\ & & & \cdots\cdots & & 0 & \\ & & & & & & \\ a_{n1} & 0 & & & & 0 & a_{nn} \end{bmatrix}$$

如果用一维数组 B 按行主次序存储 A 的非零元素，问：

（1）A 中非零元素的行下标与列下标的关系；

（2）给出 A 中非零元素 a_{ij} 的下标(i,j)与 B 中的下标 R 的关系;

（3)假定矩阵中每个元素占一个存贮单元且 B 的起始地址为 A_0,给出利用 a_{ij} 的下标(i,j)定位在 B 中的位置公式。 【上海交通大学 1998 三(12 分)】

36. 对于一个对称矩阵采用压缩存储，只存放它的上三角部分，并按列存放。例如对于一个 n*n 的对称矩阵 A （如右图），用一个一维数组 B 来存放它的上三角部分：

B=[A_{11}，A_{12}，A_{22}，A_{13}，A_{23}，A_{33}，A_{14}，…，A_{1n}，A_{2n}，…，A_{nn}]

$$\begin{bmatrix} A_{33} & \cdots & \cdots & A_{3n} \\ & \cdots & \cdots & \cdots \\ & & \cdots & \cdots \\ & & & \\ & & & A_{nn} \end{bmatrix}$$

同时有两个函数：MAX(i,j)和 MIN(i,j)，分别计算下标 i 和 j 中的大者与小者。试利用它们给出求任意一个 Aij 在 B 中存放位置的公式。(若式中没有 MAX(I,j)和 MIN（i， j）则不给分）

【清华大学 1997 五 (10 分)】

37. 用三元数组表示稀疏矩阵的转置矩阵,并简要写出解题步骤。【山东工业大学 1995 五（10 分)】

38. 简述广义表属于线性结构的理由。【西北大学 2000 一、5 (3 分)】

39. 数组，广义表与线性表之间有什么样的关系？【西北工业大学 1998 一、2 (4 分)】

40. 什么是广义表？请简述广义表和线性表的主要区别。【北京大学 1997 二、2 （5 分）】

41. 求下列广义表的运算结果。【南京航空航天大学 1998 三 （10 分）】

（1）CAR(CDR(((a,b),(c,d),(e,f))))

（2）CDR(CAR(((a,b),(c,d),(e,f))))

（3）CAR(CDR(CAR(((a,b),(e,f)))))

（4）CDR(CAR(CDR(((a,b),(e,f)))))

（5）CDR(CDR(CAR(((a,b),(e,f)))))

注：CAR 运算相当于有些教材中的 Head 运算，CDR 运算相当于 Tail 运算。

42. 利用广义表的 Head 和 Tail 运算，把原子 d 分别从下列广义表中分离出来，L1＝(((((a),b),d),e))；L2＝（a,(b,((d)),e))。 【北方交通大学 1996 一、2（5 分）】

类似本题的另外叙述有：

(1) 已知广义表 L=((((a))),((b)),(c),d)，试利用 head 和 tail 运算把原子项 c 从 L 中分离出来。

【北京邮电大学 1992 三、2（25/4 分）】【青岛海洋大学 1999 一、6 （5 分）】

(2) 画出下列广义表的存储结构图，并利用取表头和取表尾的操作分离出原子 e。

（ a, (( ) , b) , (((e))))。

【清华大学 1995 二 （10 分）】

(3) 已知广义表 A=((a,b,c),(d,e,f)) 试写出从表 A 中取出原子元素 e 的运算。

【西安电子科技大学 1996 二、3 （5 分）】

(4)请将香蕉 banana 用工具 H( )—Head( )，T( )—Tail( )从 L 中取出。

L=(apple,(orange,(strawberry,(banana)),peach),pear)

【北京邮电大学 2000 三、2 （5 分）】

(5) 试利用广义表取表头 head(ls)和取表尾 tail(ls)的基本运算,将原子 d 从下列表中分解出来，请写出每一步的运算结果。

L=((a,(b)),((c,d)),(e,f)) 【北京工商大学 2001 三 （10 分）】

(6) 画出广义表 A=（a, (b,()) ,((),c))）的第一种存储结构（表结点第二指针指向余表）图，并用取首元（head()）和取尾元（tail()）函数表示原子 c。【北京工业大学 2001 二、2 (5 分)】

43. 画出下列广义表的两种存储结构图((),A,(B,(C,D)),(E,F))。【南京航空航天大学 1999 三（10 分）】

44. 假设采用以下两种结点的链表作为广义表的存贮结构，表结点：(tag=1,hp,tp)， 元素结点；(tag=0,data)。请画出下列广义表的存储结构图，并求它的深度和长度。

（ ( ) , ((( )) , ((( )))))

【北方交通大学 1998 一（13 分）】

45. 知广义表 A= (((a)), (b), c, (a), (((d, e))))

（1）画出其一种存贮结构图；

（2）写出表的长度与深度；

（3）用求头部，尾部的方式求出 e。【东北大学 1997 一、2 (5 分)】

46. 画出具有共享结构广义表 (((b,c) ,d) ,(a),((a),((b,c),d)),e,()) 的存贮表示。

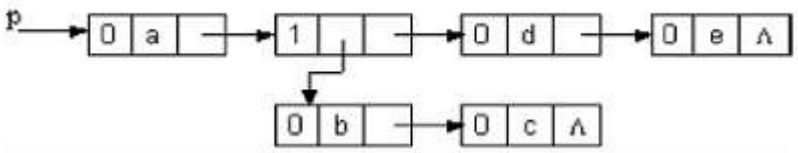
【北京工业大学 1996 一、3 (6 分)】

47. 广义表的结点结构如下:(TAG,DATA,LINK)。其中 LINK 为指向表中下一元素的指针;TAG 为标志域；DATA 为数据域，具体含义如下: TAG=0 表示该结点为原子结点，DATA 为其数据；TAG=1 表示该结点为一个子表，DATA 为指向该子表的指针。

(1)说明下列算法 A 的功能(注:算法中 p,t,m,n,r,q 为指针;算法中的 NIL 对应图中的^)

```
PROCEDURE  A(p,t)
   BEGIN
    q:=NIL;
    WHILE  p<>NIL DO
     BEGIN
      IF p^.TAG<>0 THEN
       BEGIN
         m:=p^.DATA;
         A(m,n);
         p^.DATA:=n;
       END;
      r:=p^.LINK;
      p^.LINK:=q;
      q:=p;
      p:=r
    END;
   t:=q;
END.
```

（2）对于 p 所指的广义表,画出执行算法 A 后的表结构以及 p，t 的值:



【北方交通大学 1993 六（20 分)】

类似本题的另外叙述有：

题目基本相同,差别仅在于子表(b,c)与原子 d 的前后顺序颠倒。【浙江大学 1994 六 (7 分)】

48. 写出对广义表 A=(x,((a,b),c,d))作运算 head(head(tail(A)))后的结果。

【西安电子科技大学 2000 计应用 一、5（5 分)】

49. 写出广义表的运算结果: TAIL[HEAD[TAIL[((a,b),(c,d))]]]=?

【武汉交通科技大学 1996 二、7 (3 分)】

50. 广义表中原子个数是否即为广义表的长度？【西安电子科技大学 2000 计应用一、9（5 分)】

51. 给出下列所示的 3 元多项式的广义表表示（分别以 X_1，X_2，X_3第一到第三层变元）

$P(X_1X_2X_3)=X_1^5X_2^3X_3+2X_1^5X_2^2X_3^4+5X_1^5X_2^3X_3^3+3X_1X_2^4X_3^2+X_2X_3+6$

【华南理工大学 2001 一、2（4 分)】

52. 设某表 H 如下：

A		B	C		X
a1	a2	b1	c1	c2	

其中 A,B,C 为子表名，a1,a2,b1,c1,c2,x 为其元素。

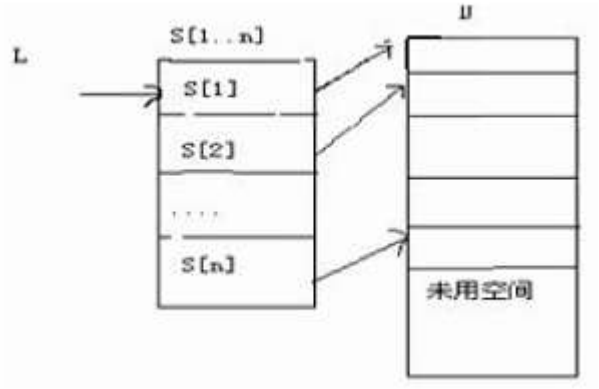
（1）试用广义表形式表示 H,并写出运算 HEAD(H)和 TAIL(H) 函数从 H 中取出单元素 a2 的运算；

（2）画出表 H 的链式存储结构；【北京科技大学 1998 三（10 分）】

五 算法设计题

1. 设有大小不等的 n 个数据组（n 个数据组中数据的总数为 m），顺序存放在空间区 D 内，每个数据占一个存储单元，数据组的首地址由数组 S 给出，（如下图所示），试编写将新数据 x 插入到第 i 个数据组的末尾且属于第 i 个数据组的算法，插入后，空间区 D 和数组 S 的相互关系仍保持正确。

【东北大学 2000 六（15 分）】



2. 以三元组表存贮的稀疏矩阵 A，B 非零元个数分别为 m 和 n。试用类 PASCAL 语言编写时间复杂度为 O（m+n）的算法将矩阵 B 加到矩阵 A 上去。A 的空间足够大，不另加辅助空间。要求描述所用结构。

【北京工业大学 1997 三（10 分）】

3. 设整数 $x_1, x_2, \cdots, x_N$ 已存放在数组 A 中，编写一 PASCAL 递归过程，输出从这 n 个数中取出所有 k 个数的所有组合（k<=n）。例：若 A 中存放的数是 1，2，3，4，5，k 为 3，则输出结果应为：543，542，541，532，531，521，432，431，421，321。 【山东大学 1992 五（13 分）】

类似本题的另外叙述有：

（1）题目基本相同，只是叙述不同，要求用 PASCAL 语言。【东南大学 2001 三 （10 分）】

4. 编写一个过程，对一个 n×n 矩阵，通过行变换，使其每行元素的平均值按递增顺序排列。

【中科院软件所 1996 】

5. 设原来将 N 个自然数 1,2,…,N 按某个顺序存于数组 A 中，经过下面的语句计算，使 A[I]的值变为 A[1]到 A[I-1]中小于原 A[I]值的个数。

```
FOR  I:=N  DOWNTO  1  DO
      BEGIN
        C：=0;
        FOR  J：=1  TO  I-1   DO
          IF  A[J]<A[I]   THEN   C：=C+1;
        A[I]：=C
      END;
```

编程将经过上述处理后的 A 还原成原来的 A。 【上海大学 1996 二 （16 分）】

6. 请编写完整的程序。如果矩阵 A 中存在这样的一个元素 A[i,j]满足条件:A[i,j]是第 i 行中值最小的元素,且又是第 j 列中值最大的元素，则称之为该矩阵的一个马鞍点。请编程

计算出 m*n 的矩阵 A 的所有马鞍点。【上海大学 2000 三（20 分）】【中科院自动化所 1997】

7. 给定一个整数数组 b[0..N-1]，b 中连续的相等元素构成的子序列称为平台。试设计算法，求出 b 中最长平台的长度。【中科院计算所 1999 五、2 （20 分）】

8. 给定 nxm 矩阵 A[a..b,c..d]，并设 A[i,j]≤A[i,j+1](a≤i≤b,c≤j≤d-1)和 A[i,j]≤A[i+1,j](a≤i≤b-1,c≤j≤d)。设计一算法判定 X 的值是否在 A 中，要求时间复杂度为 O(m+n)。【东南大学 2001 六(13 分)】

类似本题的另外叙述有：

（1）给定整型数组 B[0..m,0..n] 。已知 B 中数据在每一维方向上都按从小到大的次序排列，且整型变量 x 在 B 中存在。试设计一个程序段找出一对满足 B[i,j]=x 的(i,j)值，要求比较次数不超过 m+n.。

【清华大学 1998 六（10 分）】

(2) 给定 n×m 矩阵 A[a..b,c..d]，并设 A[i,j]<=A[i,j+1]（a<=i<=b,c<=j<=d-1）知 A[i,j]<=A[i+1,j],(a<=i<=b-1, c<=j<=d)。设计一算法以比 O(n*m)小的最坏时间复杂性判定值 x 是否在 A 中。【东南大学 1994 三(17 分)】

9. 编写算法，将自然数 1～n 按“蛇形”填入 n×n 矩阵中。例（1～4 ）如图所示：（用程序实现）

1	3	4	10
2	5	9	11
6	8	12	15
7	13	14	16

【南京航空航天大学 1997 八 （12 分）】 【中科院计算所 1996】

10. 设二维数组 a[1..m, 1..n] 含有 m*n 个整数。

(1) 写出算法(pascal 过程或 c 函数)：判断 a 中所有元素是否互不相同?输出相关信息(yes/no)；

(2) 试分析算法的时间复杂度。【华中理工大学 1999 五 （10 分）】

11. 二项式$(a+b)^n$展开式的系数为

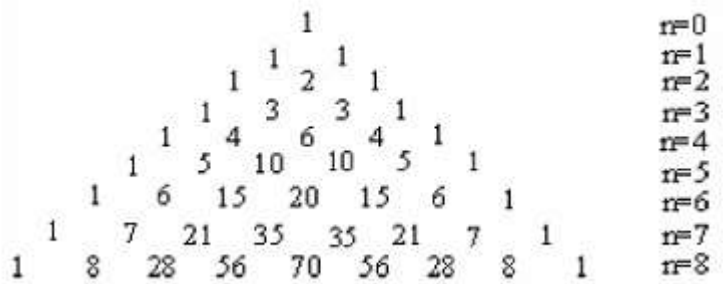
C（n，0）=1，C（n，n）=1，对于 n>=0

C（n，k）=C（n-1，k）+C（n-1，k-1），对于 0<k<n 形成著名的杨辉三角形，如图所示。

(1)试写一个递归算法，根据以上公式生成 C（n,k)。(6 分)

(2)试画出计算 C（6，4）的递归树。(4 分)

(3)试写一个非递归算法，既不用数组也不用栈，对于任意的 0<=k<=n 计算 C(n,k)（6 分）



【清华大学 1999 五 （16 分）】

12. 设任意 n 个整数存放于数组 A(1:n)中，试编写程序，将所有正数排在所有负数前面（要求算法复杂性为 0(n)）。 【山东大学 1993 三 (12 分)】.

类似本题的另外叙述有：

（1）已知数组 A[1..n]的元素类型为整型，设计算法调整 A，使其左边的所有元素小于零，右边的所有元素大于等于零。（要求算法的时间复杂度和空间复杂度均为 0（n））

【北京理工大学 2000 四、1 （4 分）】

（2）设计一个算法，把整数数组中所有的偶数放到所有的奇数之前。要求时间、空间效率尽可能高。

【华南师范大学 1999 六、1 (10 分)】

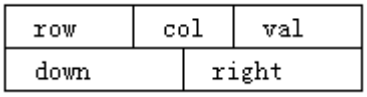
（3）设一系列正整数存放在一个数组中，试设计算法，将所有奇数存放在数组的前半部分，将所有的偶数存放在数组的后半部分。要求尽可能少用临时存储单元并使时间最少。请试着分析你实现的算法的时间复杂度和空间复杂度。 【南开大学 2000 三、2】

（4）设计算法将数组 A[1..n]调整为左右两部分，使的左边所有的元素小于右边的所有元素，并给出这一划分的分界位置。要求算法的时间复度为 O(n)。【合肥工业大学 2001 五、3 （8 分）】

13. 若 S 是 n 个元素的集合，则 S 的幂集 P(S)定义为 S 所有子集的集合。例如，S=(a,b,c),P(S)={() ,(a),(b),(c),(a,b),(a,c),(b,c),(a,b,c)}给定 S，写一递归算法求 P(S)。

【东南大学 1993 五 （15 分）】【东南大学 1997 五 （15 分）】

14. 编写算法打印出由指针 Hm 指向总表头的以十字链表形式存储的稀疏矩阵中每一行的非零元的个数。注意：行、列及总表头结点的形式为：



它们已用 val 域链接成循环链表。非零元的结点形式也同上，每一行（列）的非零元由 right（down）域把它们链接成循环链表，该行（列）的表头结点即为该行（列）循环链表的表头。

【上海大学 1998 五 （16 分）】

15. 试编写建立广义表存储结构的算法，要求在输入广义表的同时实现判断、建立。设广义表按如下形式输入（$a_1, a_2, a_3, \cdots, a_n$） n>=0，其中 a_i 或为单字母表示的原子或为广义表，n=0 时为只含空格字符的空表。(注:算法可用类 pascal 或类 c 书写) 【北京工业大学 1998 十 (15 分)】

16. 广义表是 n(n>=0)个数据元素 $a_1, a_2, a_3, \cdots, a_n$ 的有限序列。其中 a_i (1<=i<=n)或者是单个数据元素（原子），或仍然是一个广义表。广义表的结点具有不同的结构，即原子结点和子表元素结点，为了将两者统一，用了一个标志 tag，当其为 0 时表示是原子结点，其 data 域存储结点值，link 域指向下一个结点，当其 tag 为 1 时表示是子表结点，其 sublist 为指向子表的指针。因此，广义表可采用如下结构存储：

```
TYPE glist=^gnode;
     gnode=RECORD
             link:glist;
             CASE tag:0..1 OF
                0:(data:char);
                1:(sublist:glist)
           END;
```

（1）画出广义表((a,b),c)的存储结构；

（2）写出计算一个广义表的原子结点个数的递归算法表示式；

（3）编写实现上述算法的过程或函数程序。【厦门大学 2002 三 （12 分）】

17. 广义表 GL=(a_1,a_2 ,…,a_n),其中 a_k(k=1,2,...,n)或是单个数据元素（原子），或仍然是个广义表。给定如下有关广义表的类型定义：

```
TYPE tagtype=0..1;
     glist=^gnode;
     gnode=RECORD
             link:glist;            {link 域指向下一个结点}
             CASE tag:tagtype OF   {tag=0 表示原子结点}
                0: (data :integer);
                1:(sublist: glist)
           END;
```

编写一个过程或函数计算一个广义表的所有原子结点数据域之和，例如对广义表(3,(2,4,5),(6,3）) 数据域之和为 23。【厦门大学 2000 四、2 （9 分）】

18. 数组 H[1：1000] 中存放着 1000 个大小不同的正整数；

（1） 选择一分类算法使能最快地得到其中 10 个最大的数，简要说明理由；

（2） 编写一程序 seek() ，执行该程序时，在命令行中提供二个参数；

seek a n<enter> 表示需打印 H[ ]中 n 个最大数。

seek I n<enter> 表示需打印 H[ ]中 n 个最小数。 【浙江大学 1994 八 (18 分)】

19. 已知两个定长数组，它们分别存放两个非降序有序序列，请编写程序把第二个数组序列中的数逐个插入到前一个数组序列中，完成后两个数组中的数分别有序（非降序）并且第一数组中所有的数都不大于第二个数组中的任意一个数。注意，不能另开辟数组，也不能对任意一个数组进行排序操作。例如，

第一个数组为：4,12,28

第二个数组为：1,7,9,29,45

输出结果为：1,4,7----------第一个数组

9,12,28,29,45---------第二个数组 【上海大学 1998 四 （20 分)】

20. 设数组 A[1..n]中，A[n-2k+1..n-k]和[n-k+1..n]中元素各自从小到大排好序，试设计一个算法使 A[n-2k+1..n]按从小到大次序排好序。并分析算法所需的计算时间。【福州大学 1998 四、3 (10 分)】

21. 设 A[1..100]是一个记录构成的数组，B[1..100]是一个整数数组，其值介于 1 至 100 之间，现要求按 B[1..100]的内容调整 A 中记录的次序，比如当 B[1]=11 时，则要求将 A[1]的内容调整到 A[11]中去。规定可使用的附加空间为 O(1)。【中科院计算所 2000 七(15 分)】

22. 给定有 m 个整数的递增有序数组 a[1..m]和有 n 个整数的递减有序数组 b[1..n]，试写出算法:将数组 a 和 b 归并为递增有序数组 c[1..m+n]。(要求:算法的时间复杂度为 O(m+n))

【华中理工大学 2000 八、1（10 分)】

23. 在数组 A[1..n]中有 n 个数据，试建立一个带有头结点的循环链表，头指针为 h，要求链中数据从小到大排列，重复的数据在链中只保存一个。【南京理工大学 1998 七、2 （7 分)】

第五章 数组和广义表

一、选择题

1.B	2.1L	2.2J	2.3C	2.4I	2.5C	3.B	4.B	5.A	6.1H	6.2C	6.3E
6.4A	6.5F	7.B	8.1E	8.2A	8.3B	9.B	10.B	11.B	12.B	13.A	14.B
15.B	16.A	17.C	18.D	19.C	20.D	21.F	22.C	23.D	24.C	25.A	26.C
27.A											

二、判断题

1. ×	2. √	3. √	4. ×	5. ×	6. ×	7. √	8. ×	9. ×	10. ×	11. ×	12. √
13. √	14. √										

部分答案解释如下。

1. 错误。对于完全二叉树，用一维数组作存储结构是效率高的（存储密度大）。

4. 错误。数组是具有相同性质的数据元素的集合，数据元素不仅有值，还有下标。因此，可以说数祖是元素值和下标构成的偶对的有穷集合。

5. 错误。数组在维数和界偶确定后，其元素个数已经确定，不能进行插入和删除运算。

6. 错误。稀疏矩阵转置后，除行列下标及行列数互换外，还必须确定该元素转置后在新三元组中的位置。

8. 错误。广义表的取表尾运算，是非空广义表除去表头元素，剩余元素组成的表，不可能是原子。

9. 错误。广义表的表头就是广义表的第一个元素。只有非空广义表才能取表头。

10. 错误。广义表中元素可以是原子，也可以是表（包括空表和非空表）。

11. 错误。广义表的表尾，指去掉表头元素后，剩余元素所组成的表。

三、填空题

1. 顺序存储结构　　2.（1）9572（2）1228　　3.（1）9174（2）8788　　4. 1100

5. 1164 公式：$LOC(a_{ijk})=LOC(a_{000})+[v2*v3*(i-c_1)+v3*(j-c_2)+(k-c_3)]*l$ （l 为每个元素所占单元数）

6. 232　　7. 1340　　8. 1196　　9. 第 1 行第 3 列

10. (1)270 (2)27 (3)2204　　11. i(i-1)/2+j (1<=i,j<=n)

12. (1)n(n+1)/2 (2)i(i+1)/2 (或 j(j+1)/2) (3)i(i-1)/2+j (4)j(j-1)/2+i (1<=i,j<=n)

13. 1038 三对角矩阵按行存储：k=2(i-1)+j (1<=i,j<=n)

14. 33 (k=i(i-1)/2+j) (1<=i,j<=n)

15. 非零元很少(t<<m*n)且分布没有规律　　16. 节省存储空间。

17. 上三角矩阵中，主对角线上第 r(1≤r≤n) 行有 n-r+1 个元素，a_{ij} 所在行的元素数是 j-i+1 。所以，元素在一维数组的下标 k 和二维数组下标关系:k=((i-1)*(2n-i+2))/2+(j-i+1)=(i-1)(2n-i)/2+j (i≤j)

18. 93　　19. i(i-1)/2+j　　20. 线性表　　21. 其余元素组成的表

22. （1） 原子（单元素）是结构上不可再分的，可以是一个数或一个结构；而表带结构，本质就是广义表，因作为广义表的元素故称为子表。

（2）大写字母 （3）小写字母 （4）表中元素的个数（5）表展开后所含括号的层数

23. 深度　　24.（1）（） （2）(()) （3）2 （4）2

25. head（head（tail（tail（head（tail（tail（A）））））））
26. 表展开后所含括号的层数　　27.（1）5　（2）3
28. head(head(tail(LS)))　　29. head(tail(tail(head(tail(head(A))))))
30. head(tail(head(tail(H))))　31. (b)　32. (x,y,z)　33. (d,e)
34. GetHead(GetHead(GetTail(L)))
35. 本算法中，首先数组 b 中元素以逆置顺序放入 d 数组中，然后数组 a 和数组 d 的元素比较，将大者拷贝到数组 c。第一个 WHILE 循环到数组 a 或数组 d 结尾，第二个和第三个 WHILE 语句只能执行其中的一个。
（1）b[m-i+1]（2）x:=a[i]（3）i:=i+1（4）x:=d[j]（5）j:=j+1　（6）k:=k+1（7）i<=l（8）j<=m
36.（1）(i==k) **return**（2）i+1（3）i-1（4）i!=k
本算法利用快速排序思想，找到第 k 个元素的位置（下标 k-1 因而开初有 k--）。内层 do 循环以 t(t=a[low])为“枢轴”找到其应在 i 位置。这时若 i 等于 k，则算法结束。(即第一个空格处 if(i==k) **return**)。否则，若 i<k，就在 i+1 至 high 中间去查；若 i>k，则在 low 到 i-1 间去找，直到找到 i=k 为止。
37.逆置广义表的递归模型如下

f(LS)=

上面 app**END**(a,b)功能是将广义表 a 和 b 作为元素的广义表连接起来。
（1）(p->tag==0)　　//处理原子
（2）h=reverse(p->val.ptr.hp)　//处理表头
（3）(p->val.ptr.tp)　　//产生表尾的逆置广义表
（4）s->val.ptr.tp=t;　　//连接
（5）q->val.ptr.hp=h　　//头结点指向广义表
38. 本题要求将 1，2，...,n*n 个自然数，按蛇型方式存放在二位数组 A[n][n]中。“蛇型”方式，即是按“副对角线”平行的各对角线，从左下到右上，再从右上到左下，存放 n^2 个整数。对角线共 2n-1 条，在副对角线上方的对角线，题目中用 k 表示第 k 条对角线（最左上角 k=1），数组元素 x 和 y 方向坐标之和为 k+1（即题目中的 i+j=k+1）。副对角线下方第 k 条对角线与第 2n-k 条对角线对称，其元素的下标等于其对称元素的相应坐标各加(k-n)。
（1）k<=2*n-1 //共填 2*n-1 条对角线
（2）q=2*n-k　//副对角线以下的各条对角线上的元素数
（3）k%2!=0　//k 为偶数时从右上到左下，否则从左下向右上填数。(本处计算下标 i 和 j)
（4）k>=n　　//修改副对角线下方的下标 i 和 j。
（5）m++; 或 m=m+1 //为填下个数作准备，m 变化范围 1..n*n。
本题解法的另一种思路见本章算法设计题第 9 题。
39. 本题难点有二：一是如何求下一出圈人的位置，二是某人出圈后对该人的位置如何处理。
　　按题中要求，从第 s 个人开始报数，报到第 m 个人，此人出圈。n 个人围成一圈，可看作环状，则下个出圈人，其位置是(s+m-1)%n。n 是人数，是个变量，出圈一人减 1，算法中用 i 表示。对第二个问题，算法中用出圈人后面人的位置依次前移，并把出圈人的位置（下

标）存放到当时最后一个人的位置（下标）。算法最后打印出圈人的顺序。
（1）(s+m-1) MOD i //计算出圈人 s1
（2）s1:=i //若 s1=0,说明是第 i 个人出圈（i%i=0）
（3）s1 TO i-1 //从 s1 到 i 依次前移，使人数减 1，并将出圈人放到当前最后一个位置 A[i]=w。
40. 若第 n 件物品能放入背包，则问题变为能否再从 n-1 件物品中选出若干件放入背包（这时背包可放入物品的重量变为 s-w[n]）。若第 n 件物品不能放入背包，则考虑从 n-1 件物品选若干件放入背包（这时背包可放入物品仍为 s）。若最终 s=0,则有一解；否则，若 s<0 或虽然 s>0 但物品数 n<1,则无解。
（1）s-w[n],n-1 //Knap(s-w[n],n-1)=true
（2）s,n-1 // Knap←Knap(s,n-1)

四、应用题
1、958 三维数组以行为主序存储，其元素地址公式为：

$$LOC(A_{ijk})=LOC(A_{c_1c_2c_3})+[(i-c_1)V_2V_3+(j-c_2)V_3+(k-c_3)]*L+1$$

其中 c_i,d_i 是各维的下界和上界，$V_i=d_i-c_i+1$ 是各维元素个数，L 是一个元素所占的存储单元数。
2. b 对角矩阵的 b 条对角线，在主对角线上方和下方各有⌊b/2⌋条对角线(为叙述方便，下面设 a=⌊b/2⌋)。从第 1 行至第 a 行，每行上的元素数依次是 a+1,a+2,...,b-2,b-1,最后的 a 行上的元素个数是 b-1，b-2,...,a+1。中间的(n-2a)行，每行元素个数都是 b。故 b 条对角线上元素个数为 (n-2a)b+a*(a+b)。存放在一维数组 V[1..nb-a(b-a)]中，其下标 k 与元素在二维数组中下标 i 和 j 的关系为：

k=

3. 每个元素 32 个二进制位，主存字长 16 位，故每个元素占 2 个字长，行下标可平移至 1 到 11。
（1）242 （2）22 （3）s+182 （4）s+142
4. 1784 (公式：$Loc(A_{ijkl})=100$(基地址)$+[(i-c_1)v_2v_3v_4+(j-c_2)v_3v_4+(k-c_3)v_4+(l-c_4)]*4$
5. 1210+108L (LOC(A[1,3,-2])=1210+$[(k-c_3)v_2v_1+(j-c_2)v_1+(i-c_1)]$*L（设每个元素占 L 个存储单元）
6. 数组占的存储字节数=10*9*7*4=2520;A[5,0,7]的存储地址=100+[4*9*7+2*7+5]*4=1184
7. 五对角矩阵按行存储，元素在一维数组中下标（从 1 开始）k 与 i,j 的关系如下:

k=
A[15,16]是第 71 个元素，在向量[-10:m]中的存储位置是 60 。
8.（1）540 （2）108 （3）i=3,j=10，即 A[3,10] 9. k=i(i-1)/2+j
10. 稀疏矩阵 A 有 t 个非零元素，加上行数 mu、列数 nu 和非零元素个数 tu(也算一个三元组)，共占用三元组表 LTMA 的 3(t+1)个存储单元，用二维数组存储时占用 m*n 个单元，只

有当 3(t+1)<m*n 时，用 LTMA 表示 A 才有意义。解不等式得 t<m*n/3-1。

11．参见 10。

12. 题中矩阵非零元素用三元组表存储，查找某非零元素时，按常规要从第一个元素开始查找，属于顺序查找，时间复杂度为 O(n)。若使查找时间得到改善，可以建立索引，将各行行号及各行第一个非零元素在数组 B 中的位置（下标）偶对放入一向量 C 中。若查找非零元素，可先在数组 C 中用折半查找到该非零元素的行号，并取出该行第一个非零元素在 B 中的位置，再到 B 中顺序（或折半）查找该元素，这时时间复杂度为 O(logn)。

13.（1）176　　（2）76 和 108　　（3）28 和 116。

14.（1）k = 3(i-1)　(主对角线左下角，即 i=j+1)

k = 3(i-1)+1　(主对角线上，即 i=j)

k = 3(i-1)+2　(主对角线上，即 i=j-1)

由以上三式，得 k=2(i-1)+j　　(1≤i,j≤n; 1≤k≤3n-2)

（2）$10^3*10^3-(3*10^3-2)$

15. 稀疏矩阵 A 采用二维数组存储时，需要 n*n 个存储单元，完成求 Σa_{ii}(1<=i<=n)时，由于 a[i][i]随机存取，速度快。但采用三元组表时，若非零元素个数为 t，需 3(t+1)个存储单元（第一个分量中存稀疏矩阵 A 的行数，列数和非零元素个数，以后 t 个分量存各非零元素的行值、列值、元素值），比二维数组节省存储单元；但在求 Σa_{ii}(1<=i<=n)时，要扫描整个三元组表，以便找到行列值相等的非零元素求和，其时间性能比采用二维数组时差。

16. 特殊矩阵指值相同的元素或零元素在矩阵中的分布有一定规律，因此可以对非零元素分配单元（对值相同元素只分配一个单元），将非零元素存储在向量中，元素的下标 i 和 j 和该元素在向量中的下标有一定规律，可以用简单公式表示，仍具有随机存取功能。而稀疏矩阵是指非零元素和矩阵容量相比很小（t<<m*n），且分布没有规律。用十字链表作存储结构自然失去了随机存取的功能。即使用三元组表的顺序存储结构，存取下标为 i 和 j 的元素时，要扫描三元组表，下标不同的元素，存取时间也不同，最好情况下存取时间为 O(1)，最差情况下是 O(n)，因此也失去了随机存取的功能。

17．一维数组属于特殊的顺序表，和有序表的差别主要在于有序表中元素按值排序（非递增或非递减），而一维数组中元素没有按元素值排列顺序的要求。

18．n(n+1)/2（压缩存储）　或 n^2（不采用压缩存储）

19．LOC（A[i,j]）=LOC（A[3，2]）+[（i-3）*5+（j-2）]×2　（按行存放）

LOC（A[i,j]）=LOC（A[3，2]）+[（j-2）*6+（i-3）]×2　（按列存放）

20．n 阶下三角矩阵元素 A[i][j]（1<=i,j<=n,i>=j)。第 1 列有 n 个元素，第 j 列有 n-j+1 个元素，第 1 列到第 j-1 列是等腰梯形，元素数为(n+(n-j+2)(j-1)/2，而 a_{ij} 在第 j 列上的位置是为 i-j+1。所以 n 阶下三角矩阵 A 按列存储，其元素 a_{ij} 在一维数组 B 中的存储位置 k 与 i 和 j 的关系为：

k=(n+(n-(j-1)+1)(j-1)/2+(i-j+1)=(2n-j)(j-1)/2+i

21．三对角矩阵第一行和最后一行各有两个非零元素，其余每行均有三个非零元素，所以共有 3n-2 个元素。

（1）主对角线左下对角线上的元素下标间有 i=j+1 关系，k 与 i 和 j 的关系为 k=3(i-1)；主对角线上元素下标间有关系 i=j，k 与 i 和 j 的关系为 k=3(i-1)+1；主对角线右上那条对角线上元素下标间有关系 i=j-1，k 与 i 和 j 的关系为 k=3(i-1)+2。综合以上三等式，有 k=2(i-1)+j　(1<=i,j<=n, |i-j|<=1)

（2）i=k/3+1；（1≤k≤3n-2）　// k/3 取小于 k/3 的最大整数。下同

j=k-2(i-1)=k-2(k/3)=k%3+k/3

22．这是一个递归调用问题，运行结果为：DBHEAIFJCKGL

23. （1）FOR 循环中，每次执行 PerfectShuffle(A,N)和 CompareExchange(A,N)的结果：

第 1 次：A[1..8]=[90,30,85,65,50,80,10,100]

A[1..8]=[30,90,65,85,50,80,10,100]

第 2 次：A[1..8]=[30,50,90,80,65,10,85,100]

A[1..8]=[30,50,80,90,10,65,85,100]

第 3 次：A[1..8]=[30,10,50,65,80,85,90,100]

A[1..8]=[10,30,50,65,80,85,90,100]

（2）Demo 的功能是将数组 A 中元素按递增顺序排序。

（3）PerfectShuffle 中 WHILE 循环内是赋值语句，共 2N 次，WHILE 外成组赋值语句，相当 2N 个简单赋值语句；CompareExchange 中 WHILE 循环内是交换语句，最好情况下不发生交换，最差情况下发生 N 次交换，相当于 3N 个赋值语句；Demo 中 FOR 循环循环次数 $\log_2 2N$，故按赋值语句次数计算 Demo 的时间复杂度为：最好情况：$O(4N*\log_2 2N)\approx O(N\log(2*N))$；最差情况：$O((4N+3N)*\log_2 2N)\approx O(N\log(2*N))$。

24. 这是一个排序程序。运行后 B 数组存放 A 数组各数在排序后的位置。结果是：

A={121,22,323,212,636,939,828,424,55,262}

B={3,1,6,4,8,10,9,7,2,5}

C={22,55,121,212,262,323,424,639,828,939}

25. （1）c=　　（2）a=

26. （1）同上面 26 题（1）

（2）对 c 数组的赋值同所选择的下标 i 和 j 的次序（指外层循环用 j 内层用 i）没有关系

（3）同上题 26（2）

（4）对 i,j 下标取反序后，重复执行第（3）步，A 数组所有元素均变为 2。（在机器上验证，反复循环 3 次后，所有元素均变为 2）

27. 错误有以下几处：

（1）过程参数没有类型说明；　（2）出错条件判断：缺少 OR（i+k>last+1）；

（3）删除元素时 FOR 循环应正向，不应用反向 **DOWNTO**；　（4）count 没定义；

低效体现在两处：

（1）删除 k 个元素时，不必一个一个元素前移，而应一次前移 k 个位置；

（2）last 指针不应一次减 1，而应最后一次减 k。

正确的高效算法如下：

```
const  m=64;
TYPE ARR=ARRAY[1..m] OF integer;
PROCEDURE delk（VAR A:ARR; VAR last:integer;i,k: integer);
 {从数组 A[1..last]中删除第 i 个元素起的 k 个元素，m 为 A 的上限}
VAR count: integer;
BEGIN
  IF（i<0）OR（i>last）OR（k<0）OR（last>m）OR（i+k>last+1）
    THEN write（' error'）
    ELSE[FOR count: = i+k TO last DO  A[count-k]: =A[count];
         last: =last-k; ]
```

END；

28. 这是计数排序程序。

（a）c[i](1<=i<=n)中存放 A 数组中值为 i 的元素个数。

（b）c[i](1<=i<=n)中存放 A 数组中小于等于 i 的个数。

（c）B 中存放排序结果，B[1..n]已经有序。

（d）算法中有 4 个并列 for 循环语句，算法的时间复杂度为 O(n)。

29. 上三角矩阵第一行有 n 个元素，第 i-1 行有 n-（i-1）+1 个元素，第一行到第 i-1 行是等腰梯形，而第 i 行上第 j 个元素（即 a_{ij}）是第 i 行上第 j-i+1 个元素，故元素 A_{ij}在一维数组中的存储位置（下标 k）为：

k=(n+(n-(i-1)+1))(i-1)/2+(j-i+1)=(2n-i+2)(i-1)/2+j-i+1

30. 将上面 29 题的等式进一步整理为：

k=（n+1/2）i-i^2/2+j-n，

则得 f_1（i）=（n+1/2）i-i^2/2，f_2（j）=j，c=-n。

31.（1）将对称矩阵对角线及以下元素按行序存入一维数组中，结果如下：

（2）因行列表头的“行列域”值用了 0 和 0，下面十字链表中行和列下标均从 1 开始。

注：上侧列表头 H_i 和左侧行表头 H_i 是一个（即 H_1、H_2、H3 和 H_4），为了清楚，画成了两个。

32.（1）k=(2n-j+2)(j-1)/2+i-j+1　　（当 i≥j 时，本题 n=4）

k=(2n-i+2)(i-1)/2+j-i+1　　（当 i<j 时，本题 n=4）

（2）稀疏矩阵的三元组表为：s=((4,4,6),(1,1,1),(1,4,2),(2,2,3),(3,4,5),(4,1,2),(4,3,5)。其中第一个三元组是稀疏矩阵行数、列数和非零元素个数。其它三元组均为非零元素行值、列值和元素值。

33.（1）k=2（i-1）+j　（1≤i，j≤n，|i-j|≤1）

i=floor（k/3）+1　// floor（a）是取小于等于 a 的最大整数

j=k-2（i-1）

推导过程见上面第 25 题。

（2）行逻辑链接顺序表是稀梳矩阵压缩存储的一种形式。为了随机存取任意一行的非零元，需要知道每一行第一个非零元在三元组表中的位置。为此，除非零元的三元组表外，还需要一个向量，其元素值是每行第一个非零元在三元组表中的位置。其类型定义如下：

```
typedef struct
  { int mu,nu,tu;          //稀梳矩阵的行数、列数和非零元素个数
    int rpos[maxrow+1];  //行向量，其元素值是每行第一个非零元在三元组表中的位置。

    Triple data[maxsize];
  }SparsMatrix;
```

因篇幅所限，不再画出行逻辑链接顺序表。

34. 各维的元素数为 d_i-c_i+1，则 $a[i_1, i_2, i_3]$的地址为：

$a_0+[(i_1-c_1)(d_3-c_3+1)(d_2-c_2+1)+(i_2-c_2)(d_2-c_2+1)+(i_3-c_3)]*L$

35. 主对角线上元素的坐标是 i=j，副对角线上元素的坐标 i 和 j 有 i+j=n+1 的关系

（1）i=j 或 i=n+1-j　（1≤i，j≤n）

（2）非零元素分布在两条主、副对角线上，除对角线相交处一个元素（下称“中心元素”）外，其余每行都有两个元素。主对角线上的元素，在向量 B 中存储的下标是 k=2i-1(i=j, 1≤i，j≤n，1<=k<=2n-1)。

副对角线上的元素，在中心元素前，在向量 B 中存储的下标是 k=2i(i<>j, 1≤i，j≤n/2)；在中心元素后，其下标是 k=2(i-1)(i<>j，n/2+1≤i，j≤n, 1<=k<=2n-1)。

（3）a_{ij}在 B 中的位置 K=

36. 由于对称矩阵采用压缩存储，上三角矩阵第一列一个元素，第二列两个元素，第 j 列 j 个元素。上三角矩阵共有 n (n+1)/2 个元素。我们将这些元素存储到一个向量 B[n(n+1)/2+1]中。可以看到 B[k]和矩阵中的元素 a_{ij}之间存在着一一对应关系：

则其对应关系可表示为： k=　　(1<=i,j<=n, 1<=k<=n(n+1)/2)

```
int MAX(int x,int y)
  { return(x>y?x:y);
  }
int MIN(int x,int y)
  { return(x<y?x:y);
  }
```

37. 设用 mu,nu 和 tu 表示稀疏矩阵行数，列数和非零元素个数，则转置矩阵的行数，列数和非零元素的个数分别是 nu,mu 和 tu。转置可按转置矩阵的三元组表中的元素顺序进行，

即按稀疏矩阵的列序，从第 1 列到第 nu 列，每列中按行值递增顺序，找出非零元素，逐个放入转置矩阵的三元组表中，转时行列值互换，元素值复制。按这种方法，第 1 列到第 1 个非零元素一定是转置后矩阵的三元组表中的第 1 个元素，第 1 列非零元素在第 2 列非零元素的前面。这种方法时间复杂度是 O(n*P)，其中 p 是非零元素个数，当 p 和 m*n 同量级时，时间复杂度为 $O(n^3)$。

另一种转置方法称作快速转置，使时间复杂度降为 O(m*n)。它是按稀疏矩阵三元组表中元素的顺序进行。按顺序取出一个元素，放到转置矩阵三元组表的相应位置。这就要求出每列非零元素个数和每列第一个非零元素在转置矩阵三元组表中的位置，设置了两个附加向量。

38. 广义表中的元素，可以是原子，也可以是子表，即广义表是原子或子表的有限序列，满足线性结构的特性：在非空线性结构中，只有一个称为“第一个”的元素，只有一个成为“最后一个”的元素，第一元素有后继而没有前驱，最后一个元素有前驱而没有后继，其余每个元素有唯一前驱和唯一后继。从这个意义上说，广义表属于线性结构。

39. 数组是具有相同性质的数据元素的集合，同时每个元素又有唯一下标限定，可以说数组是值和下标偶对的有限集合。n 维数组中的每个元素，处于 n 个关系之中，每个关系都是线性的，且 n 维数组可以看作其元素是 n-1 维数组的一个线性表。而广义表与线性表的关系，见上面 38 题的解释。

40. 线性表中的元素可以是各种各样的，但必须具有相同性质，属于同一数据对象。广义表中的元素可以是原子，也可以是子表。其它请参见 38

41. (1)(c,d)　(2)(b)　(3) b　(4)(f)　(5)()

42. Head (Tail (Head (Head (L1))))

Head (Head (Head (Tail (Head (Tail (L2))))))

类似本题的另外叙述的几个题解答如下：

(1) head (head (tail (tail (L)))), 设 L= (a, (c), b), (((e))))

(2) head (head (head (head (tail (tail (L))))))

(3)head(tail(head(tail(A))))

(4) H (H (T (H (T (H (T (L)))))))

(5) tail (L) = (((c,d)), (e,f))

head (tail (L)) = ((c,d))

head (head (tail (L))) = (c,d)

tail (head (head (tail (L)))) = (d)

head (tail (head (head (tail (L))))) =d

(6) head (tail (head (head (tail (tail (A))))))

43. 广义表的第一种存储结构的理论基础是，非空广义表可唯一分解成表头和表尾两部分，而由表头和表尾可唯一构成一个广义表。这种存储结构中，原子和表采用不同的结点结构

（“异构”，即结点域个数不同）。

原子结点两个域：标志域 tag=0 表示原子结点，域 DATA 表示原子的值；子表结点三个域：tag=1 表示子表，hp 和 tp 分别是指向表头和表尾的指针。在画存储结构时，对非空广义表不断进行表头和表尾的分解，表头可以是原子，也可以是子表，而表尾一定是表（包括空表）。上面是本题的第一种存储结构图。

广义表的第二种存储结构的理论基础是，非空广义表最高层元素间具有逻辑关系：第一个元素无前驱有后继，最后一个元素无后继有前驱，其余元素有唯一前驱和唯一后继。有人将这种结构看作扩充线性结构。这种存储结构中，原子和表均采用三个域的结点结构（“同构”）。结点中都有一个指针域指向后继结点。原子结点中还包括标志域 tag=0 和原子值域 DATA；子表结点还包括标志域 tag=1 和指向子表的指针 hp。在画存储结构时，从左往右一个元素一个元素的画，直至最后一个元素。下面是本题的第二种存储结构图。

由于存储结构图占篇幅较大，下面这类题均不再解答。

44.深度为 5，长度为 2

45.（1）略

（2）表的长度为 5，深度为 4

（3）head（tail（head（head（head（tail（tail（tail（tail（A））））））））））

46.共享结构广义表 A=（((b,c)，d)，(a)，((a)，((b,c)，d))，e，()）的存储表示：

47.（1）算法 A 的功能是逆置广义表 p（即广义表由 p 指针所指）。逆置后的广义表由 t 指向。

（2）逆置后的广义表由 t 指向，这时 p=nil。

48. (a,b)　　49. (d)

50. 否。广义表的长度不是广义表中原子个数，而是指广义表中所含元素的个数，广义表中的元素可以是原子，也可以是子表。广义表元素多于 1 个时，元素间用逗号分开。

51. $p(x_1\ x_2\ x_3)=2\ x_1^5\ x_2^2\ x_3^4+5\ x_1^5\ x_2^3\ x_3^3+3\ x_1\ x_2^4\ x_3^2+(x_1^5\ x_2^3+\ x_2)\ x_3+6$

52. (1) $H(A(a_1,a_2),B(b_1),C(c_1,\ c_2),x)$

HEAD(TAIL(HEAD(H))) $=a_2$

(2) 略

五.算法设计题

1.[题目分析]本题是在向量 D 内插入元素问题。首先要查找插入位置，数据 x 插入到第 i 个数据组的末尾，即是第 i+1 个数据组的开始，而第 i（1≤i≤n）个数据组的首地址由数组 s（即数组元素 s[i]）给出。其次，数据 x 插入后，还要维护数组 s，以保持空间区 D 和数组 s 的正确的相互关系。

```
void Insert (int s[], datatype D[], x, int i,m)
//在 m 个元素的 D 数据区的第 i 个数据组末尾，插入新数据 x，第 i 个数据组的首址由数组 s 给出。
 {if (i<1 || i>n) {printf ("参数错误"); exit ( 0 );}
  if (i==n)  D[m]=x;    // 在第 n 个数据组末尾插入元素。
  else {for (j=m-1;j>=s[i+1];j--) D[j+1]=D[j]; // 第 i＋1 个数据组及以后元素后移
        D[s[i+1]]=x;  // 将新数据 x 插入
        for(j=i+1;j<=n;j++) s[j]++;  // 维护空间区 D 和数组 s 的的关系。
       }  //结束元素插入
  m++;     //空间区 D 的数据元素个数增 1。
  }// 算法 Insert 结束
```

[算法讨论] 数据在空间区从下标 0 开始，最后一个元素的下标是 m-1。设空间区容量足够大，未考虑空间溢出问题。数组 s 随机存数，而向量 D 数据插入，引起数组元素移动，时间复杂度是 O（n）。

2. [题目分析]设稀疏矩阵的非零元素的三元组以行序为主存储在三元组表中。矩阵的相加是对应元素的相加。对两非零元素相加，若行号不等，则行号大者是结果矩阵中的非零元素。

若行号相同，则列号大者是结果中一非零元素；若行号列号相同，若对应元素值之和为零，不予存储，否则，作为新三元组存到三元组表中。题目中要求时间复杂度为 O（m+n）。因此需从两个三元组表的最后一个元素开始相加。第一个非零元素放在 A 矩阵三元组表的第 m+n 位置上。结果的三元组至多是 m+n 个非零元素。最后若发生对应元素相加和为零的情况，对三元组表中元素要进行整理，以便使第一个三元组存放在下标 1 的位置上。

```
CONST maxnum=大于非零元素数的某个常量
TYPE tuple=RECORD
            i,j: integer;  v: elemtp;
           END;
 sparmattp=RECORD
            mu, nu, tu: integer;
            data: ARRAY[1..maxnum] OF tuple;
          END;
PROC AddMatrix (VAR A: sparmattp; B: sparmattp);
// 稀疏矩阵 A 和 B 各有 m 和 n 个非零元素，以三元组表存储。A 的空间足够大，本算法实现两个稀疏矩阵相加，结果放到 A 中。
  L:=m; p:=n; k:=m+n; // L,p 为 A，B 三元组表指针，k 为结果三元组表指针（下标）。
  A.tu:=m+n; // 暂存结果矩阵非零元素个数
 WHILE (L≥1) AND (p≥1) DO
  [CASE  // 行号不等时，行号大者的三元组为结果三元组表中一项。
    A.data[L].i>B.data[p].i: A.data[k]:=A.data[L]; L:=L-1; // A 中当前项为结果项
    A.data[L].i<B.data[p].i: A.data[k]:=B.data[p]; p:=p-1; //B 中当前项为结果当前项
    A.data[L].i=B.data[p].i:
                  CASE  //行号相等时，比较列号
                    A.data[L].j>B.data[p].j: A.data[k]:=A.data[L]; L:=L-1;
                    A.data[L].j<B.data[p].j: A.data[k]:=B.data[p]; p:=p-1;
                    A.data[L].j=B.data[p].j: IF A.data[L].v+B.data[p].v≠0
THEN
                                                [A.data[L].v=A.data[L].v+
B.data[p].v;
                                                 A.data[k]:= A.data[L]; ]
                                                 L:=L-1; p:=p-1;
                  ENDC;       //结束行号相等时的处理
    ENDC;     //结束行号比较处理。
    k:=k-1;  //结果三元组表的指针前移（减 1）
  ]//结束 WHILE 循环。
WHILE p>0 DO[A.data[k]:=B.data[p]; k:=k-1; p:=p-1; ]  //处理 B 的剩余部分。
WHILE L>1 DO[A.data[k]:=A.data[L]; k:=k-1; L:=L-1; ]  //处理 A 的剩余部分。
IF k>1 THEN    //稀疏矩阵相应元素相加时，有和为零的元素，因而元素总数<m+n。
 [FOR p:=k TO m+n DO A[p-k+1]:=A[p]; // 三元组前移，使第一个三元组的下标为 1。
  A.tu=m+n-k+1; ]                    // 修改结果三元组表中非零元素个数。
ENDP;   // 结束 addmatrix
```

[算法讨论]算法中三元组的赋值是“成组赋值”，可用行值、列值和元素值的三个赋值句代替。A 和 B 的三元组表的当前元素的指针 L 和 p，在每种情况处理后均有修改，而结果三元组表的指针 k 在 CASE 语句后统一处理（k:=k-1）。算法在 B 的第一个元素“大于”A 的最后一个元素时，时间复杂度最佳为 O（n），最差情况是每个元素都移动（赋值）了一次，且出现了和为零的元素，致使最后（m+n-k+1）个元素向前平移一次，时间复杂度最差为 O（m+n）。

3.[题目分析]从 n 个数中，取出所有 k 个数的所有组合。设数已存于数组 A[1..n]中。为使结果唯一，可以分别求出包括 A[n]和不包括 A[n]的所有组合。即包括 A[n]时，求出从 A[1..n-1]中取出 k-1 个元素的所有组合，不包括 A[n]时，求出从 A[1..n-1]中取出 k 个元素的所有组合。

```
CONST n=10; k=3;
TYPE ARR=ARRAY[1..n] OF integer;
VAR  A, B: ARR; // A 中存放 n 个自然数，B 中存放输出结果。
PROC outresult; //输出结果
  FOR j: =1 TO k DO write (B[j]); writeln;
ENDP;
PROC nkcombination (i,j,k: integer);
//从 i 个数中连续取出 k 个数的所有组合，i 个数已存入数组 A 中，j 为结果数组 B 中的下标。
 IF k=0 THEN outresult
   ELSE IF (i-k≥0) THEN [ B[j]:=A[i]; j:=j+1;
                          nkcombination (i-1, k-1, j);
                         nkcombination (i-1, k, j-1); ]
ENDP;
```

[算法讨论]本算法调用时，i 是数的个数（题目中的 n），k≤i，j 是结果数组的下标。按题中例子，用 nkcombination（5，1，3）调用。若想按正序输出，如 123，124，…，可将条件表达式 i-k≥0 改为 i+k-1≤n，其中 n 是数的个数，i 初始调用时为 1，两个调用语句中的 i-1 均改为 i+1。

4.[题目分析]题目中要求矩阵两行元素的平均值按递增顺序排序，由于每行元素个数相等，按平均值排列与按每行元素之和排列是一个意思。所以应先求出各行元素之和，放入一维数组中，然后选择一种排序方法，对该数组进行排序，注意在排序时若有元素移动，则与之相应的行中各元素也必须做相应变动。

```
void Translation (float *matrix, int n)
//本算法对 n×n 的矩阵 matrix，通过行变换，使其各行元素的平均值按递增排列。
{int i,j,k,l;
 float sum, min;       //sum 暂存各行元素之和
 float *p, *pi, *pk;
 for(i=0; i<n; i++)
  {sum=0.0; pk=matrix+i*n;  //pk 指向矩阵各行第 1 个元素.
   for (j=0; j<n; j++){sum+=*(pk); pk++;}  //求一行元素之和.
    *(p+i)=sum;             //将一行元素之和存入一维数组.
  }//for i
 for(i=0; i<n-1; i++)     //用选择法对数组 p 进行排序
   {min=*(p+i); k=i;      //初始设第 i 行元素之和最小.
```

```
    for(j=i+1;j<n;j++) if(p[j]<min) {k=j; min=p[j];}  //记新的最小值及行号.
    if(i!=k)              //若最小行不是当前行,要进行交换(行元素及行元素之和)
      {pk=matrix+n*k;   //pk 指向第 k 行第 1 个元素.
       pi=matrix+n*i;   //pi 指向第 i 行第 1 个元素.
       for(j=0;j<n;j++) //交换两行中对应元素.
         {sum=*(pk+j); *(pk+j)=*(pi+j); *(pi+j)=sum;}
       sum=p[i]; p[i]=p[k]; p[k]=sum;  //交换一维数组中元素之和.
      }//if
    }//for i
  free(p); //释放 p 数组.
}// Translation
```

[算法分析] 算法中使用选择法排序,比较次数较多,但数据交换(移动)较少.若用其它排序方法,虽可减少比较次数,但数据移动会增多.算法时间复杂度为 $O(n^2)$.

5. [题目分析] 因为数组中存放的是从 1 到 N 的自然数，原程序运行后,数组元素 A[i](1<=i<=N)中存放的是 A[1]到 A[i-1]中比原 A[i]小的数据元素的个数。易见 A[N]+1 就是原 A[N]的值(假定是 j,1<=j<=N)。设一元素值为 1 的辅助数组 flag，采用累加，确定一个值后，flag 中相应元素置零。下面程序段将 A 还原成原来的 A:

```
VAR flag:ARRAY[1..N] OF integer;
FOR i:=1 TO N DO flag[i]:=1;   //赋初值
FOR i:=N DOWNTO 1 DO
  BEGIN sum:=0; j:=1; found:=false;
        WHILE j<=N AND NOT found DO
          BEGIN sum:=sum+flag[j];
                IF sum=A[i]+1 THEN BEGIN flag[j]:=0; found:=true; END;
          END;
        A[i]:=j;
  END;
```

6.[题目分析] 寻找马鞍点最直接的方法,是在一行中找出一个最小值元素,然后检查该元素是否是元素所在列的最大元素,如是,则输出一个马鞍点,时间复杂度是 O(m*(m+n)).本算法使用两个辅助数组 max 和 min,存放每列中最大值元素的行号和每行中最小值元素的列号,时间复杂度为 O（m*n+m)，但比较次数比前种算法会增加，也多使用向量空间。

```
int m=10, n=10;
void Saddle(int A[m][n])
 //A 是 m*n 的矩阵,本算法求矩阵 A 中的马鞍点.
{int max[n]={0}, //max 数组存放各列最大值元素的行号,初始化为行号 0;
     min[m]={0}, //min 数组存放各行最小值元素的列号,初始化为列号 0;
    i, j;
 for(i=0;i<m;i++) //选各行最小值元素和各列最大值元素.
   for(j=0;j<n;j++)
     {if(A[max[j]][j]<A[i][j]) max[j]=i;  //修改第 j 列最大元素的行号
      if(A[i][min[i]]>A[i][j]) min[i]=j;  //修改第 i 行最小元素的列号.
     }
 for (i=0;i<m;i++)
```

```
    {j=min[i];                  //第 i 行最小元素的列号
     if(i==max[j])printf("A[%d][%d]是马鞍点，元素值是%d",i,j,A[i][j]);  //是马鞍点
     }
 }// Saddle
```

[算法讨论] 以上算法假定每行（列）最多只有一个可能的马鞍点，若有多个马鞍点，因为一行(或一列)中可能的马鞍点数值是相同的，则可用二维数组 min2，第一维是行向量，是各行行号，第二维是列向量，存放一行中最大值的列号。对最大值也同样处理，使用另一二维数组 max2，第一维是列向量，是各列列号，第二维存该列最大值元素的行号。最后用类似上面方法，找出每行(i)最小值元素的每个列号（j），再到 max2 数组中找该列是否有最大值元素的行号（i），若有，则是马鞍点。

7.[题目分析]我们用 l 代表最长平台的长度，用 k 指示最长平台在数组 b 中的起始位置(下标)。用 j 记住局部平台的起始位置，用 i 指示扫描 b 数组的下标，i 从 0 开始，依次和后续元素比较，若局部平台长度（i-j）大于 l 时，则修改最长平台的长度 k（l=i-j）和其在 b 中的起始位置（k=j），直到 b 数组结束，l 即为所求。

```
void Platform (int b[ ], int N)
 //求具有 N 个元素的整型数组 b 中最长平台的长度。
{l=1;k=0;j=0;i=0;
 while(i<n-1)
 {while(i<n-1 && b[i]==b[i+1]) i++;
  if(i-j+1>l) {l=i-j+1;k=j;}    //局部最长平台
  i++; j=i; }                   //新平台起点
 printf("最长平台长度%d，在 b 数组中起始下标为%d", l, k);
 }// Platform
```

8.[题目分析]矩阵中元素按行和按列都已排序，要求查找时间复杂度为 O（m+n），因此不能采用常规的二层循环的查找。可以先从右上角（i=a,j=d）元素与 x 比较，只有三种情况：一是 A[i,j]>x， 这情况下向 j 小的方向继续查找；二是 A[i,j]<x，下步应向 i 大的方向查找；三是 A[i,j]=x，查找成功。否则，若下标已超出范围，则查找失败。

```
 void search(datatype A[ ][ ], int a,b,c,d, datatype x)
  //n*m 矩阵 A,行下标从 a 到 b,列下标从 c 到 d,本算法查找 x 是否在矩阵 A 中.
  {i=a; j=d; flag=0; //flag 是成功查到 x 的标志
   while(i<=b && j>=c)
     if(A[i][j]==x) {flag=1;break;}
     else if (A[i][j]>x) j--; else i++;
    if(flag) printf("A[%d][%d]=%d",i,j,x);       //假定 x 为整型.
    else printf("矩阵 A 中无%d 元素", x);
  }算法 search 结束。
```

[算法讨论]算法中查找 x 的路线从右上角开始，向下（当 x>A[i,j]）或向左（当 x<A[i,j]）。向下最多是 m,向左最多是 n。最佳情况是在右上角比较一次成功,最差是在左下角(A[b,c])，比较 m+n 次，故算法最差时间复杂度是 O(m+n)。

9.[题目分析]本题的一种算法前面已讨论（请参见本章三、填空题 38）。这里给出另一中解法。分析数的填法，是按“从右上到左下”的”蛇形”，沿平行于副对角线的各条对角线上，将自然数从小到大填写。当从右上到左下时，坐标 i 增加，坐标 j 减小，当 j 减到小于 0 时结束，然后 j 从 0 开始增加，而 i 从当前值开始减少，到 i<0 时结束。然后继续如此循

环。当过副对角线后，在 i>n-1 时，j=j+2，开始从左下向右上填数；而当 j>n-1 时 i=i+2，开始从右上向左下的填数，直到 n*n 个数填完为止。

```
void Snake_Number(int A[n][n],int n)
  //将自然数 1..n*n,按”蛇形”填入 n 阶方阵 A 中.
{i=0; j=0; k=1;                  //i,j 是矩阵元素的下标,k 是要填入的自然数.
 while(i<n && j<n)
  {while(i<n && j>-1)            //从右上向左下填数,
     {A[i][j]=k++; i++ ;j--;}
   if((j<0)&&(i<n)) j=0;         //副对角线及以上部分的新 i,j 坐标.
      else {j=j+2; i=n-1;}       // 副对角线以下的新的 i,j 坐标.
   while(i>-1 && j<n)            //从左下向右上
      {A[i][j]=k++; i--; j++;}
   if(i<0 && j<n) i=0;
      else{i=i+2; j=n-1;}
  }//最外层 while
}//Snake_Number
```

10.[题目分析]判断二维数组中元素是否互不相同，只有逐个比较,找到一对相等的元素，就可结论为不是互不相同。如何达到每个元素同其它元素比较一次且只一次？在当前行，每个元素要同本行后面的元素比较一次(下面第一个循环控制变量 p 的 for 循环)，然后同第 i+1 行及以后各行元素比较一次，这就是循环控制变量 k 和 p 的二层 for 循环。

```
int JudgEqual(ing a[m][n],int m,n)
 //判断二维数组中所有元素是否互不相同，如是，返回 1；否则，返回 0。
{for(i=0;i<m;i++)
  for(j=0;j<n-1;j++)
   { for(p=j+1;p<n;p++) //和同行其它元素比较
      if(a[i][j]==a[i][p]) {printf(“no”); return(0); }
           //只要有一个相同的，就结论不是互不相同
     for(k=i+1;k<m;k++)  //和第 i+1 行及以后元素比较
      for(p=0;p<n;p++)
         if(a[i][j]==a[k][p]) {printf(“no”); return(0); }
   }// for(j=0;j<n-1;j++)
  printf(yes”); return(1);   //元素互不相同
}//算法 JudgEqual 结束
```

（2）二维数组中的每一个元素同其它元素都比较一次，数组中共 m*n 个元素，第 1 个元素同其它 m*n-1 个元素比较，第 2 个元素同其它 m*n-2 个元素比较，……，第 m*n-1 个元素同最后一个元素(m*n)比较一次，所以在元素互不相等时总的比较次数为(m*n-1)+(m*n-2)+…+2+1=（m*n）(m*n-1)/2。在有相同元素时,可能第一次比较就相同,也可能最后一次比较时相同,设在(m*n-1)个位置上均可能相同,这时的平均比较次数约为（m*n）(m*n-1)/4，总的时间复杂度是 O(n^4)。

11．二项式$(a+b)^n$展开式的系数的递归定义为：

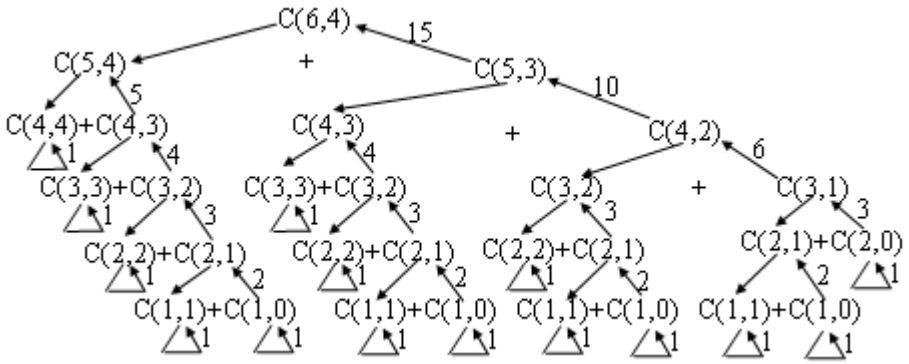
C(n,k)=

C(n,k)=C_n^k=

(1)int BiForm(int n,k)　　//二项式展开式的系数的递归算法

```
{if(n<0 || k<0 || k>=n) {printf(“参数错误\n ”);exit(0);}
 if(k==0 || k==n) return(1);
 else return(BiForm(n-1,k)+BiForm(n-1,k-1);
 }
```

(2)C(6,4)的递归树



(3)计算 C(n,k) (0<=k<=n)的非递归算法

```
int cnk(int n,int k)
 {int i; long x=1,y=1;
  for (i=1;i<=k;i++) x*=i;
  for (i=n-k+1;i<=n;i++) y*=i;
  return(y/x)
 }//cnk
```

12.[题目分析]本题属于排序问题，只是排出正负，不排出大小。可在数组首尾设两个指针 i 和 j，i 自小至大搜索到负数停止，j 自大至小搜索到正数停止。然后 i 和 j 所指数据交换，继续以上过程，直到 i=j 为止。

```
void Arrange(int A[],int n)
   //n 个整数存于数组 A 中，本算法将数组中所有正数排在所有负数的前面
 {int i=0,j=n-1,x;  //用类 C 编写，数组下标从 0 开始
  while(i<j)
   {while(i<j && A[i]>0)  i++;
    while(i<j && A[j]<0)  j--;
    if(i<j) {x=A[i]; A[i++]=A[j]; A[j--]=x; }//交换 A[i] 与 A[j]
   }
 }//算法 Arrange 结束.
```

[算法讨论]对数组中元素各比较一次，比较次数为 n。最佳情况(已排好,正数在前,负数在后)不发生交换，最差情况(负数均在正数前面)发生 n/2 次交换。用类 c 编写，数组界偶是 0..n-1。空间复杂度为 O(1).

类似本题的其它题的解答::

（1）与上面 12 题同，因要求空间复杂度也是 O(n)，可另设一数组 C，对 A 数组从左到右扫描，小于零的数在 C 中从左（低下标）到右（高下标）存，大于等于零的数在 C 中从右到

左存。

（2）将 12 题中判定正数(A[i]>0)改为判偶数(A[i]%2==0)，将判负数(A[j]<0)改为(A[j]%2!=0)。

（3）同（2），只是要求奇数排在偶数之前。

（4）利用快速排序思想，进行一趟划分。

```
int Partition(int A[],int n)
//将 n 个元素的数组 A 调整为左右两部分，且左边所有元素小于右边所有元素，返回分界位置。
{int i=0,j=n-1,rp=A[0];  //设数组元素为整型
 while(i<j)
  {while(i<j &&A[j]>=rp)  j--;
   while(i<j &&A[i]<=rp)  i++;
   if(i<j) { x=A[i];A[i]=A[j]; A[j]=x; }
  }
 A[i]=rp; return(i);   //分界元素
}// Partition
```

13. [题目分析] 设 n 个元素存放在数组 A[1..n]中。设 S 初始为空集，可依次将数组 A 的每一个元素并入 S，产生了含一个元素的若干集合，再以含一个元素的集合为初始集合，依次并入 A 的第二个（异于 S 的那个元素）元素并入 S，形成了含两个元素的若干集合，……，如此下去，直至 A[i]的全部元素并入。

```
CONST n=10;
TYPE datatype=char;
VAR A: array[1..n] OF datatype;
PROC  powerset(s:set OF datatype)
[outset(s);  //输出集合 S
 FOR i:=1 TO n DO  powerset(S+A[i]);
]
ENDP;
```

调用本过程时，参数 S 为空集[]。

14、[题目分析]设稀疏矩阵是 Amxn,Hm 是总表头指针。设 rch 是行列表头指针，则 rch->right=rch 时该行无非零元素，用 i 记行号，用一维数组元素 A[i]记第 i 行非零元个数。(为方便输出，设元素是整数。)

```
int MatrixNum(Olink Hm)
//输出由 Hm 指向的十字链表中每一行的非零元素个数
{Olink rch=Hm->uval.next, p;
 int A[]; i=1;//数组 A 记各行非零元个数,i 记行号
 while(rch!=Hm)//循环完各行列表头
  {p=rch->right; num=0; //p 是稀疏矩阵行内工作指针,num 记该行非零个数
   while(p!=rch)//完成行内非零元的查找
    {printf("M[%d][%d]=%d",p->row,p->col,p->uval.e);
     num++;p=p->right; printf("\n");//指针后移  }
    A[i++]=num;//存该行非零元个数
    rch=rch->uval.next;//移到下一行列表头
    }
```

```
    num=0
    for(j=1;j<i;j++)//输出各行非零元个数
      {num+=A[j]; printf(“第%d 行非零元个数为%d\n”,j,A[j]); }
     return(num);//稀疏矩阵非零元个数
  }算法结束
```

15、[题目分析] 广义表的元素有原子和表。在读入广义表“表达式”时，遇到左括号‘(’就递归的构造子表，否则若是原子，就建立原子结点；若读入逗号‘,’，就递归构造后续子表；若 n=0，则构造含空格字符的空表，直到碰到输入结束符号（‘#’）。设广义表的形式定义如下：

```
typedef  struct node
{int tag;                    //tag=0 为原子，tag=1 为子表
 struct node *link;          //指向后继结点的指针
 union {struct node *slink; //指向子表的指针
        char data;           //原子
        }element;
}Glist;
Glist *creat ()
//建立广义表的存储结构
{char ch; Glist *gh;
 scanf(“%c”,&ch);
 if(ch==’’ ) gh=null;
   else {gh=(Glist*)malloc(sizeof(Glist));
         if(ch== ‘(’ ){gh->tag=1; //子表
                       gh->element.slink=creat(); } //递归构造子表
         else {gh->tag=0;gh->element.data=ch;} //原子结点
        }
 scanf(“%c”,&ch);
 if(gh!=null) if(ch== ‘,’ ) gh->link=creat();  //递归构造后续广义表
              else gh->link=null;
 return(gh);
 }
}算法结束
```

16、(1)略

(2)求广义表原子个数的递归模型如下

$$f(p)=\begin{cases}0 & p=NIL\\ 1 & p\neq NIL \text{ 且 } p\uparrow.tag=0\\ f(p\uparrow.sublist)+f(p\uparrow.link) & p\neq NIL \text{ 且 } p\uparrow.tag=1\end{cases}$$

```
PROC Number(p:glist; VAR n: integer)
VAR m:integer;
n:=0;
IF p<>NIL THEN
  [IF p^.tag=0 THEN n:=1 ELSE Number(p^.sublist,m)
   n:=n+m; Number(p^.link,m);  n:=n+m; ]
ENDP;
```

17. int Count(glist *gl)
//求广义表原子结点数据域之和，原子结点数据域定义为整型
{if(gl==null) return(0);
 else if (gl->tag==0) return((p->data)+count(gl->link));
      else return(count(gl->sublist)+count(gl->link)); }
}// Count

18.（1）在 n 个正整数中，选出 k(k<<m）个最大的数，应使用堆排序方法。对深度为 h 的堆，筛选算法中关键字的比较次数至多为 2（h-1）次。建堆总共进行的关键字比较次数不超过 4n，堆排序在最坏情况下的时间复杂度是 O(nlogn)。

int r[1000]; // r[1000]是整型数组

（2）void sift(int r[],int k,m,tag)
//已知 r[k+1..m]是堆，本算法将 r[k..m]调整成堆，tag=1 建立大根堆,tag=2 建立小根堆
{i=k;j=2*i;x=r[k];
 while (j<=m)
  {if (tag==2) //建立小根堆
     {if (j<m && r[j]>r[j+1]) j++;//沿关键字小的方向筛选
      if(r[j]<x)) {r[i]=r[j];i=j;j=2*i;}
      else break;}
   else //建立大根堆
     {if (j<m && r[j]<r[j+1]) j++;//沿关键字小的方向筛选
      if(r[j]>x) {r[i]=r[j];i=j;j=2*i;}
      else break;}
   }
    r[i]=x;
  }//sift
main(int argc,char *argv[])
//根据命令行中的输入，从 1000 个数中选取 n 个最大数或 n 个最小数
{int m=1000,i,j;
 n=augv[2];   //从命令行输入的第二个参数是需要输出的数的个数
 if(n>m){printf(“参数错误\n”);exit(0);}
 for(i=0;i<m;i++) scanf(“%d”,&r[i]); //输入 1000 个大小不同的正整数
 if (augv[1]== ‘a’)   //输出 n 个最大数，要求建立大根堆
  {for(i=m/2;i>0;i--) sift(r,i,m,1)
   printf(“%d 个最大数依次为\n”,n);
   for(i=m;i>m-n+1;i--) //输出 n 个最大数
        {printf(“%5d”,r[i]); j++; if((j+1)%5==0) printf(“\n”);//一行打印 5 个数
   sift(r,1,i-1,1); } //调堆
  }
 else //(augv[1]== ‘i’)   //输出 n 个最小数，要求建立小根堆
  {for(i=m/2;i>0;i--) sift(r,i,m,2)
   printf(“%d 个最小数依次为\n”,n);
   for(i=m;i>m-n+1;i--) //输出 n 个最小数

```
        {printf("%5d",r[i]);  j++; if((j+1)%5==0) printf("\n");//一行打印5个数
         sift(r,1,i-1,2); } //调堆
        }
    }//main
```

[算法讨论]算法讨论了建堆，并输出 n（n 小于等于 m）个最大(小)数的情况，由于要求输出 n 个最大数或最小数，必须建立极大化堆和极小化堆。注意输出时的 for 循环控制到变量 i 从 m 变化到 m-n+1，这是堆的性质决定的，只有堆顶元素才是最大(小)的。要避免使 i 从 1 到 n 来输出 n 个最大(小)数的错误。

19、[题目分析] 题目要求调整后第一数组（A）中所有数均不大于第二个数组（B）中所有数。因两数组分别有序，这里实际是要求第一数组的最后一个数 A[m-1]不大于第二个数组的第一个数 B[0]。由于要求将第二个数组的数插入到第一个数组中。因此比较 A[m-1]和 B[0],如 A[m-1]>B[0],则交换。交换后仍保持 A 和 B 有序。重复以上步骤,直到 A[m-1]<=B[0]为止。

```
void ReArranger (int A[],B[],m,n)
//A 和 B 是各有 m 个和 n 个整数的非降序数组，本算法将 B 数组元素逐个插入到 A 中，使 A 中各元素均不大于 B 中各元素，且两数组仍保持非降序排列。
{ while (A[m-1]>B[0])
  {x=A[m-1];A[m-1]=B[0]; //交换 A[m-1]和 B[0]
   j=1;
   wkile(j<n && B[j]<x) B[j-1]=B[j++];    //寻找 A[m-1]的插入位置
   B[j-1]=x;
   x=A[m-1];i=m-2;
   wkile(i>=0 && A[i]>x) A[i+1]=A[i--];  //寻找 B[0]的插入位置
   A[i+1]=x;
  }
}算法结束
```

20、[题目分析]本题中数组 A 的相邻两段分别有序，要求将两段合并为一段有序。由于要求附加空间为 O(1),所以将前段最后一个元素与后段第一个元素比较，若正序，则算法结束；若逆序则交换，并将前段的最后一个元素插入到后段中，使后段有序。重复以上过程直到正序为止。

```
void adjust(int A[],int n)
//数组 A[n-2k+1..n-k]和[n-k+1..n]中元素分别升序，算法使 A[n-2k+1..n]升序
{i=n-k;j=n-k+1;
while(A[i]>A[j])
  {x=A[i]; A[i]=A[j]; //值小者左移，值大者暂存于 x
   k=j+1;
   while (k<n && x>A[k]) A[k-1]=A[k++];  //调整后段有序
   A[k-1]=x;
   i--; j--;      //修改前段最后元素和后段第一元素的指针
  }
}算法结束
```

[算法讨论]最佳情况出现在数组第二段[n-k+1..n]中值最小元素 A[n-k+1]大于等于第一段值最大元素 A[n-k]，只比较一次无须交换。最差情况出现在第一段的最小值大于第二

段的最大值，两段数据间发生了 k 次交换，而且每次段交换都在段内发生了平均（k-1）次交换，时间复杂度为 $O(n^2)$。

21、[题目分析]题目要求按 B 数组内容调整 A 数组中记录的次序，可以从 i=1 开始，检查是否 B[i]=i。如是，则 A[i]恰为正确位置，不需再调；否则，B[i]=k≠i，则将 A[i]和 A[k]对调，B[i]和 B[k]对调，直到 B[i]=i 为止。

```
void CountSort (rectype A[],int B[])
//A 是 100 个记录的数组，B 是整型数组，本算法利用数组 B 对 A 进行计数排序
{int i,j,n=100;
i=1;
while(i<n)
 {if(B[i]!=i)  //若 B[i]=i 则 A[i]正好在自己的位置上，则不需要调整
   { j=i;
     while (B[j]!=i)
       { k=B[j]; B[j]=B[k]; B[k]=k;    // B[j]和 B[k]交换
         r0=A[j];A[j]=A[k]; A[k]=r0; } //r0 是数组 A 的元素类型,A[j]和 A[k]
交换
   i++;} //完成了一个小循环，第 i 个已经安排好
 }//算法结束
```

22、[题目分析]数组 A 和 B 的元素分别有序，欲将两数组合并到 C 数组，使 C 仍有序，应将 A 和 B 拷贝到 C，只要注意 A 和 B 数组指针的使用，以及正确处理一数组读完数据后将另一数组余下元素复制到 C 中即可。

```
void union(int A[],B[],C[],m,n)
//整型数组 A 和 B 各有 m 和 n 个元素，前者递增有序，后者递减有序，本算法将 A 和 B
归并为递增有序的数组 C。
{i=0; j=n-1; k=0;// i，j，k 分别是数组 A,B 和 C 的下标，因用 C 描述，下标从 0 开
始
 while(i<m && j>=0)
  if(a[i]<b[j]) c[k++]=a[i++] else c[k++]=b[j--];
 while(i<m) c[k++]=a[i++];
 while(j>=0) c[k++]=b[j--];
}算法结束
```

[算法讨论]若不允许另辟空间，而是利用 A 数组（空间足够大），则初始 k=m+n-1，请参见第 2 章算法设计第 4 题。

23、[题目分析]本题要求建立有序的循环链表。从头到尾扫描数组 A，取出 A[i]（0<=i<n），然后到链表中去查找值为 A[i]的结点，若查找失败，则插入。

```
LinkedList creat(ElemType A[],int n)
//由含 n 个数据的数组 A 生成循环链表，要求链表有序并且无值重复结点
{LinkedList h;
h=(LinkedList)malloc(sizeof(LNode));//申请结点
h->next=h;  //形成空循环链表
for(i=0;i<n;i++)
  {pre=h;
   p=h->next;
   while(p!=h && p->data<A[i])
```

```
      {pre=p; p=p->next;} //查找 A[i]的插入位置
   if(p==h || p->data!=A[i])                    //重复数据不再输入
      {s=(LinkedList)malloc(sizeof(LNode));
       s->data=A[i]; pre->next=s; s->next=p;//将结点 s 链入链表中
      }
   }//for
  return(h);
}算法结束
```

第六章　　树和二叉树

一、选择题

1．已知一算术表达式的中缀形式为 A+B*C-D/E，后缀形式为 ABC*+DE/-，其前缀形式为（　　）

A．-A+B*C/DE　　B．-A+B*CD/E　　C．-+*ABC/DE　　D．-+A*BC/DE

【北京航空航天大学 1999 一、3 （2 分）】

2．算术表达式 a+b*（c+d/e）转为后缀表达式后为（　　）【中山大学 1999 一、5】

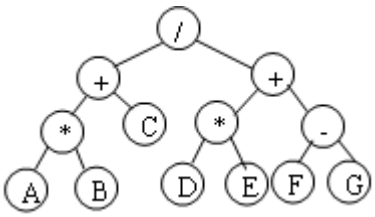
A．ab+cde/*　　B．abcde/+*+　　C．abcde/*++　　D．abcde*/++

3．设有一表示算术表达式的二叉树（见下图），它所表示的算术表达式是（　　）

【南京理工大学 1999 一、20（2 分）】

A．A*B+C/(D*E)+(F-G)　B．(A*B+C)/(D*E)+(F-G)

C．(A*B+C)/(D*E+（F-G）)　D．A*B+C/D*E+F-G



4．设树 T 的度为 4，其中度为 1，2，3 和 4 的结点个数分别为 4，2，1，1　则 T 中的叶子数为（　　）

A．5　　B．6　　C．7　　D．8

【南京理工大学 2000 一、8 （1.5 分）】

5．在下述结论中，正确的是（　　）【南京理工大学 1999 一、4 （1 分）】

①只有一个结点的二叉树的度为 0；　②二叉树的度为 2；　③二叉树的左右子树可任意交换；

④深度为 K 的完全二叉树的结点个数小于或等于深度相同的满二叉树。

A．①②③　　B．②③④　　C．②④　　D．①④

6．设森林 F 对应的二叉树为 B，它有 m 个结点，B 的根为 p,p 的右子树结点个数为 n,森林 F 中第一棵树的结点个数是（　　）

A．m-n　B．m-n-1　　C．n+1　D．条件不足，无法确定 【南京理工大学 2000 一、17（1.5 分）】

7．　树是结点的有限集合，它（（1））根结点，记为 T。其余结点分成为 m（m>0）个（（2））的集合 T1，T2，…，Tm，每个集合又都是树，此时结点 T 称为 Ti 的父结点，Ti 称为 T 的子结点（1≤i≤m）。一个结点的子结点个数称为该结点的（（3））。二叉树与树是两个不同的概念，二叉树也是结点的有限集合，它（（4））根结点。可以把树的根结点的层数定义为 1，其他结点的层数等于其父结点所在层数加上 1。令 T 是一棵二叉树，Ki 和 Kj 是 T 中子结点数小于 2 的结点中的任意两个，它们所在的层数分别为 λKi 和 λKj，当关系式 |λKi-λKj| ≤1 一定成立时，则称 T 为一棵（(5)）。供选择的答案：

（1）(4) A．有 0 个或 1 个　B．有 0 个或多个　C．有且只有一个　　D．有 1 个或 1 个以上

（2）　A．互不相交　　B.允许相交　　C.允许叶结点相交　D.允许树枝结点相交

（3）　A．权　　B.维数　　C.次数　　D.序

（5）A．丰满树　　B.查找树　　C.平衡树　　D.完全树【上海海运学院 1999 二、

2(5 分)】

8．若一棵二叉树具有 10 个度为 2 的结点，5 个度为 1 的结点，则度为 0 的结点个数是（　）

A．9　　B．11　　C．15　　D．不确定 【北京工商大学 2001 一.7(3 分)】

9．在一棵三元树中度为 3 的结点数为 2 个，度为 2 的结点数为 1 个，度为 1 的结点数为 2 个，则度为 0 的结点数为（　　）个

A．4　　B．5　　C．6　　D．7 【哈尔滨工业大学 2001 二、2 （2 分）】

10．设森林 F 中有三棵树，第一，第二，第三棵树的结点个数分别为 M1，M2 和 M3。与森林 F 对应的二叉树根结点的右子树上的结点个数是（　　）。【北方交通大学 2001 一、16 （2 分）】

A．M1　　B．M1+M2　　C．M3　　D．M2+M3

11．具有 10 个叶结点的二叉树中有（　）个度为 2 的结点，【北京航空航天大学 2000 一、5（2 分）】

A．8　　B．9　　C．10　　D．11

12．一棵完全二叉树上有 1001 个结点，其中叶子结点的个数是（　　）【西安交通大学 1996 三、2 (3 分)】

A．250　B．500　C．254　D．505　E．以上答案都不对

13．设给定权值总数有 n 个，其哈夫曼树的结点总数为(　　) 【福州大学 1998 一、5 (2 分)】

A．不确定　　B．2n　　C．2n+1　　D．2n-1

14．有 n 个叶子的哈夫曼树的结点总数为（　　）。【青岛大学 2002 二、1 （2 分）】

A．不确定　　B．2n　　C．2n+1　　D．2n-1

15．若度为 m 的哈夫曼树中，其叶结点个数为 n，则非叶结点的个数为（ ）。【中科院计算所 1999 一、2（2 分）】

A．n-1　　B．⌊n/m⌋-1　　C．⌈(n-1)/(m-1)⌉　　D．⌈n/(m-1)⌉-1　E.⌈(n+1)/(m+1)⌉-1

16．有关二叉树下列说法正确的是（　　）【南京理工大学 2000 一、11 （1.5 分）】

A．二叉树的度为 2　　B．一棵二叉树的度可以小于 2

C．二叉树中至少有一个结点的度为 2　D．二叉树中任何一个结点的度都为 2

17．二叉树的第 I 层上最多含有结点数为（　）

【中山大学 1998 二、7 （2 分）】【北京理工大学 2001 六、5（2 分）】

A．2^I　　B．$2^{I-1}-1$　　C．2^{I-1}　　D．2^I-1

18．一个具有 1025 个结点的二叉树的高 h 为（　　）【南京理工大学 1999 一、19 （2 分）】

A．11　　B．10　　C．11 至 1025 之间　　D．10 至 1024 之间

19．一棵二叉树高度为 h,所有结点的度或为 0，或为 2，则这棵二叉树最少有(　　)结点

A．2h　　B．2h-1　　C．2h+1　　D．h+1 【南京理工大学 2001 一、11(1.5 分)】

20．对于有 n 个结点的二叉树，其高度为（　　）【武汉交通科技大学 1996 一、5 (4 分)】

A．$n\log_2 n$　　B．$\log_2 n$　　C．$\lfloor\log_2 n\rfloor|+1$　　D．不确定

21．一棵具有 n 个结点的完全二叉树的树高度（深度）是（　　）【南京理工大学 1996 一、8 （2 分）】

A．⌊logn⌋+1　　B．logn+1　　C．⌊logn⌋　　D．logn-1

22．深度为 h 的满 m 叉树的第 k 层有（　）个结点。(1=<k=<h)【北京航空航天大学 2000

一、4（2 分）】

A．m^{k-1}　　B．$m^{k}-1$　　C．m^{h-1}　　D．$m^{h}-1$

23．在一棵高度为 k 的满二叉树中，结点总数为（　　）【北京工商大学 2001 一、3 (3 分)】

A．2^{k-1}　　B．2^{k}　　C．$2^{k}-1$　　D．$\lfloor \log 2^{k} \rfloor+1$

24．高度为 K 的二叉树最大的结点数为（　　）。【山东大学 2001 二、3 (1 分)】

A．2^{k}　　B．2^{k-1}　　C．$2^{k}-1$　　D．$2^{k-1}-1$

25．一棵树高为 K 的完全二叉树至少有（　　）个结点【南京理工大学 1998 一、3 （2 分)】

A．$2^{k}-1$　　B．$2^{k-1}-1$　　C．2^{k-1}　　D．2^{k}

26．将有关二叉树的概念推广到三叉树，则一棵有 244 个结点的完全三叉树的高度（）

A．4　　B．5　　C．6　　D．7 【南京理工大学 2000 一、5 1.5 分)】

27．利用二叉链表存储树，则根结点的右指针是（　　）。【青岛大学 2001 五、5 （2 分)】

A．指向最左孩子　　B．指向最右孩子　　C．空　　D．非空

28．对二叉树的结点从 1 开始进行连续编号，要求每个结点的编号大于其左、右孩子的编号，同一结点的左右孩子中，其左孩子的编号小于其右孩子的编号，可采用(　　)次序的遍历实现编号。【北京理工大学 2000 一、4 （2 分)】

A．先序　　B．中序　　C．后序　　D．从根开始按层次遍历

29．树的后根遍历序列等同于该树对应的二叉树的(　　)．【北京理工大学 2001 六、6 （2 分)】

A．先序序列　　B．中序序列　　C．后序序列

30．若二叉树采用二叉链表存储结构，要交换其所有分支结点左、右子树的位置，利用(　　)遍历方法最合适。

A．前序　　B．中序　　C．后序　　D．按层次【北京航空航天大学 1999 一、4 （2 分)】

31．在下列存储形式中，哪一个不是树的存储形式？（　　)【北方交通大学 2001 一、23 （2 分)】

A．双亲表示法　B．孩子链表表示法 C．孩子兄弟表示法 D．顺序存储表示法

32．一棵二叉树的前序遍历序列为 ABCDEFG，它的中序遍历序列可能是（　　)【北京工业大学 2001 一、2 (2 分)】

A．CABDEFG　　B．ABCDEFG　　C．DACEFBG　　D．ADCFEG

33．已知一棵二叉树的前序遍历结果为 ABCDEF,中序遍历结果为 CBAEDF,则后序遍历的结果为（　　)。

A．CBEFDA　　B．FEDCBA　　C．CBEDFA　　D．不定　　【浙江大学 1999 四、2 (4 分)】

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

A．acbed　　B．decab　　C．deabc　　D．cedba　　【山东大学 2001 二、7 (1 分)】

35．某二叉树中序序列为 A,B,C,D,E,F,G，后序序列为 B,D,C,A,F,G,E 则前序序列是：

A．E,G,F,A,C,D,B　　B．E,A,C,B,D,G,F　　C．E,A,G,C,F,B,D　　D．上面的都不对

【南京理工大学 2000 一、14 （1.5 分)】

36．上题的二叉树对应的森林包括多少棵树(　　)【南京理工大学 2000 一、15 （1.5 分)】

A．1　　B．2　　C．3　　D．概念上是错误的

37．二叉树的先序遍历和中序遍历如下： 先序遍历：EFHIGJK；中序遍历：HFIEJKG 。该二叉树根的右子树的根是：【北方交通大学 2001 一、21（2 分）】

A、 E　　B、 F　　C、 G　　D、 H

38．将一棵树 t 转换为孩子—兄弟链表表示的二叉树 h，则 t 的后根序遍历是 h 的

A．前序遍历　　B．中序遍历　　C．后序遍历（　　）【北京邮电大学 2001 一、2 （2 分）】

39．某二叉树 T 有 n 个结点，设按某种顺序对 T 中的每个结点进行编号，编号为 1，2，… ，n，且有如下性质：T 中任一结点 V，其编号等于左子树上的最小编号减 1，而 V 的右子树的结点中，其最小编号等于 V 左子树上结点的最大编号加 1。这时是按（　　）编号的。

A.中序遍历序列 B.前序遍历序列 C.后序遍历序列 D.层次顺序 【长沙铁道学院 1998 三、1(2 分)】

40．下面的说法中正确的是（　　）.

（1）任何一棵二叉树的叶子结点在三种遍历中的相对次序不变；

（2）按二叉树定义，具有三个结点的二叉树共有 6 种。

A.(1)(2)　B.(1)　C.(2)　D.(1)、(2)都错 【南京理工大学 2001 一、10 （1.5 分）】

41．对于前序遍历与中序遍历结果相同的二叉树为（1）；

对于前序遍历和后序遍历结果相同的二叉树为（2）。【中科院计算所 1999 一、4 （4 分）】

A．一般二叉树　　B．只有根结点的二叉树　　C．根结点无左孩子的二叉树

D．根结点无右孩子的二叉树 E．所有结点只有左子数的二叉树 F．所有结点只有右子树的二叉树

42．一棵非空的二叉树的先序遍历序列与后序遍历序列正好相反，则该二叉树一定满足（　　）

【南开大学 2000 一、2】

A．所有的结点均无左孩子 B．所有的结点均无右孩子 C．只有一个叶子结点 D．是任意一棵二叉树

43．在二叉树结点的先序序列，中序序列和后序序列中，所有叶子结点的先后顺序（　　）

A．都不相同　　B．完全相同　C．先序和中序相同，而与后序不同

D．中序和后序相同，而与先序不同 【北方交通大学 2001 一、25 （2 分）】

44．某二叉树的前序序列和后序序列正好相反，则该二叉树一定是（）的二叉树。【武汉大学 2000 二、4】

A．空或只有一个结点　　B．任一结点无左子树　　C．高度等于其结点数　　D．任一结点无右子树

45．在完全二叉树中，若一个结点是叶结点，则它没（　　）。【北方交通大学 2001 一、22（2 分）】

A．左子结点　　B．右子结点　　C．左子结点和右子结点　　D．左子结点，右子结点和兄弟结点

46．在下列情况中，可称为二叉树的是（　　）

A．每个结点至多有两棵子树的树　　B．哈夫曼树　　C．每个结点至多有两棵子树的有序树

D．每个结点只有一棵右子树　　E．以上答案都不对 【西安交通大学 1996 三、4 (3 分)】

47．一棵左子树为空的二叉树在先序线索化后，其中空的链域的个数是：（　　）

A．不确定　　B．0　　C．1　　D．2　【合肥工业大学 1999 一、5 （2 分）】

48．一棵左右子树均不空的二叉树在先序线索化后，其中空的链域的个数是：（　　）。

A．0　　B．1　　C．2　　D．不确定 【合肥工业大学 2000 一、5 （2 分）】

49．若 X 是二叉中序线索树中一个有左孩子的结点，且 X 不为根，则 x 的前驱为（　　）

【南京理工大学 1996 一、6 (2 分)】

A.X 的双亲　B.X 的右子树中最左的结点　C.X 的左子树中最右结点　D.X 的左子树中最右叶结点

50．引入二叉线索树的目的是（　　）

A．加快查找结点的前驱或后继的速度　B．为了能在二叉树中方便的进行插入与删除

C．为了能方便的找到双亲　　D．使二叉树的遍历结果唯一【南京理工大学 1998 一、5 （2 分）】

51．线索二叉树是一种（　　）结构。

A．逻辑　　B．逻辑和存储　C．物理　　D．线性【西安电子科技大学 1996 一、9 （2 分）】

52．n 个结点的线索二叉树上含有的线索数为（　　）

A．2n　　B．n－1　　C．n＋1　　D．n 【中山大学 1998 二、8 （2 分）】

53．（　　）的遍历仍需要栈的支持.

A．前序线索树　　B．中序线索树　　C．后序线索树　【中科院计算所 1999 一、1 （2 分）】

54．二叉树在线索后，仍不能有效求解的问题是（　　）。

A．前（先）序线索二叉树中求前（先）序后继　B．中序线索二叉树中求中序后继

C．中序线索二叉树中求中序前驱　D．后序线索二叉树中求后序后继 【武汉大学 2000 二、3 二、5】

55．设 F 是一个森林，B 是由 F 变换得的二叉树。若 F 中有 n 个非终端结点，则 B 中右指针域为空的结点有（　　）个。

A．n-1　　B．n　　C．n+1　　D．n+2　【西安电子科技大学 1998 一、10 （2 分）】

56．如果 T_2 是由有序树 T 转换而来的二叉树，那么 T 中结点的后序就是 T_2 中结点的（　　）。

A．先序　　B．中序　　C．后序　　D．层次序　【西安电子科技大学 1996 一、2 （2 分）】

57．由 3 个结点可以构造出多少种不同的有向树？（　　）

A．2　　B．3　　C．4　　D．5　【北方交通大学 2001 一、6 （2 分）】

58．由 3 个结点可以构造出多少种不同的二叉树？（　　）

A．2　　B．3　　C．4　　D．5　【北方交通大学 2001 一、7 （2 分）】

59.下述二叉树中,哪一种满足性质:从任一结点出发到根的路径上所经过的结点序列按其关键字有序（）。

A．二叉排序树　B．哈夫曼树 C．AVL 树　D．堆

【中国科技大学 1998 二、8（2 分）】【中科院计算所 1998 二、8(2 分)】

60.在叶子数目和权值相同的所有二叉树中,最优二叉树一定是完全二叉树,该说法（　　）。

A．正确　B．错误 【中国科技大学 1998 二、10（2 分）】【中科院计算所 1998 二、10（2 分）】

61．最优二叉树（哈夫曼树）、最优查找树均为平均查找路径长度 $\sum_{i=1}^{n} w_i h_i$ 最小的树，其中对最优二叉树，n 表示（1），对最优查找树，n 表示（2），构造这两种树均（3）。【中科院计算所 1999 一、3 (6 分)】

A．结点数　B．叶结点数　C．非叶结点数　D．度为 2 的结点数　E．需要一张 n 个关键字的有序表

F．需要对 n 个关键字进行动态插入　G．需要 n 个关键字的查找概率表　H．不需要任何前提

62．下述编码中哪一个不是前缀码（　　）。【中科院计算所 2000 一、2 （2 分)】

A.（00，01，10，11）　B.（0，1，00，11）　C.（0，10，110，111）　D.（1，01，000，001）

63．下面几个符号串编码集合中，不是前缀编码的是（　　）。

A ．　{0,10,110,1111}　　B ．　{11,10,001,101,0001}

C．{00,010,0110,1000}

D．{b,c,aa,ac,aba,abb,abc}　【西安电子科技大学 2001 应用 一、6（2 分)】

64．当一棵有 n 个结点的二叉树按层次从上到下，同层次从左到右将数据存放在一维数组 A[1..n]中时，数组中第 i 个结点的左孩子为（　　）【南京理工大学 1999 一、18(2 分)】

A．A[2i](2i=<n)　　B．A[2i+1](2i+1=< n)　　C．A[i/2]　　D．无法确定

65．一棵有 n 个结点的二叉树，按层次从上到下，同一层从左到右顺序存储在一维数组 A[1..n]中，则二叉树中第 i 个结点（i 从 1 开始用上述方法编号）的右孩子在数组 A 中的位置是（　　）

A．A[2i](2i<=n)　B．A[2i+1](2i+1<=n)　C．A[i-2]　D．条件不充分，无法确定

【南京理工大学 2000 一、4(1.5 分)】

66．从下列有关树的叙述中,选出 5 条正确的叙述(共 5 分)（　　）

A．二叉树中每个结点有两个子结点,而树无此限制,因此二叉树是树的特殊情况。

B．当 K≥1 时高度为 K 的二叉树至多有 2^{k-1} 个结点。

C．用树的前序周游和中序周游可以导出树的后序周游。

D．线索二叉树的优点是便于在中序下查找前驱结点和后继结点。

E．将一棵树转换成二叉树后,根结点没有左子树。

F．一棵含有 N 个结点的完全二叉树,它的高度是 $\lfloor LOG_2N \rfloor+1$。

G．在二叉树中插入结点,该二叉树便不再是二叉树。

H．采用二叉树链表作树的存储结构,树的前序周游和其相应的二叉树的前序周游的结果是一样的。

I．哈夫曼树是带权路径最短的树,路径上权值较大的结点离根较近。

J．用一维数组存储二叉树时,总是以前序周游存储结点。【山东工业大学 1995 三、 (5 分)】

二、判断题

1．二叉树是度为 2 的有序树。【长沙铁道学院 1997 一、3(1 分)】【中科院软件所 1997 一、9(1 分)】

2．完全二叉树一定存在度为 1 的结点。【青岛大学 2002 一、4 （1 分)】

3．对于有 N 个结点的二叉树，其高度为 $\log_2 n$。【上海海运学院 1998 一、6 （1 分)】

4．深度为K的二叉树中结点总数≤2^k-1。【南京航空航天大学 1995 五、1 （1分）】

5. 二叉树以后序遍历序列与前序遍历序列反映的同样的信息(他们反映的信息不独立)。
【华南理工大学2002 一、7 (1分)】

6. 二叉树的遍历结果不是唯一的.【南京理工大学 1997 二、8 （2分）】

7. 二叉树的遍历只是为了在应用中找到一种线性次序。【青岛大学 2001 四、4 （1分）】

8. 树可用投影法进行中序遍历。 【青岛大学 2002 一、6 （1分）】

9. 一个树的叶结点，在前序遍历和后序遍历下，皆以相同的相对位置出现。
【上海海运学院 1995 一、4 （1分）】

10. 二叉树的前序遍历并不能唯一确定这棵树，但是，如果我们还知道该树的根结点是那一个，则可以确定这棵二叉树。【上海海运学院 1995 一、6 （1分）】

11. 一棵一般树的结点的前序遍历和后序遍历分别与它相应二叉树的结点前序遍历和后序遍历是一致的。【上海海运学院 1996 一、6 （1分）】

12．对一棵二叉树进行层次遍历时，应借助于一个栈。【南京航空航天大学 1995 五、3 （1分）】

13．用树的前序遍历和中序遍历可以导出树的后序遍历。【北京邮电大学 1999 二、3 （2分）】

14.采用二叉链表作存储结构,树的前序遍历和其相应的二叉树的前序遍历的结果是一样的。
【北京邮电大学2000 一、2(1分)】

15. 用一维数组存储二叉树时，总是以前序遍历顺序存储结点。【上海海运学院 1995 一、8（1分）】

16. 中序遍历二叉链存储的二叉树时，一般要用堆栈；中序遍历检索二叉树时，也必须使用堆栈。
【上海海运学院1998 一、7(1分)】

17．中序遍历一棵二叉排序树的结点就可得到排好序的结点序列【中科院软件所 1999 六、1-1 （2分）】

18. 后序线索二叉树是不完善的，要对它进行遍历，还需要使用栈。【 长沙铁道学院 1998 一、2 (1分)】

19．任何二叉树的后序线索树进行后序遍历时都必须用栈。【西安交通大学 1996 二、2（3分）】

20．任何一棵二叉树都可以不用栈实现前序线索树的前序遍历。【西安交通大学 1996 二、1 (3分)】

21．由一棵二叉树的前序序列和后序序列可以唯一确定它。【中科院软件所 1997 一、3 （1分）】

22．完全二叉树中，若一个结点没有左孩子，则它必是树叶。
【东南大学 2001 一、1-8（1分）】【中科院软件所1997 一、2（1分）】【山东大学2001 一、4 (1分)】

23. 二叉树只能用二叉链表表示。【南京理工大学 1997 二、6 （2分）】

24. 一棵有 n 个结点的二叉树，从上到下，从左到右用自然数依次给予编号，则编号为 i 的结点的左儿子的编号为2i(2i< n)，右儿子是2i+1（2i+1<n)。【南京理工大学 1997 二、11 （2分）】

25. 给定一棵树，可以找到唯一的一棵二叉树与之对应。【青岛大学 2001 一、5 （1分）】

26. 一棵树中的叶子数一定等于与其对应的二叉树的叶子数。【青岛大学 2002 一、5 （1分）】

27. 用链表(llink-rlink)存储包含n个结点的二叉树,结点的2n个指针区域中有n-1个空

指针。

【上海海运学院 1996 一.5(1 分)】

28. 二叉树中每个结点至多有两个子结点,而对一般树则无此限制.因此,二叉树是树的特殊情形.

【上海海运学院 1997 一.5(1 分)】

29. 树形结构中元素之间存在一个对多个的关系。【燕山大学 1998 二、1 (2 分)】

30. 在二叉树的第 i 层上至少有 2^{i-1} 个结点(i>=1)。【燕山大学 1998 二、3 (2 分)】

31. 必须把一般树转换成二叉树后才能进行存储。【南京航空航天大学 1997 一、4 (1 分)】

32. 完全二叉树的存储结构通常采用顺序存储结构。【南京航空航天大学 1996 六、3 (1 分)】

33. 将一棵树转成二叉树，根结点没有左子树;【北京邮电大学 1999 二、2 (2 分)】

34. 在二叉树中插入结点，则此二叉树便不再是二叉树了。【北京邮电大学 2000 一、5 (1 分)】

35. 二叉树是一般树的特殊情形。【北京邮电大学 2000 一、9 (1 分) 2002 一、6 (1 分)】

36. 树与二叉树是两种不同的树型结构。【东南大学 2001 一、1-7 (1 分)】

37. 非空的二叉树一定满足：某结点若有左孩子，则其中序前驱一定没有右孩子

【合肥工业大学 2001 二、5 (1 分)】

38. 在任意一棵非空二叉排序树，删除某结点后又将其插入，则所得二叉排序树与删除前原二叉排序树相同。【中科院软件所 1997 一、7 (1 分)】

39. 度为二的树就是二叉树。【大连海事大学 2001 一、7 (1 分)】

40. 深度为 k 具有 n 个结点的完全二叉树，其编号最小的结点序号为 $\lfloor 2^{k-2} \rfloor+1$。

【东北大学 1997 二、3 (2 分)】

41.下面二叉树的定义只有一个是正确的，请在正确的地方画“√”。

（1）它是由一个根和两株互不相交的、称为左子树和右子树的二叉树组成。

（2）（a）在一株二叉树的级 i 上，最大结点数是 2^{i-1}（$i \geqslant 1$）

（b）在一棵深度为 k 的二叉树中，最大结点数是 $2^{k-1}+1$（$k \geqslant 1$）。

（3）二叉树是结点的集合，满足如下条件：

（a）它或者是空集;

（b）或者是由一个根和两个互不相交的、称为左子树和右子树的二叉树组成。

【中科院自动化所 1995 一、2（6 分)】

42. 在中序线索二叉树中，每一非空的线索均指向其祖先结点。【合肥工业大学 2000 二、5（1 分)】

43. 线索二叉树的优点是便于是在中序下查找前驱结点和后继结点。

【上海海运学院 1995 ,96,97 一、7（1 分)】

44. 二叉树中序线索化后，不存在空指针域。【青岛大学 2000 四、3 (1 分)】

45. 霍夫曼树的结点个数不能是偶数。【北京邮电大学 2000 一、6 (1 分)】

46. 一棵哈夫曼树的带权路径长度等于其中所有分支结点的权值之和。【合肥工业大学 2000 二、4 (1 分)】

47. 哈夫曼树无左右子树之分。【青岛大学 2000 四、8 (1 分)】

48. 当一棵具有 n 个叶子结点的二叉树的 WPL 值为最小时，称其树为 Huffman 树，且其二叉树的形状必是唯一的。【南京航空航天大学 1995 五、6 (1 分)】

49. 哈夫曼树是带权路径长度最短的树，路径上权值较大的结点离根较近。

【北京邮电大学 1999 二、5 (2 分)】

50. 用链表（llink-rlink）存储包含 n 个结点的二叉树时，结点的 2n 个指针区域中有 n+1 个空指针。（ ）

【上海海运学院 1999 一、6（1 分）】

三、填空题

1．二叉树由__(1)__，__(2)__，__(3)__三个基本单元组成。【燕山大学 1998 一、5 （3 分）】

2．树在计算机内的表示方式有__(1)__，__(2)__，__(3)__。【哈尔滨工业大学 2000 二、4 (3 分)】

3．在二叉树中，指针 p 所指结点为叶子结点的条件是_____。【合肥工业大学 1999 三、7（2 分）】

4．中缀式 a+b*3+4*（c-d）对应的前缀式为__(1)__，若 a=1，b=2，c=3，d=4，则后缀式 db/cc*a-b*+的运算结果为__(2)__。【西南交通大学 2000 一、6】

5．二叉树中某一结点左子树的深度减去右子树的深度称为该结点的____。【燕山大学 1998 一、9(1 分)】

6．具有 256 个结点的完全二叉树的深度为_____。【燕山大学 1998 一、4 （1 分）】

7．已知一棵度为 3 的树有 2 个度为 1 的结点，3 个度为 2 的结点，4 个度为 3 的结点，则该树有_____个叶子结点。【厦门大学 2000 六、2 （16%/3 分）】

8．深度为 k 的完全二叉树至少有___(1)___个结点，至多有___(2)___个结点。

【厦门大学 2001 一、4 （14%/5 分）】【南京理工大学 1999 二、5 （4 分）】

9．深度为 H 的完全二叉树至少有__(1)__个结点；至多有__(2)__个结点；H 和结点总数 N 之间的关系是 __(3)__。

【中科院计算所 1998 一、3（3 分）1999 二、4（3 分）】【中国科技大学 1998 一、3(4 分)】

10．在顺序存储的二叉树中，编号为 i 和 j 的两个结点处在同一层的条件是_____。

【厦门大学 2002 六、3 （4 分）】

11．在完全二叉树中，编号为 i 和 j 的两个结点处于同一层的条件是_____。

【合肥工业大学 2000 三、6 （2 分）】

12．一棵有 n 个结点的满二叉树有__(1)__个度为 1 的结点、有__(2)__个分支 （非 终端）结点和__(3)__个叶子，该满二叉树的深度为__(4)__。【华中理工大学 2000 一、6 （3 分））

13．假设根结点的层数为 1，具有 n 个结点的二叉树的最大高度是_____。

【北方交通大学 2001 二、1】

14.在一棵二叉树中,度为零的结点的个数为 N0,度为 2 的结点的个数为 N2,则有 N0 =_____

【北方交通大学 2001 二、6】【南京理工大学 1999 二、4 （2 分）】

15．设只含根结点的二叉树的高度为 0，则高度为 k 的二叉树的最大结点数为_____,最小结点数为_____。【北京大学 1997 一、1 （4 分）】

16．设有 N 个结点的完全二叉树顺序存放在向量 A[1:N]中，其下标值最大的分支结点为_____。

【 长沙铁道学院 1997 二、3 (2 分)】

17．高度为 K 的完全二叉树至少有_____个叶子结点。【合肥工业大学 1999 二、6（2 分）】

18．高度为 8 的完全二叉树至少有_____个叶子结点。【合肥工业大学 2001 三、6（2 分）】

19．已知二叉树有 50 个叶子结点，则该二叉树的总结点数至少是_____。

【厦门大学 2002 六、4（4 分）】

20．一个有 2001 个结点的完全二叉树的高度为_____。【南京理工大学 1997 三、2（1 分）】

21．设 F 是由 T1,T2,T3 三棵树组成的森林,与 F 对应的二叉树为 B,已知 T1,T2,T3 的结点数

分别为 n1,n2 和 n3 则二叉树 B 的左子树中有__(1)_个结点，右子树中有_(2)__个结点。

【南京理工大学 2000 二、9（3 分）】

22．一个深度为 k 的，具有最少结点数的完全二叉树按层次，（同层次从左到右）用自然数依此对结点编号，则编号最小的叶子的序号是__(1)_；编号是 i 的结点所在的层次号是_(2)__（根所在的层次号规定为 1 层）。【南京理工大学 2001 二、2（2 分）】

23．如某二叉树有 20 个叶子结点，有 30 个结点仅有一个孩子，则该二叉树的总结点数为_____。

【南京理工大学 2001 二、3（2 分）】

24．如果结点 A 有 3 个兄弟，而且 B 是 A 的双亲，则 B 的度是_____。

【西安电子科技大学 1999 软件 一、4（2 分）】

25．高度为 h 的 2-3 树中叶子结点的数目至多为_____。【西安电子科技大学 1999 软件 一、6（2 分）】

26．完全二叉树中，结点个数为 n，则编号最大的分支结点的编号为_____。

【北京轻工业学院 2000 一、3 (2 分)】

27．设一棵完全二叉树叶子结点数为 k,最后一层结点数>2，则该二叉树的高度为_____。

【北京科技大学 1998 一、3】

28．对于一个具有 n 个结点的二元树，当它为一棵_(1)_二元树时具有最小高度，当它为一棵_(2)_时，具有最大高度。【哈尔滨工业大学 2001 一、3 (2 分)】

29．具有 N 个结点的二叉树，采用二叉链表存储，共有_____个空链域。【重庆大学 2000 一、8】

30．8 层完全二叉树至少有_____个结点，拥有 100 个结点的完全二叉树的最大层数为_____。

【西南交通大学 2000 一、1】

31．含 4 个度为 2 的结点和 5 个叶子结点的二叉树，可有_____个度为 1 的结点。

【北京工业大学 2001 一、6 (2 分)】

32．一棵树 T 中，包括一个度为 1 的结点，两个度为 2 的结点，三个度为 3 的结点，四个度为 4 的结点和若干叶子结点，则 T 的叶结点数为_____。【山东大学 2001 三、2 (2 分)】

33． n（n 大于 1）个结点的各棵树中，其深度最小的那棵树的深度是_(1)__。它共有_(2)__个叶子结点和_(3)__个非叶子结点，其中深度最大的那棵树的深度是_(4)__，它共有_(5)__个叶子结点和_(6)__个非叶子结点。【山东大学 2001 三、7 (2 分)】

34． 每一棵树都能唯一的转换为它所对应的二叉树。若已知一棵二叉树的前序序列是 BEFCGDH，对称序列是 FEBGCHD，则它的后序序列是_(1)__。设上述二叉树是由某棵树转换而成，则该树的先根次序序列是_(2)__。【山东工业大学 1997 二、 (6 分)】

35．先根次序周游树林正好等同于按_(1)__周游对应的二叉树，后根次序周游树林正好等同于按__(2)_周游对应的二叉树。【山东工业大学 1999 二、1 （4 分)】

36．二叉树结点的对称序序列为 A,B,C,D,E,F,G,后序序列为 B,D,C,A,F,G,E,则该二叉树结点的前序序列为_(1)__,则该二叉树对应的树林包括_(2)__棵树。【北京大学 1997 一、2（4 分)】

37．二叉树的先序序列和中序序列相同的条件是_____。【合肥工业大学 2000 三、7（2 分)】

38．已知一棵二叉树的前序序列为 abdecfhg,中序序列为 dbeahfcg，则该二叉树的根为_(1)__，左子树中有_(2)__， 右子树中有_(3)__。【南京理工大学 1996 二、1（6 分)】

39．设二叉树中每个结点均用一个字母表示，若一个结点的左子树或右子树为空，用 . 表示，现前序遍历二叉树，访问的结点的序列为 ABD.G...CE.H..F..，则中序遍历二叉树时，访问的结点序列为_(1)__;后序遍历二叉树时，访问的结点序列为_(2)__。【南京理工大学

1999 二、3（4 分)】

40．已知二叉树前序为 ABDEGCF，中序为 DBGEACF，则后序一定是____。【青岛大学 2000 六、3(2 分)】

41．现有按中序遍历二叉树的结果为 abc，问有__(1)__种不同的二叉树可以得到这一遍历结果，这些二叉树分别是__(2)__。【中国矿业大学 2000 一、5（3 分)】

42．一个无序序列可以通过构造一棵_____树而变成一个有序序列，构造树的过程即为对无序序列进行排序的过程。【西安电子科技大学 1999 软件 一、4（2 分)】

43．利用树的孩子兄弟表示法存储，可以将一棵树转换为_____。【重庆大学 2000 一、9】

44．若一个二叉树的叶子结点是某子树的中序遍历序列中的最后一个结点，则它必是该子树的_____序列中的最后一个结点。【武汉大学 2000 一、2】

45．先根次序周游树林正好等同于按_____周游对应的二叉树；后根次序周游树林正好等同于_____周游对应的二叉树。【山东大学 1999 二、1 （4 分)】

46．在一棵存储结构为三叉链表的二叉树中，若有一个结点是它的双亲的左子女，且它的双亲有右子女，则这个结点在后序遍历中的后继结点是_____。【中国人民大学 2001 一、4（2 分）】

47．一棵左子树为空的二叉树在先序线索化后，其中的空链域的个数为：_____。

【厦门大学 2002 六、1 （4 分)】

48．具有 n 个结点的满二叉树，其叶结点的个数是_____。【北京大学 1994】

49．设一棵后序线索树的高是 50，结点 x 是树中的一个结点，其双亲是结点 y,y 的右子树高度是 31，x 是 y 的左孩子。则确定 x 的后继最多需经过_____中间结点（不含后继及 x 本身）

【南京理工大学 2000 二、8（1.5 分)】

50．线索二元树的左线索指向其_____，右线索指向其_____。

【哈尔滨工业大学 2000 二、3 (2 分)】

51．设 y 指向二叉线索树的一叶子，x 指向一待插入结点，现 x 作为 y 的左孩子插入，树中标志域为 ltag 和 rtag，并规定标志为 1 是线索，则下面的一段算法将 x 插入并修改相应的线索，试补充完整：(lchild,rchild 分别代表左，右孩子)

```
x^.ltag:= (1)  ; x^.lchild:= (2)  ; y^.ltag:= (3)  ;
y^.lchild:= (4)  ; x^.rtag:= (5)  ; x^.rchild:= (6)  ;
IF (x^.lchild<>NIL) AND (x^lchild^.rtag=1)  THEN  x^.lchild^.rchild:= (7)  ;
```

【南京理工大学 1997 三、7 （9 分)】

52．哈夫曼树是_____。【北京理工大学 2001 七、4 （2)】【 长沙铁道学院 1998 二、3 (2 分)】

53．若以{4，5，6，7，8}作为叶子结点的权值构造哈夫曼树，则其带权路径长度是_____。

【西安电子科技大学 2001 软件 一、3 （2 分)】【厦门大学 2002 六、2（4 分)】

54．有数据 WG={7，19，2，6，32，3，21，10}，则所建 Huffman 树的树高是__(1)__，带权路径长度 WPL 为__(2)__。【南京理工大学 1999 三、6（4 分)】

55．有一份电文中共使用 6 个字符:a,b,c,d,e,f,它们的出现频率依次为 2,3,4,7,8,9，试构造一棵哈夫曼树，则其加权路径长度 WPL 为__(1)__，字符 c 的编码是__(2)__。【中国矿业大学 2000 一、7（3 分)】

56．设 n_0 为哈夫曼树的叶子结点数目，则该哈夫曼树共有_____个结点。

【西安电子科技大学 1999 软件 一、2（2 分)】

57．①二叉树用来表示表达式，因为需要保存各子树的值，修改二叉树的结点结构为(Lchild,Data,Val,Rchild)。其中 Lchild,Rchild 的意义同前，Val 用来存放以该结点为根

的子树的值，值的类型依具体情况而定。为了简便起见，算法假定所考虑的表达式只有+，-，*，/ 四种二目运算，且已表示成相应的二叉树。算法所计算的表达式值放在根结点的 Val 域中。

```
PROC  Postorder-eval(t:ptrType)
BEGIN IF (t!=NULL)
   BEGIN  (1)_____;  (2)_____;
        CASE  t^.data:
         '+' :   t^.Val:=t^. Lchild^. Val + t^. Rchild ^. Val;   BREAK;
         '-' :   t^.Val:=t^. Lchild^. Val - t^. Rchild ^. Val;   BREAK;
         '*' :   t^.Val:=t^. Lchild^. Val * t^. Rchild ^. Val;   BREAK;
         '/' :   t^.Val:=t^. Lchild^. Val / t^. Rchild ^. Val;   BREAK;
        otherwise: (3)___; BREAK;
       ENDCASE  END
END;
```

②

```
PROC  Delete(x:datatype,A:tree)
 BEGIN tempA:= (4)   ;
       WHILE (tempA^.Item!=x) AND (tempA!=NULL) DO
         IF (x<tempA^.item) BEGIN r:=tempA; tempA:= (5)   ; END
         ELSE BEGIN r:=tempA;tempA:=tempA^.Rchild;END;//tempA 为要删结点，r 为
tempA 的父亲
      IF (6)   return(x);
      IF (tempA^.Lchild!=NULL) AND (tempA^.rchild!=NULL)
        BEGIN t:=tempA; q:=tempA^.Rchild;
              WHILE (q^.Lchild!=NULL) DO BEGIN t:=q; q:=q^.Lchild; END;
              t^.Lchild:= (7)___; //删去 q
              q^.Lchild :=tempA^.Lchild; q^.Rchild:=tempA^.Rchild;
              IF (tempA^.item< r^.item) r^.Lchild := (8)_ ELSE r^.Rchild:=q //
用 q 代替 tempA
        END;
      ELSE            IF(tempA^.Lchild!=NULL)            IF(tempA^.item<r^.item)
r^.Lchild:=tempA^.Lchild
                                    ELSE r^.Rchild:=tempA^.Lchild
           ELSE   IF(tempA^.Rchild!=NULL)   IF(tempA^.item<r^.item)   r^.Rchild:=
(9)
                                              ELSE r^.Lchild:=tempA^.Rchild
      ELSE  //tempA 为树叶
        IF(10)   r^.Lchild:=NULL ELSE r^.Rchild:=NULL
END;
```

【中山大学 1999 四、 (20 分)】

58. 下面的类 PASCAL 语言递归算法的功能是判断一棵二叉树（采用二叉链表存贮结构）是否为完全二叉树。请把空缺的两部分补写完整。（提示：利用完全二叉树结点序号性质）

```
    TYPE link=^node;
         node=RECORD key:keytype;  l,r:link; END;
    VAR  all:boolean; n:integer; root:link;
    FUNC num(t:link):integer;
```

```
    BEGIN (1)_____END;
  PROC chk(t:link;m{t 所指结点应有序号}:integer)
    BEGIN (2)_____END;
  BEGIN          {建二叉树，其根由 root 指出 }
   n:=num(root);{求结点数}  all:=true;  chk(root,1);
   IF all THEN writeln（'该树为完全二叉树!'）ELSE  writeln （' 该树非完全二叉树!'）
  END;   【北京工业大学 1997 二、2 (10 分)】
```

59．将二叉树 bt 中每一个结点的左右子树互换的 C 语言算法如下，其中 ADDQ(Q,bt),DELQ(Q),EMPTY(Q)分别为进队，出队和判别队列是否为空的函数，请填写算法中得空白处，完成其功能。

```
  typedef struct node
     {int data ; struct node *lchild, *rchild; }btnode;
  void  EXCHANGE(btnode *bt)
  {btnode *p, *q;
   if (bt){ADDQ(Q,bt);
           while(!EMPTY(Q))
            {p=DELQ(Q);  q=(1)___; p->rchild=(2)___; (3)___=q;
             if(p->lchild) (4)___; if(p->rchild) (5)___;
             }
  }   }//【北京科技大学 2000 二、(10 分)】
```

60．设 t 是给定的一棵二叉树，下面的递归程序 count(t)用于求得:二叉树 t 中具有非空的左,右两个儿子的结点个数 N2;只有非空左儿子的个数 NL;只有非空右儿子的结点个数 NR 和叶子结点个数 N0。N2、NL、NR、N0 都是全局量，且在调用 count(t)之前都置为 0.

```
typedef  struct node
  {int  data; struct node *lchild,*rchild;}node;
int N2,NL,NR,N0;
void count(node *t)
{if (t->lchild!=NULL) if (1)___ N2++; else NL++;
 else  if (2)___ NR++; else  (3)___ ;
 if(t->lchild!=NULL)(4)____; if (t->rchild!=NULL) (5)____;
} /*call form :if(t!=NULL) count(t);*/
```

【上海大学 2000 一、3 (10 分)】

61．下面是求二叉树高度的类 PASCAL（注：编者略）及类 C 写的递归算法试补充完整

［说明］（1）考生可根据自己的情况任选一个做（都选不给分）

（2）二叉树的两指针域为 lchild 与 rchild, 算法中 p 为二叉树的根，lh 和 rh 分别为以 p 为根的二叉树的左子树和右子树的高，hi 为以 p 为根的二叉树的高，hi 最后返回。

```
  height(p)
   {if ((1)___)
     {if(p->lchild==null) lh=(2)_____; else  lh=(3)_____;
      if(p->rchild==null) rh=(4)_____; else  rh=(5)_____;
      if (lh>rh) hi=(6)___; else  hi=(7)_____;
     }
    else   hi=(8)_____;
```

```
    return    hi;
  }//
```
【南京理工大学 1997 三、8 （15 分）】

62. 二叉树以链方式存储，有三个域，数据域 data，左右孩子域 lchild,rchild。树根由 tree 指向 。现要求按层次从上到下，同层次从左到右遍历树。下面算法中用到 addx(p)，将指针 p 进队，delx( )将队头元素返回并退队，notempty 在队不空时返回 true，否则为 false，将算法补充完整:

```
PROC  processnode(p);
  IF(1)_____THEN [write(p ^ .data); (2)_____ ]
ENDP;
PROC   trave(tree);
  write(tree ^ .data); (3)_____;
  WHILE  notempty()  DO
  [ r:=delx( ); processnode(r ^ .lchild); processnode((4)_____)]
ENDP;
```
【南京理工大学 1999 三、5 （4 分）】

63 阅读下列程序说明和程序，填充程序中的_____

【程序说明】本程序完成将二叉树中左、右孩子交换的操作。交换的结果如下所示(编者略)。本程序采用非递归的方法，设立一个堆栈 stack 存放还没有转换过的结点，它的栈顶指针为 tp。交换左、右子树的算法为：

（1）把根结点放入堆栈。

（2）当堆栈不空时，取出栈顶元素，交换它的左、右子树，并把它的左、右子树分别入栈。

（3）重复（2）直到堆栈为空时为止。

```
typedef  struct  node  *tree;
struct node{int data; tree lchild,rchild;}
exchange(tree t)
{tree  r,p; tree  stack [500]; int  tp=0;
 (1)_____
 while (tp>=0)
   {(2)_____
    if((3)_____)
       {r=p->lchild; p->lchild=p->rchild; p->rchild=r
        stack[(4)_____]=p->lchild; stack[++tp]=p->rchild;
        }
   }}
```
【中科院自动化研究所 1994 二、2 （15 分）】

64. 下面使用类 pascal 语言写的对二叉树进行操作的算法，请仔细阅读

```
TYPE pointer=^tnodetp;
     tnodetp=RECORD data: char; llink,rlink: pointer; END;
     linkstack=^linknodet;
     linknodet=RECORD data:pointer;  next;linkstack; END;
PROC unknown (VAR t:pointer);
VAR  p,temp: pointer;
BEGIN p:=t;
      IF p<> NIL THEN
        [temp:=p^.llink ; p^.llink:=p^.rlink;; p^.rlink:=temp;
```

```
            unknown(p^.llink); unknown(p^.rlink); ]
    END;
```

① 指出该算法完成了什么功能

② 用栈将以上算法改为非递归算法 unknown1,其中有若干语句或条件空缺请在空缺处填写上适当的语句或条件

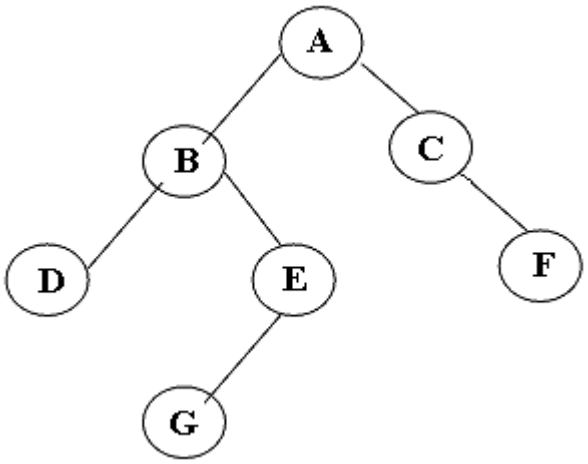
```
    PROC inistack(VAR s:linkstack);
    (1)_____; s^.next:=NIL;
    ENDP;
    FUNC empty (s:linkstack):boolean;
     IF (2)_____THEN empty:=true ELSE empty:=false;
    ENDF;
    FUNC gettop(s:linkstack):pointer;
      gettop:= (3)_____;
    ENDF;
    FUNC pop(VAR s:linkstack)：pointer;
    VAR p:linkstack;
        pop:=s^.next^.data; p:=s^.next; (4)_____; (5)_____;
    ENDF;
    PROC push (VAR s:linkstack;x:pointer);
    VAR  p:linkstack;
      new(p); p^.data:=x; (6)_____; s^.next:=p;
    ENDP;
    PROC unknown1(VAR t:pointer);
    VAR  p,temp: pointer; finish: boolean;
    BEGIN
       inistack(s); finish:=false;  p:=t;
       REPEAT
         WHILE p<> NIL DO
           [temp:=p^.llink; p^.llink:=p^.rlink; p^.rlink:=temp;
            (7)_____;  p:=p^.llink;    ];
            IF (8)____THEN [p:=gettop(s);temp;=pop(s);] ELSE (9)_____
       UNTIL (10)___
    ENDP;
```

【北方交通大学 2000 三、 (25 分)】

65. 具有 n 个结点的完全二叉树，已经顺序存储在一维数组 A[1..n]中，下面算法是将 A 中顺序存储变为二叉链表存储的完全二叉树。请填入适当的语句在下面的_____上，完成上述算法。

```
    TYPE ar=ARRAY[1..n] OF datatype;
         pointer=RECORD data:datatype; lchild, rchild: pointer; END;
    PROCEDURE  btree(VAR a: ar; VAR p:pointer);
       VAR i:integer;
       PROCEDURE  createtree(VAR t: pointer;i: integer)
        BEGIN  (1)_____; t^.data=a[i];
              IF(2)____THEN  creattree((3)_____) ELSE  t^.lchild:=NIL;
              IF(4)____THEN  createtree((5)_____) ELSE  t^.rchild:=NIL;
```

```
    END;
BEGIN
   j:= (6)  ;   createtree(p,j)
END;
```
【北京邮电大学 1998 五、(15 分)】



66. 设一棵二叉树的结点定义为

```
struct BinTreeNode{
  ElemType data; BinTreeNode *leftchild,*rightchild; }
```

现采用输入广义表表示建立二叉树。具体规定如下：

（1）树的根结点作为由子树构成的表的表名放在表的最前面；

（2）每个结点的左子树和右子树用逗号隔开。若仅有右子树没有左子树，逗号不能省略。

（3）在整个广义表输入的结尾加上一个特殊的符号（例如“#”）表示输入结束。

例如，对于如右图所示的二叉树，其广义表表示为 A（B（D，E（G）），C（，F））。

此算法的基本思路是：依次从保存广义表的字符串 ls 中输入每个字符。若遇到的是字母（假设以字母作为结点的值），则表示是结点的值，应为它建立一个新的结点，并把该结点作为左子女（当 k=1）或右子女（当 k=2）链接到其双亲结点上。若遇到的是左括号“(”，则表明子表的开始，将 k 置为 1；若遇到的是右括号“)”，则表明子表结束。若遇到的是逗号“,”，则表示以左子女为根的子树处理完毕，接着处理以右子女为根的子树，将 K 置为 2。在算法中使用了一个栈 s，在进入子表之前，将根结点指针进栈，以便括号内的子女链接之用。在子表处理结束时退栈。相关的栈操作如下：

MakeEmpty(s)	置空栈	Push(s,p)	元素 p 入栈
Pop(s)	退栈	Top(s)	存取栈顶元素的函数

下面给出了建立二叉树的算法，其中有 5 个语句缺失，请阅读此算法并把缺失的语句补上。（每空 3 分）

```
void CreatBinTree(BinTreeNode *&BT,char ls){
     Stack<BinTreeNode*>s;    MakeEmpty(s);
     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
    switch(ch){
       case '(' : (1)___; k=1; break;
       case ')' :  pop(s);     break;
       case',' : (2)___;       break;
     default                   :p=new                    BinTreeNode;
(3)____;p->leftChild=NULL;p->rightChild=NULL;
                  if(BT==NULL)    (4)_____;    else    if    (k==1)
                  top(s)->leftChild=p;
                  else top(s)->rightChild=p;
   }
(5)____; } }  【清华大学 2001 六、 (15 分)】
```

67. 判断带头结点的双向循环链表 L 是否对称相等的算法如下所示，请在划线处填上正确的语句

```
FUNCTION  equal(l:pointer) :boolean;
VAR p,q:pointer;  result: Boolean;
BEGIN result =true ; p:= l^.link; q:=l^.pre ;
      WHILE (p<>q) AND ((1)_______)DO
        IF  p^.data=q^.data THEN BEGIN (2)___; (3)____; END;
                            ELSE  result=false ;
      return(result);
END;   【华南师范大学 2000 年 五、1 ( 9 分)】
```

68. 下列是先序遍历二叉树的非递归子程序，请阅读子程序（C 语言与 PASCAL 语言过程功能完全相同，任选其一），填充空格，使其成为完整的算法。

C语言函数：

```
void example(b)
 btree  *b;
{ btree *stack[20], *p;
   int top;
   if (b!=null)
    { top=1; stack[top]=b;
      while (top>0)
       { p=stack[top]; top--;
         printf("%d",p->data);
         if (p->rchild!=null)
          {(1)___; (2)___;
          }
         if (p->lchild!=null)
          (3)___; (4)__;
   }}}}
```

PASCAL语言过程

```
PROCEDURE example(b:btree);
VAR stack:ARRAY[1..20] OF btree;
   top:integer; p:btree;
BEGIN
  IF b<>NIL  THEN
    BEGIN  top:=1; stack[top]:=b;
     WHILE top>0 DO
     BEGIN p:=stack[top];top:=top-1;
            write(p^.data);
            IF p^.rchild<>NIL
             THEN BEGIN
                  (1)__;(2)_; END;
            IF p^,lchild<>NIL
             THEN BEGIN  (3)__; 4)__;
END  END END END;
```

【同济大学 2001 三、 (10 分)】

69. 下述是一个由二叉树的前序序列和中序序列构造该二叉树的算法，其中，数组 A[1..n] 存放前序序列，数组 B[1..n]存放中序序列，s 为根结点指针，i,j 为树 s 的前序序列在 A[1..n] 中的开始位置和结束位置，x,y 为树 s 的中序序列在 B[1..n]中的开始位置和结束位置。所生成的二叉树采用二叉链表存储结构，其结点的形式为（lchild,data,rchild）。请在算法

的空框中填入适当语句，使其成为一个完整的算法。

```
PROCEDURE creatBT(i,j,x,y: integer; VAR s: link);
VAR  k,L: integer;
BEGIN s:= NIL;
      IF(1)__THEN
        BEGIN new (s); s^.data:=a[i]; k:=x;
              WHILE(2)_____DO  k:=k+1;
              L:= (3)____;
              IF k=x THEN s^.lchild:=NIL; ELSE(4)_____;
              IF k=y THEN s^.rchild:=NIL; ELSE(5)_____;
        END
END;
```

【西安交通大学 1996 五、1 (9 分)】

70. 已知中序遍历 bt 所指二叉树算法如下，s 为存储二叉树结点指针的工作栈，请在划线处填入一条所缺语句。

```
PROC inorder (bt:bitreptr);
  inistack(s); (1)_____;
  WHILE NOT empty(s)  DO
   [WHILE gettop(s)<>NIL DO push(s,gettop(s)↑.lchild);  (2)_____;
     IF NOT empty(s) THEN  [visit (gettop(s)^); p:=pop(s); (3)_____ ] ]
ENDP;{inorder}
```

【北京轻工业学院 1999 一、 (9 分)】

71. 以下程序是二叉链表树中序遍历的非递归算法，请填空使之完善。二叉树链表的结点类型的定义如下：

```
typedef struct node   /*C 语言/
 {char data; struct  node *lchild,*rchild;}*bitree;
void vst(bitree bt)                    /*bt 为根结点的指针*/
 { bitree  p; p=bt;  initstack(s);     /*初始化栈 s 为空栈*/
  while(p || !empty(s))               /*栈 s 不为空*/
   if(p) { push (s,p); (1)___; }      /*P 入栈*/
   else { p=pop(s); printf("%c",p->data); (2)___; }  /*栈顶元素出栈*/
} 
```

【西南交通大学 2000 一、10】

72. 二叉树存储结构同上题，以下程序为求二叉树深度的递归算法，请填空完善之。

```
 int depth(bitree bt)   /*bt 为根结点的指针*/
 {int hl,hr;
  if (bt==NULL) return((1)___);
  hl=depth(bt->lchild); hr=depth(bt->rchild);
  if((2)___)  (3)_____;
  return(hr+1);
 }
```

【西南交通大学 2000 一、11】

73. n 个结点的完全二叉树存储在数组 a 中，下面为非递归的先序遍历算法。

```
PROC preorder(a);
 BEGIN top:=0; t:=1;
      WHILE  (t<=n) OR (1)___DO
        BEGIN WHILE t<=n DO BEGIN write(a[t]); top:=top+1; s[top]:=t; t:=
(2) ;END;
```

```
                IF top>0 THEN BEGIN t:=s[top]*2+1; top:= (3)  ; END;
          END;
    END; 【中山大学 1998 四、3 (6分)】
```

74. 后序遍历二叉树的非递归算法，bt 是二叉树的根，S 是一个栈，maxsize 是栈的最大容量。

```
TYPE bitreptr=^bnodetp;
     bnodetp=RECORD data:datatype; lchild,rchild:bitreptr END;
TYPE stacktyp=RECORD data:ARRAY[1..maxsize] OF bitreptr; top: 0..maxsize; END;
PROCEDURE posterorder (bt: bitreptr);
BEGIN S.top:=0; p:=bt;
  REPEAT
    WHILE p<>NIL DO BEGIN S.top:=S.top+1; IF S.top>maxsize THEN stackfull
                                          ELSE BEGIN S.data[S.top]:=p;  (1)  ;
END
                      END;
    IF S.data[S.top]^.rchild<>NIL THEN  (2)
       ELSE BEGIN REPEAT write (S.data[S.top]^.data); S.top=S.top-1;
                  UNTIL S.top=0 OR S.data[S.top]^.rchild<>S.data[S.top+1];
                  IF S.data[S.top]^.rchild<>S.data[S.top+1] THEN (3)  ;
            END;
  UNTIL(4)  ;
END;  【西北工业大学 1999 六、1 (7分)】
```

75. 由二叉树的前序遍历和中序遍历序列能确定唯一的一棵二叉树，下面程序的作用是实现由已知某二叉树的前序遍历和中序遍历序列，生成一棵用二叉链表表示的二叉树并打印出后序遍历序列，请写出程序所缺的语句。

```
#define  MAX 100
typedef struct Node
 {char info; struct Node *llink, *rlink; }TNODE;
 char pred[MAX],inod[MAX];
 main(int argc,int **argv)
{ TNODE  *root;
  if(argc<3) exit 0;
  strcpy(pred,argv[1]); strcpy(inod,argv[2]);
  root=restore(pred,inod,strlen(pred));
  postorder(root);
}
TNODE *restore(char *ppos,char *ipos,int n)
{ TNODE  *ptr;  char *rpos;  int  k;
  if(n<=0) return NULL;
  ptr->info=(1)_____;
  for((2)_____ ; rpos<ipos+n;rpos++)  if(*rpos==*ppos) break;
  k=(3)_____;
  ptr->llink=restore(ppos+1, (4)_____,k );
  ptr->rlink=restore ((5)_____+k,rpos+1,n-1-k);
```

```
return ptr;
}
postorder(TNODE*ptr)
{ if(ptr=NULL) return;
  postorder(ptr->llink);  postorder(ptr->rlink);  printf("%c",ptr->info);
 }
```
【中科院计算所 2000 三、 (10 分)】

76. 已给如下关于二叉树的类型说明：

```
TYPE tree=^node ;
     node=RECORD data :integer; left ,right:tree END;
```

以下过程实现对二叉树 t 前序遍历的非递归算法：

```
PROCEDURE preorder(t:tree );
VAR  stack: ARRAY [1..100] OF tree; nd: tree; top: integer;
BEGIN top:=1; stack[top]:=t;
  WHILE(1)___ DO
    BEGIN nd:=stack[top];top:=top -1; write (nd^.data);
          IF (nd^.right<>NIL) THEN BEGIN top:=top +1; (2)___  END;
          IF (3)___THEN BEGIN  (4)___ ; stack[top]:= nd^.left; END
    END
 END;
```

【厦门大学 2000 三、1 （8 分)】

77. 下面是中序线索树的遍历算法，树有头结点且由指针 thr 指向。树的结点有五个域，分别为数据域 data，左、右孩子域 lchild、rchild 和左、右标志域 ltag,rtag。规定，标志域为 1 是线索，0 是指向孩子的指针。

```
inordethread(thr)
 {p=thr->lchild;
  while ((1)_____)
    { while((2)_____)  p= (3)___;
      printf(p->data);
       while((4)_____) { p=(5)___; printf(p->data);}
    p= (6) ;}
}
```

【南京理工大学 2000 三、1（6 分)】

78. 下面的算法在中序线索树中找由指针所指结点的后继并由指针指向该后继结点，试补充完整（线索树的结点有五个域 data,lchild,rchild,左、右标志域 ltag、rtag，并规定标志 0 指向孩子，1 指向线索。

```
PROC  inorder_next(p);
 (1)__ ;
  IF p ^.rtag=0 THEN  WHILE(2)___DO  q:= (3)___;
  return(q)
ENDP;
```

【南京理工大学 1998 三、1 （6 分)】

79. 线索二叉树有数据域 data，左右孩子域 lchild 和 rchild，左右标志 ltag 及 rtag，规定标志为 1 对应的孩子域是线索，0 则为指向孩子的指针。规定在储存线索二叉树时，完成下面中序线索化过程。(存储线索二叉树，不增加头结点，只在原有的由 tree 指向的二叉树中增加线索，此处也不考虑 c 语言的具体语法与约定，线索化前所有的标志 tag 都是 0)。

```
/* pre 是同 tree 类型相同的指针，初值是 null */
```

```
thread_inorder (tree)
 { if(tree!=null)
     { thread_inorder((1)____);
       if(tree->lchild==(2)_____) { tree->ltag=1; tree->lchild=pre; }
       if((3)____ == null){ (4)_____; (5)_____;}
        pre=p;  thread-inorder((6)_____);
     }
} 【南京理工大学 2001 三、5 （6 分)】
```

80. 如下的算法分别是后序线索二叉树求给定结点 node 的前驱结点与后继结点的算法，请在算法空格处填上正确的语句。设线索二叉树的结点数据结构为(lflag,left,data,right,rflag)，其中： lflag=0，left 指向其左孩子，lflag=1，left 指向其前驱；rflag=0，right 指向其右孩子，rflag=1，right 指向其后继。

```
prior(node,x)
 { if (node !=null)
   if ((1)_____ ) *x=node->right; else *x=node->left;
  }
next(bt, node, x)     /*bt 是二叉树的树根*/
  {(2)____;
   if (node!=bt  &&  node!=null)
     if  (node->rflag)  (3)_____ ;
      else {do t=*x; (4)_____;while  (*x==node); *x=t; }
   } 【南京航空航天大学 1996 十、 (8 分)】
```

四、应用题

1. 从概念上讲，树，森林和二叉树是三种不同的数据结构，将树，森林转化为二叉树的基本目的是什么，并指出树和二叉树的主要区别。【西安电子科技大学 2001 软件 二、1（5 分)】

2. 树和二叉树之间有什么样的区别与联系？

【西北工业大学 1998 一、3(4 分)】【厦门大学 2000 五、2(14%/3 分)】【燕山大学 2001 三、1(5 分)】

3. 请分析线性表、树、广义表的主要结构特点，以及相互的差异与关联。

【大连海事大学 2001 三(10 分)】

4. 设有一棵算术表达式树，用什么方法可以对该树所表示的表达式求值？

【中国人民大学 2001 二、3(4 分)】

5. 将算术表达式((a+b)+c*(d+e)+f)*(g+h)转化为二叉树。【东北大学 2000 三、1 （4 分)】

6. 一棵有 n(n>0)个结点的 d 度树，若用多重链表表示，树中每个结点都有 d 个链域，则在表示该树的多重链表中有多少个空链域？为什么？【长沙铁道学院 1998 四、1 (6 分)】

7. 一棵二叉树中的结点的度或为 0 或为 2，则二叉树的枝数为 2(n0-1)，其中 n0 是度为 0 的结点的个数。

【南京理工大学 1998 六、 (3 分)】

类似本题的另外叙述有：

（1）若二叉树中度为 1 的结点数为 0，则该二叉树的总分支数为 2（n0-1)，其中 n0 为叶结点数。

【西北工业大学 1998 三、1（5 分）】

8．一个深度为 L 的满 K 叉树有以下性质：第 L 层上的结点都是叶子结点，其余各层上每个结点都有 K 棵非空子树，如果按层次顺序从 1 开始对全部结点进行编号，求：

1）各层的结点的数目是多少？ 2）编号为 n 的结点的双亲结点（若存在）的编号是多少？

3）编号为 n 的结点的第 i 个孩子结点（若存在）的编号是多少？

4）编号为 n 的结点有右兄弟的条件是什么？如果有，其右兄弟的编号是多少？

请给出计算和推导过程。【西北工业大学 1999 五(10 分)】【中科院自动化所 1996 二、1(10 分)】

类似本题的另外叙述有：

（1）一棵高度为 h 的满 k 叉树有如下性质：根据结点所在层次为 0；第 h 层上的结点都是叶子结点；其余各层上每个结点都有 k 棵非空子树，如果按层次自顶向下，同一层自左向右,顺序从 1 开始对全部结点进行编号，试问：

1)各层的结点个数是多少?(3 分) 2)编号为 i 的结点的双亲结点（若存在）的编号是多少?(3 分)

3)编号为 i 的结点的第 m 个孩子结点（若存在）的编号是多少？（3 分）

4)编号为 i 的结点有右兄弟的条件是什么？其右兄弟结点的编号是多少？（3 分）

【清华大学 1999 八 (12 分)】

9．证明任一结点个数为 n 的二叉树的高度至少为 O(logn).【浙江大学 2000 四、 (5 分)】

10．有 n 个结点并且其高度为 n 的二叉树的数目是多少?

【西安电子科技大学 2000 计应用一、3(5 分)】

11．已知完全二叉树的第七层有 10 个叶子结点，则整个二叉树的结点数最多是多少？

【西安电子科技大学 2000 计应用 一、4 （5 分)】

12．高度为 10 的二叉树，其结点最多可能为多少？【首都经贸大学 1998 一、1 （4 分)】

13．任意一个有 n 个结点的二叉树，已知它有 m 个叶子结点，试证明非叶子结点有(m-1)个度为 2，其余度为 1。【西安电子科技大学 2001 计应用 二、3 （5 分)】

14. 已知 A[1..N]是一棵顺序存储的完全二叉树，如何求出 A[i]和 A[j]的最近的共同祖先？

【中国人民大学 2001 二、5 （4 分)】

15．给定 K(K>=1),对一棵含有 N 个结点的 K 叉树（N＞ 0)、请讨论其可能的最大高度和最小高度。

【大连海事大学 2001 五、（8 分)】

16．已知一棵满二叉树的结点个数为 20 到 40 之间的素数，此二叉树的叶子结点有多少个？

【东北大学 1999 一、1 （3 分)】

17．一棵共有 n 个结点的树，其中所有分支结点的度均为 K，求该树中叶子结点的个数。

【东北大学 2000 一、3 （4 分)】

18． 如在内存中存放一个完全二叉树，在树上只进行下面两个操作：

（1）寻找某个结点双亲 （2）寻找某个结点的儿子；

请问应该用何种结构来存储该二叉树？【东北大学 2001 一、3 （3 分)】

19.求含有 n 个结点、采用顺序存储结构的完全二叉树中的序号最小的叶子结点的下标。要求写出简要步骤。【北京工业大学 2000 二、3 （ 5 分)】

20．设二叉树 T 中有 n 个顶点，其编号为 1，2，3，…,n,若编号满足如下性质：

（1）T 中任一顶点 v 的编号等于左子树中最小编号减 1；

（2）对 T 中任一顶点 v,其右子树中最小编号等于其左子树中的最大编号加 1。试说明对二叉树中顶点编号的规则（按何种顺序编号）。【山东大学 1992 一、1 （3 分）】

21．若一棵树中有度数为 1 至 m 的各种结点数为 $n_1,n_2,\cdots,n_m$(n_m表示度数为 m 的结点个数)请推导出该树中共有多少个叶子结点 n_0的公式。【北京邮电大学 1993 二 1(6 分)】【西安交通大学 1996 四、1(5 分)】【南京航空航天大学 1998 五(10 分)】【东南大学 1999 一 4（8 分)】【山东大学 1993 一 2(4 分)】

【山东师范大学 2001 二 3(12 分)　2001 二 2(15 分)】

22．若一棵完全二叉树中叶子结点的个数为 n,且最底层结点数≧2，则此二叉树的深度 H=?

【北京科技大学 2001 一、6 （2 分)】

23．已知完全二叉树有 30 个结点，则整个二叉树有多少个度为 0 的结点?

【山东师范大学 1996 五、3(2 分)】

24．在一棵表示有序集 S 的二叉搜索树(binary search tree)中,任意一条从根到叶结点的路径将 S 分为 3 部分：在该路径左边结点中的元素组成的集合 S1；在该路径上的结点中的元素组成的集合 S2；在该路径右边结点中的元素组成的集合 S3。S=S1∪S2∪S3。若对于任意的 a∈S1,b∈S2，c∈S3 是否总有 a≤b≤c?为什么?【清华大学 2000 四(10 分)】【武汉大学 2000 三、3】

25．试证明,在具有 n(n>=1)个结点的 m 次树中,有 n(m-1)+1 个指针是空的。【复旦大学 1998 四(8 分)】

26．对于任何一棵非空的二叉树,假设叶子结点的个数为 n_0,而次数为 2 的结点个数为 n_2,请给出 n_0和 n_2之间所满足的关系式 $n_0=f(n_2)$.要求给出推导过程。【复旦大学 1998 五 (8 分)】

27．对于任意一棵非空的二叉树 T，我们用 n_0表示 T 中叶子结点的个数，用 n_2表示 T 中有两棵非空子树的结点的个数。（1）给出 n_0和 n_2所满足的关系式。（2）证明你在（1）中给出的关系式成立。

【复旦大学 1997 三 (10 分)】

28．试求有 n 个叶结点的非满的完全二叉树的高度;【中科院计算所 2000 五、 (5 分)】

29．对于具有 n 个叶子结点，且所有非叶子结点都有左右孩子的二叉树，

(1)试问这种二叉树的结点总数是多少？（5 分）

(2)试证明 $\sum_{i=1}^{n} 2^{-(l_i-1)}=1$。其中:$l_i$表示第 i 个叶子结点所在的层号(设根结点所在层号为 1)。(10 分)

【北方交通大学 1995 三、 (15 分)】

30．假设高度为 H 的二叉树上只有度为 0 和度为 2 的结点，问此类二叉树中的结点数可能达到的最大值和最小值各为多少？【北京邮电大学 1996 一、1 （4 分)】

31．一棵满 k 叉树，按层次遍历存储在一维数组中，试计算结点下标的 u 的结点的第 i 个孩子的下标以及结点下标为 v 的结点的父母结点的下标。【北京邮电大学 2001 四、4(5 分)】

32．二叉树有 n 个顶点，编号为 1，2，3，… ，n，设：

＊ T 中任一顶点 V 的编号等于左子树中最小编号减 1；

＊ T 中任一顶点 V 的右子树中的最小编号等于其左子树中的最大编号加 1；

试描绘该二叉树。【东南大学 1999 一、2 （7 分)】

33. 设 T 是具有 n 个内结点的扩充二叉树，I 是它的内路径长度，E 是它的外路径长度。

（1）试利用归纳法证明 E=I+2n,　n>=0.（5 分）

（2）利用（1）的结果试说明：成功查找的平均比较次数 s 与不成功查找的平均比较次数 u 之间的关系可用公式表示 s=(1+1/n)u-1,n>=1。【清华大学 1998 四、(10 分)】

34. 一棵非空的有向树中恰有一个顶点入度为 0,　其它顶点入度为 1,但一个恰有一个顶点入度为 0,其它顶点入度为 1 的有向图却不一定是一棵有向树,请举例说明。【中科院计算所 1999 三、1 （5 分)】

35. 试给出下列有关并查集（mfsets）的操作序列的运算结果：

union(1,2),union(3,4),union(3,5),union(1,7),union(3,6),union(8,9),

union(1,8),union(3,10),union(3,11),union(3,12),union(3,13),

union(14,15),union(16,0),union(14,16),union(1,3),union(1,14).

(union 是合并运算，在以前的书中命名为 merge)

要求

（1）对于 union(i,j),以 i 作为 j 的双亲；　　（5 分）

（2）按 i 和 j 为根的树的高度实现 union(i,j),高度大者为高度小者的双亲；　　（5 分）

（3)按 i 和 j 为根的树的结点个数实现 union(i,j),结点个数大者为结点个数小者的双亲。(5 分)

【清华大学 2001 一、 (15 分)】

36. 证明：在任何一棵非空二叉树中有下面的等式成立：叶结点的个数=二度结点的个数+1

【天津大学 1999 四 】

37. 对于一个堆栈，若其入栈序列为 1，2，3，…，n，不同的出入栈操作将产生不同的出栈序列。其出栈序列的个数正好等于结点个数为 n 的二叉树的个数，且与不同形态的二叉树一一对应。请简要叙述一种从堆栈输入（固定为 1，2，3……n）/输出序列对应一种二叉树形态的方法，并以入栈序列 1，2，3（即 n=3）为例加以说明。【浙江大学 1998 年 五、1 (7 分)】

38. 如果给出了一个二叉树结点的前序序列和对称序序列,能否构造出此二叉树?若能，请证明之。若不能，请给出反例。如果给出了一个二叉树结点的前序序列和后序序列，能否构造出此二叉树?若能，请证明之。若不能，请给出反例。【北京大学 1998 二、2 （5 分)】

类似本题的另外叙述有：

（1）二叉树的中序与后序序列能唯一地定义一棵二叉树吗? 这里所指序列中的符号代表树结点中的标识符吗？二叉树的前序与后序序列能唯一地定义一棵二叉树吗?为什么?【东南大学 1993 一、4（8 分)】

39. 试证明:同一棵二叉树的所有叶子结点，在前序序列。对称序序列以及后序序列中都按相同的相对位置出现（即先后顺序相同)，例如前序 abc,后序 bca 对称序 bac。【山东工业大学 1997 七、 (10 分)】

40. 由二叉树的中序序列及前序序列能唯一的建立二叉树,试问中序序列及后序序列是否也能唯一的建立二叉树，不能则说明理由，若能对中序序列 DBEAFGC 和后序序列 DEBGFCA 构造二叉树。

【南京理工大学 1998 四、 (3 分)】

41. 证明，由一棵二叉树的前序序列和中序序列可唯一确定这棵二叉树。设一棵二叉树的前序序列为 ABDGECFH，中序序列为：DGBEAFHC 。试画出该二叉树。【浙江大学 1996 六、(8 分)】

类似本题的另外叙述有：

(1) 证明：由一棵二叉树的前序序列和中序序列可唯一确定这棵二叉树。

【长沙铁道学院 1997 五、2(10 分)】

（2）证明：由二叉树的中序遍历序列和后序遍历序列可唯一地确定出该二叉树。

【华南理工大学 2001 一、3 （4 分)】

（3）二叉树已知其中序扫描序列和后序扫描序列如何确定这一棵二叉树,并举例说明.【山东大学 2001 软件与理论 二 、1 （4 分)】

42. 试证明：仅仅已知一棵二叉树的后序遍历序列和先序遍历序列，不能唯一地确定这棵二叉树。

【大连海事大学 2001 九、（8 分)】

类似本题的另外叙述有：

(1) 由二叉树的前序遍历和后序遍历结果能否唯一确定一棵二叉树？解释你的论断。【西安电子科技大学 2001 计应用 二、4 （5 分)】

(2) 假定某二叉树的前序遍历序列为 ABCDEFGHIJ,后序遍历序列为 CEFDBJIHGA,据此两个序列能否唯一确定此二叉树? 若不能,试画出两样具有同样上述遍历序列的二叉树.【武汉交通科技大学 1996 二 8（3 分)】

43. ① 试找出满足下列条件的二叉树

1）先序序列与后序序列相同 2）中序序列与后序序列相同

3）先序序列与中序序列相同 4）中序序列与层次遍历序列相同

② 已知一棵二叉树的中序序列和后序序列分别为 DBEAFIHCG 和 DEBHIFGCA，画出这棵二叉树。

【东北大学 1999 六、 (4 分)】

类似本题的另外叙述有：

（1）试画出在先根次序和中根次序下结点排列顺序皆相同的所有类型的二叉树形。

试画出在先根次序和后根次序下结点排列顺序皆相同的所有类型的二叉树形。

【吉林大学 1995 四、1,2 （每题 7 分)】

（2）找出所有的二叉树，其结点在下列两种遍历下恰好都有同样的遍历序列。

1）先序遍历和中序遍历 2）先序遍历和后序遍历【北京理工大学 1999 三(6 分)】

（3）找出所有的二叉树，其结点在下列两种遍历下，恰好都是以同样的顺序出现：

1）前序遍历和中序遍历。 2）前序遍历和后序遍历。【南京航空航天大学 1995 六(5 分)】

（4）试找出分别满足下列条件的所有二叉树。

1）先序序列和中序序列相同 2）中序序列和后序序列相同

3）先序序列和后序序列相同 【南京航空航天大学 2001 二、(10 分)】

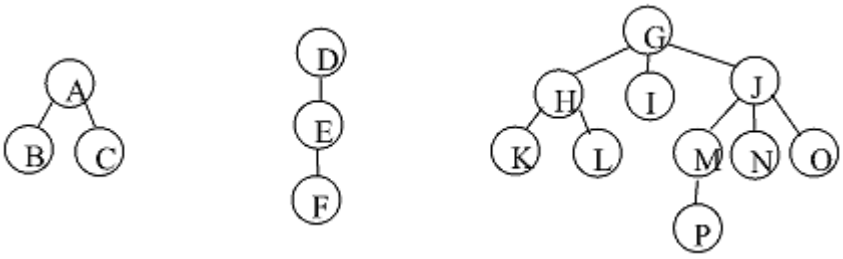
（5）找出所有满足下列条件的二叉树：

1）它们在先序遍历和中序遍历时，得到的结点访问序列相同；

2）它们在后序遍历和中序遍历时，得到的结点访问序列相同；

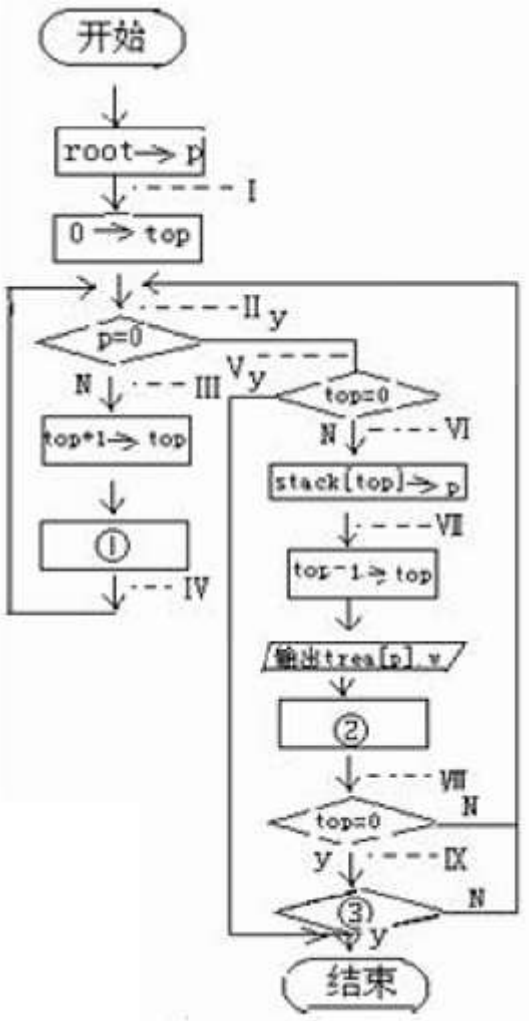
3）它们在先序遍历和后序遍历时，得到的结点访问序列相同;【东南大学 2000 一、4(6 分)】

44. 将下列由三棵树组成的森林转换为二叉树。(只要求给出转换结果)



【南京航空航天大学 1998 一、 (10 分)】

45. 阅读下列说明和流程图，回答问题（1）和问题（2）。



说明：流程图是用来实现中序遍历，二叉树存放在数组 tree 中，每个数组元素存放树中一个结点，每个结点的形式为(值，左指针，右指针)，分别用 tree[i].v，tree[i].l，tree[i].r 来表示第 i 个结点的值，左指针，右指针，其中左，右指针的值为所指结点在数组中的下标，若指针的值为 0，表示它指向空树，图中指针 root 用以指向二叉树的根结点。问题：

（1）填充流程图中的①、②、③，使其按中序遍历二叉树。

（2）把流程图中的（A）框移至哪个位置（图中Ⅰ～Ⅸ）使流程图的算法从中序遍历变成后序遍历。

【上海海运学院 1997 年 四、(13 分)】

46. 设一棵二叉树的先序、中序遍历序列分别为

先序遍历序列： A B D F C E G H 中序遍历序列： B F D A G E H C

（1）画出这棵二叉树。

（2）画出这棵二叉树的后序线索树。

（3）将这棵二叉树转换成对应的树（或森林）。【南京航空航天大学 1997 二、 (10 分)】

47. 已知一棵二叉树的对称序和后序序列如下：

对称序：GLDHBEIACJFK　后序：　LGHDIEBJKFCA

(1) (2 分)给出这棵二叉树：

(2) (2 分)转换为对应的森林：

(3) (4 分)画出该森林的带右链的先根次序表示法：

$$Itag=\begin{cases}1---\text{无左子女}\\2---\text{有左子女}\end{cases}$$

(4) (4 分) 画出该森林带度数的后根次序表示法：

(5) (4 分)在带度数的后根次序表示法中，不包含指针，但仍能完全反映树的结构。写出以结点 x 为根的子树在后根次序序列中的前驱的求法。(用语言叙述，不用写算法)【山东大学 1998 八、(16 分)】

48. 设某二叉树的前序遍历序列为:ABCDEFGGI，中序遍历序列为:BCAEDGHFI：

（1）试画出该二叉树；

（2）写出由给定的二叉树的前序遍历序列和中序遍历序列构造出该二叉树的算法。

（3）设具有四个结点的二叉树的前序遍历序列为 a b c d； S 为长度等于四的由 a，b，c，d 排列构成的字符序列，若任取 S 作为上述算法的中序遍历序列，试问是否一定能构造出相应的二叉树，为什么？试列出具有四个结点二叉树的全部形态及相应的中序遍历序列。

【浙江大学 1997 六、 (15 分)】

<u>类似本题的另外叙述有：</u>

（1）已知二叉树的先序序列： CBHEGAF,　中序序列: HBGEACF, 试构造该二叉树

【北京理工大学 2001 八、2 （4 分)】

（2）已知二叉树按中序排列为 BFDAEGC，按前序排列为 ABDFCEG，要求画出该二叉树。

【山东师范大学 1996　五、1 (2 分)】

（3）已知一棵二叉树的前序序列 A,B,D,C,E,F，中序序列 B,D,A,E,F,C. 画出这棵二叉树。

【燕山大学 1999 四、 (5 分)】

（4）已知一棵二叉树的前序遍历结果是：ABCDEFGHIJ,中序遍历的结果是：BCEDAGHJIF,试画出这棵二叉树。【厦门大学 1998 六、1 （7 分)】

（5）已知二叉树 BT 各结点的先序、中序遍历序列分别为 ABCDEGF 和 CBAEDF，试画出该二叉树。

【北京工业大学 1998 二、 (6 分)】

49. 假设一棵二叉树的前序序列为 ABCD,它的中序序列可能是 DABC 吗?【石油大学 1998 一、1(5 分)】

<u>类似本题的另外叙述有：</u>

（1）一棵前序序列为 1，2，3，4，的二叉树，其中序序列可能是 4，1，2，3 吗？设一棵二叉树的前序序列为 1，2，3，4，5，6，7，8，9，其中序序列为 2，3，1，5，4，7，8，6，9，试画出该二叉树。

【东南大学 1996 一、2 （7 分)　1998 一、3】

50. 一棵非空的二叉树其先序序列和后序序列正好相反，画出这棵二叉树的形状。

【西安电子科技大学 2000 软件 一、8 （5 分)】

51. 已知一棵二叉树的后序遍历序列为 EICBGAHDF，同时知道该二叉树的中序遍历序列为 CEIFGBADH，试画出该二叉树。【重庆大学 2000 二、2】

类似本题的另外叙述有：

（1）已知二叉树ＢＴ各结点的中序和后序序列分别为ＤＦＢＡＣＥＧ和ＦＤＢＧＥＣＡ，试构造出该二叉树ＢＴ，并作简要说明。【北方交通大学 1997 二、 (8 分)】

（2）已知二叉树的中序遍历序列为ＧＦＢＥＡＮＨＭ，后序遍历的结点序列为ＧＥＢＦＨＮＭＡ，画出此二叉树的形态。【青岛海洋大学 1999 一、5（5 分)】

（3）已知二叉树的后序序列为 ABCDEFG 和中序序列为 ACBGEDF，构造出该二叉树。

【福州大学 1998 三、1 (6 分)】

（4）已知某二叉树的后序遍历和中序遍历如下，构造出该二叉树。

后序遍历序列： ＧＤＢＥＩＨＦＣＡ 中序遍历序列：ＤＧＢＡＥＣＨＩＦ

【厦门大学 2000 七、1 （20%/3 分)】

（5）已知一个二分树的中序序列和后序序列如下：

中序：ＡＢＣＤＥＦＧＨＩＪ 后序：ＡＣＤＢＨＪＩＧＦＥ

试画出此二分树的结构。 【首都经贸大学 1998 二、1 （10 分)】

52. 假设一棵二叉树的层次序列为 ABCDEFGHIJ，中序序列 DBGEHJACIF。请画出这棵二叉树。

【武汉大学 2000 三、1】【东南大学 2000 一、1 （6 分)】

类似本题的另外叙述有：

（1）假设一棵二叉树的层次次序（按层次递增顺序排列，同一层次自左向右）为 ABECFGDHI,中序序列为 BCDAFEHIG。请画出该二叉树，并将其转换为对应的森林。【山东大学 2001 四、 (6 分)】

53. 已知一个森林的先序序列和后序序列如下，请构造出该森林。

先序序列：ABCDEFGHIJKLMNO

后序序列：CDEBFHIJGAMLONK 【合肥工业大学 2000 四、1 （5 分)】

54. 画出同时满足下列两条件的两棵不同的二叉树。

(1)按先根序遍历二叉树顺序为 ABCDE。

(2)高度为 5 其对应的树（森林）的高度最大为 4。【东北大学 1997 一、3 （5 分)】

55. 用一维数组存放的一棵完全二叉树；ABCDEFGHIJKL。请写出后序遍历该二叉树的访问结点序列。

【西安电子科技大学 1999 计应用 一、6 （5 分)】

56. 一棵二叉树的先序、中序、后序序列如下，其中一部分未标出，请构造出该二叉树。

先序序列 ：＿＿ＣＤＥ＿ＧＨＩ＿Ｋ

中序序列 ：ＣＢ＿＿ＦＡ＿ＪＫＩＧ

后序序列 ：＿ＥＦＤＢ＿ＪＩＨ＿Ａ 【厦门大学 2002 七、1 （6 分)】

类似本题的另外叙述有：

（1）一棵二叉树的先序、中序和后序序列分别如下，其中有一部分为显示出来。试求出空格处的内容，并画出该二叉树。

先序序列: ＿Ｂ＿Ｆ＿ＩＣＥＨ＿Ｇ

中序序列：Ｄ＿ＫＦＩＡ＿ＥＪＣ＿

后序序列：＿Ｋ＿ＦＢＨＪ＿Ｇ＿Ａ 【西安电子科技大学 2000 计应用 五、2 （5 分)】

（2）已知一棵二叉树的先序 中序和后序序列如下，其中空缺了部分，请画出该二叉树。

先序：_ B C _ E F G _ I J K _

中序：C B E D _ G A J _ H _ L

后序：_ E _ F D _ J _ L _ H A 【合肥工业大学 2001 四、1 （5 分)】

（3）已知含有 8 个结点的一棵二叉树，按先序、中序、后序进行遍历后，有些结点序号不清楚如下图示。要求构造出一棵符合条件的二叉树。

先根序遍历 _ 2 3 _ 5 _ 7 8

中根序遍历 3 _ 4 1 _ 7 8 6

后根序遍历 _ 4 2 _ _ 6 5 1 【东北大学 1996 一、3 （5 分)】

57. M 叉树的前序和后序遍历分别与由它转换成的二叉树的哪种遍历相对应？

【中国人民大学 2000 一、2 （4 分)】

58. 证明：在二叉树的三种遍历序列中，所有叶子结点间的先后关系都是相同的。要求每步论断都指出根据。【北京工业大学 2001 二、3 （5 分)】

59. 下表中 M、N 分别是一棵二叉树中的两个结点，表中行号 i=1，2，3，4 分别表示四种 M、N 的相对关系，列号 j=1，2，3 分别表示在前序、中序、后序遍历中 M,N 之间的先后次序关系。要求在 i，j 所表示的关系能够发生的方格内打上对号。例如：如果你认为 n 是 m 的祖先，并且在中序遍历中 n 能比 m 先被访问，则在（3，2）格内打上对号

	先根遍历时n先被访问	中根遍历时n先被访问	后根遍历时n先被访问
N在M的左边			
N在M的右边			
N是M的祖先			
N是M的子孙			

【南京理工大学 2001 四、 (10 分)】

60. 用一维数组存放的一棵完全二叉树如下图所示：

A	B	C	D	E	F	G	H	I	J	K	L

写出后序遍历该二叉树时访问结点的顺序。

【北京工业大学 1996 一、4 （6 分)】

61. 设树形 T 在后根次序下的结点排列和各结点相应的次数如下：

后根次序：B D E F C G J K I L H A

次　　数：0 0 0 0 3 0 0 0 2 0 2 4

请画出 T 的树形结构图。 【吉林大学 2001 一、2 （4 分)】

62. 已知二叉树采用二叉链表方式存放，要求返回二叉树 T 的后序序列中的第一个结点的指针，是否可不用递归且不用栈来完成？请简述原因。【西北大学 2001 三 6】

63. 对于二叉树 T 的两个结点 n1 和 n2，我们应该选择树 T 结点的前序、中序和后序中哪两个序列来判断结点 n1 必定是结点 n2 的祖先,并给出判断的方法。不需证明判断方法的正确性。

【复旦大学 1999 五 (10 分)】

64. 设二叉树的存储结构如下(每题 5 分,共 15 分)

LINK	0	0	2	3	7	5	8	0	10	1
INFO	J	H	F	D	B	A	C	E	G	I
RLINK	0	0	0	9	4	0	0	0	0	0

其中,T 为树根结点的指针,LLINK、RLINK 分别指向结点的左右子女,INFO 为其数据域,请完成下列各题:

（1）画出二叉树 T 的逻辑结构. （2）写出按前序、中序和后序周游二叉树 T 得到的结点序列.
（3）画出二叉树 T 的后序线索树。 【山东工业大学 1995 六、(15 分)】

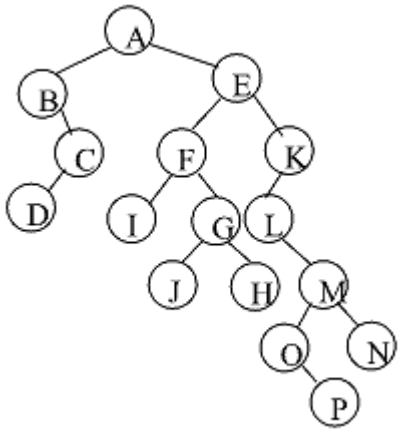
65. 在二叉树的前序遍历和中序遍历的递归算法中，最后一个递归调用语句在调用时所保留的参数有什么作用？如何清除最后这个递归语句？【北京邮电大学 1994 三、 (8 分)】

66. 在二叉树的 L link-Rlink 存储表示中，引入“线索”的好处是什么？
【山东大学 1999 六、 1 (2 分)】

67. 按下面要求解下图中二叉树的有关问题：
(1)对此二叉树进行后序后继线索化 ； (2)将此二叉树变换为森林；
(3)用后根序遍历该森林,；写出遍历后的结点序列。【北京邮电大学 1996 五、 (10 分)】

类似本题的另外叙述有：

（1)已知一棵二叉树的先序遍历序列为：AEFBGCDHIKJ，中序遍历序列为：EFAGBCHKIJD。试写出此二叉树的后序遍历序列，并用图画出它的后序线索二叉树。【同济大学 2000 一、(10 分)】



68. 对下图所示二叉树分别按前序、中序、后序遍历，给出相应的结点序列，同时给二叉树加上中序线索。
【青岛海洋大学 1999 年一、1 （5 分)】

69. 假设一个二叉树的两种遍历如下：
前序：ABFGCHDEIJLK 中序：FGBHCDILJKEA
（1）画出这棵二叉树以及它的中序线索树；
（2）写出在中序线索树的情况下，找到结点 N 的前驱结点的算法 INORDER-PRIOR(N,X)
【上海海运学院 1996 四、 (10 分)】

70. 已知一棵二叉树的中序（或中根）遍历结点排列为 DGBAECHIF，后序（或后根）遍历结点排列为 GDBEIHFCA，
（1）试画出该二叉树；
（2）试画出该二叉树的中序穿线（或线索）树；
（3）试画出该二叉树（自然）对应的森林；【吉林大学 2000 一、1 （5 分)】

71. 设二叉树 BT 的存储结构如下:

	1	2	3	4	5	6	7	8	9	10
Lchild	0	0	2	3	7	5	8	0	10	1
Data	J	H	F	D	B	A	C	E	G	I
Rchild	0	0	0	9	4	0	0	0	0	0

其中 BT 为树根结点的指针，其值为 6,Lchild,Rchild 分别为结点的左、右孩子指针域,data 为结点的数据域。试完成下列各题:

（1）画出二叉树 BT 的逻辑结构;

（2）写出按前序、中序、后序遍历该二叉树所得到的结点序列;

（3）画出二叉树的后序线索树。【中国矿业大学 2000 二、 (15 分)】

72. 请说明是否存在这样的二叉树，即它可以实现后序线索树进行后序遍历时不使用栈；而对前序线索树进行前序遍历时，又有什么样的二叉树可不使用栈。【西安电子科技大学 1996 二、1 (5 分)】

73. 一棵左右子树均不空的二叉树在先序线索化后，其空指针域数为多少？

【西安电子科技大学 2000 计应用 一、2 (5 分)】

74. 在前序线索树上，要找出结点 p 的直接后继结点，请写出相关浯句。结点结构为 (ltag,lc,data,rtag,rc)。【西北大学 2000 二、6 (5 分)】

75. 对于后序线索二叉树，怎样查找任意结点的直接后继；对于中序线索二叉树，怎样查找任意结点的直接前驱？【西北工业大学 1998 一、4 (4 分)】

76. 将下列树的孩子一兄弟链表改为后根遍历全线索链表。【清华大学 1994 二、 (10 分)】

Data	A	B	C	D	E	F	G	H	I	J	K
Ltag	0	0	0	0	0	0	0	0	0	0	0
Fch	2	0	5	7	8	0	11	0	0	0	0
Rtag	0	0	0	0	0	0	0	0	0	0	0
Nsib	0	3	4	0	6	0	0	9	10	0	0

77. 已知一棵二叉树的前序遍历为 ABECDFGHIJ，中序遍历为 EBCDAFHIGJ。试画出这棵树和它的中序线索树。假定用于通讯的电文仅有 8 个字母 C1，C2，…，C8 组成，各个字母在电文中出现的频率分别为 5，25，3，6，10，11，36，4，试为这 8 个字母设计哈夫曼编码树。【上海海运学院 1998 四(10 分)】

78. 设有正文 AADBAACACCDACACAAD,字符集为 A,B,C,D,设计一套二进制编码，使得上述正文的编码最短。【首都经贸大学 1997 一、5 (4 分)】

类似本题的另外叙述有：

（1）设有正文 MNOPPPOPMMPOPOPPOPNP，字符集为 M，N，O，P，设计一套二进制编码，使得上述正文的编码最短。【首都经贸大学 1998 一、5 (4 分)】

79. 给定集合{15,3,14,2,6,9,16,17}

(1)（3 分）用□表示外部结点，用○表示内部结点，构造相应的 huffman 树：

(2) (2 分)计算它的带权路径长度：

(3)（3 分)写出它的 huffman 编码：

(4)（3 分)huffman 编码常用来译码，请用语言叙述写出其译码的过程。

【山东大学 1998 七、】【山东工业大学 2000 七、 (11 分)】

类似本题的另外叙述有：

（1） 如果通信字符 a,b,c,d 出现频度分别为：7，5，2，4 请画出哈夫曼树并给出相应的哈夫曼编码。【青岛大学 2001 七、1 (5 分)】

（2）给定一组数列（15,8,10,21,6,19,3）分别代表字符 A,B,C,D,E,F,G 出现的频度，试叙述建立哈夫曼树的算法思想，画出哈夫曼树，给出各字符的编码值，并说明这种编码的优点。

【青岛大学 2000 七、 (10 分)】

（3）设通信中出现 5 中字符 A、B、C、D、E 对应的频率为 0.2,0.1,0.5,0.15,0.25 构造哈夫曼树，并给出对应字符的编码。【青岛大学 2002 四、2 (10 分)】

（4） 设 A、B、C、D、E、F 六个字母出现的概率分别为 7,19,2,6,32,3。试写出为这

六个字母设计的HUFFMAN编码，并画出对应的HUFFMAN树.【山东工业大学 1995 四(10分)】

（5）设用于通信的电文由 8 个字母组成，字母在电文中出现的频率分别为:7,19,2,6,32,3,21,10。试为这 8 个字母设计哈夫曼编码.使用 0-7 的二进制表示形式是另一种编码方案,试比较这两种方案的优缺点。【南京航空航天大学 1999 四、 (10 分)】

（6）假设用于通讯的电文由 8 个字符组成，其出现的频率为 5,29,7,8,14,23,3,11。试为这 8 个字符设计哈夫曼编码。【燕山大学 1999 五、 (5 分)】

（7）假设用于通信的电文由字符集{a,b,c,d,e,f,g}中的字母构成。它们在电文中出现的频度分别为{0.31,0.16,0.10,0.08,0.11,,0.20,0.04},

1) 为这 7 个字母设计哈夫曼编码；

2）对这 7 个字母进行等长编码，至少需要几位二进制数？哈夫曼编码比等长编码使电文总长压缩多少？【北京邮电大学 2001 四、2 （5 分)】

（8）试构造一棵二叉树,包含权为 1,4,9,16,25,36,49,64,81,100 等 10 个终端结点,且具有最小的加权路径长度 WPL。【北方交通大学 1993 年 五（12 分)】

（9）带权结点为{5,6,7,8,9}，构造 Huffman 树，计算带权路径长度。【西北大学 2001 年三、3】

（10）以数据集{2,5,7,9,13}为权值构造一棵哈夫曼树，并计算其带权路径长度。

【西安电子科技大学 1999 计应用 一、4 （5 分)】

（11）假设用于通讯的电文仅由 8 个字母组成，字母在电文中出现的频率分别为 7,19,2,6,32,3，21,10。试为这 8 个字母设计哈夫曼编码。使用 0∽7 的二进制表示形式是另一 种编码方案。对于上述实例，比较两种方案的优缺点。【大连海事大学 1996 五、2 (8 分)】。

（12）设用于通讯的电文仅由 8 个字母组成，他们在电文中出现的频率分别为 0.30，0.07，0.10，0.03，0.20，0.06,0.22，0.02，试设计哈夫曼树及其编码。使用 0---7 的二进制表示形式是另一种编码方案。给出两种编码的对照表、带权路径长度 WPL 值并比较两种方案的优缺点。【厦门大学 1999 三、3】

(13) 给定一组权值 2,3,5,7,11,13,17,19,23,29，31，37，41，试画出用 Huffman 算法建造的 Huffman 树。【吉林大学 2000 一、2 （4 分)】

(14) 以数据集{3,4,5,8,12,18,20,30}为叶结点，构造一棵哈夫曼树，并求其带权路径长度。【山东师范大学 1996 五 5(2 分)】

80. 给定权 W1,W2,…，Wm 。说明怎样来构造一个具有最小的加权路径长度的 k 叉树。试对于权 1,4,9,16,25, 36,49,64,81,100 来构造最优的三叉树，并给出其最小加权路径长度。【北方交通大学 1994 年 四(12 分)】

81. 已知下列字符 A、B、C、D、E、F、G 的权值分别为 3、12、7、4、2、8，11，试填写出其对应哈夫曼树 HT 的存储结构的初态和终态。【北京工业大学 1998 五、 (10 分)】

82. 什么是前缀编码？举例说明如何利用二叉树来设计二进制的前缀编码。【中山大学 1999 三、1 (3 分)】

83. 如果一棵 huffman 树 T 有 n_0个叶子结点，那么，树 T 有多少个结点，要求给出求解过程。

【复旦大学 1999 四、 (10 分)】

84. 设 T 是一棵二叉树，除叶子结点外，其它结点的度数皆为 2，若 T 中有 6 个叶结点，试问：

（1）T 树的最大深度 Kmax=?最小可能深度 Kmin=?

（2）T 树中共有多少非叶结点？

(3) 若叶结点的权值分别为 1,2,3,4,5,6。请构造一棵哈曼夫树，并计算该哈曼夫树

的带权路径长度 wpl。【北京邮电大学 1992 一、3】

85. 对如下算法，解答下列问题。

```
PROCEDURE inorder(T:bitree);
BEGIN top:=1; s[top]:=T;
  REPEAT
    WHILE s[top]<>NIL DO BEGIN s[top+1]:=s[top]^.lchild; top:=top+1; END;
    IF top>1 THEN BEGIN top:=top-1;WRITE
(s[top]^.data);s[top]:=s[top]^.rchild;END;
  UNTIL top=0
END;
```

（1）该算法正确吗？循环结束条件 top=0 能否满足？

（2）若将 IF top>1…改为 IF top>0…是否正确？

（3）若将结束条件改为 top=1,其它不变，是否正确？

（4）若仅将结束处条件改为(top=1)AND (s[top]=NIL),是否正确？

（5）试找出二叉树中各结点在栈中所处层次的规律。 【西安电子科技大学 2000 计应用 三(10 分)】

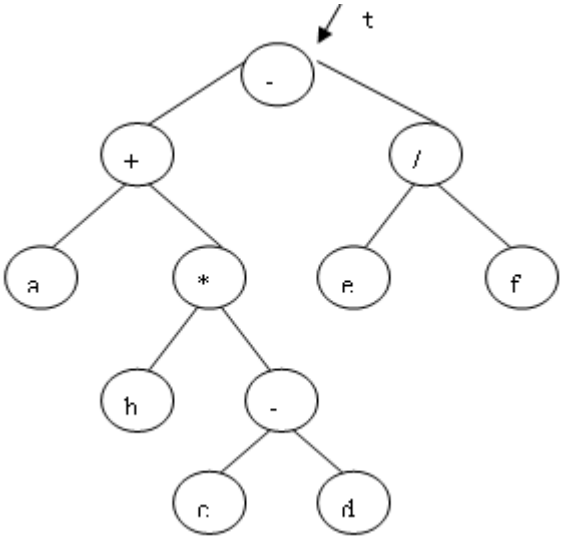
五、算法设计题

1. 假设一个仅包含二元运算符的算术表达式以链表形式存储在二叉树 BT 中，写出计算该算术表达式值的算法。【东北大学 2000 三、2 （10 分)】

2. 给出算法将二叉树表示的表达式二叉树按中缀表达式输出，并加上相应的括号。【北京邮电大学 2001 五、3 （10 分)】

3.（此题统考生做） 用 PASCAL 语言（或类 PASCAL 语言）完成下列各题：

（1）设表达式 a+b*(c-d)-e/f 可以表示成如下二叉树结构：



其中 t 为根结点指针，试运用后序遍历二叉树的规则，写出对表达式求值的算法：EXPVALUE【北京科技大学 1998 年 八、1 （10 分)】

4. 编程求以孩子—兄弟表示法存储的森林的叶子结点数。要求描述结构。【北京工业大学 2000 五(10 分)】

5. 假定用两个一维数组 L[N]和 R[N]作为有 N 个结点 1，2,…， N 的二元树的存储结构。L[i]和 R[i]分别指示结点 i 的左儿子和右儿子；L[i]=0（R[i]=0）表示 i 的左（右）儿子

为空。试写一个算法，由 L 和 R 建立一个一维数组 T[n]，使 T[i]存放结点 i 的父亲；然后再写一个判别结点 U 是否为结点 V 的后代的算法。【哈尔滨工业大学 1999 七 (14 分)】

类似本题的另外叙述有：

（1）假定用两个一维数组 L[1..n]和 R[1..n]作为有 n 个结点的二叉树的存储结构，L[i]和 R[i]分别指示结点 i 的左孩子和右孩子，0 表示空。写一算法，建立一维数组 T[1..n]，使 T 中第 i(i=1,2,...,n)个分量指示结点 i 的双亲，然后判别结点 u 是否为 v 的子孙的算法。【华南师范大学 2000 六(17 分)】

6．要求二叉树按二叉链表形式存储，

（1）写一个建立二叉树的算法。（2）写一个判别给定的二叉树是否是完全二叉树的算法。

完全二叉树定义为：深度为 K，具有 N 个结点的二叉树的每个结点都与深度为 K 的满二叉树中编号从 1 至 N 的结点一一对应。此题以此定义为准。【西北大学 2000 六 (12 分)】

类似本题的另外叙述有：

（1）试写一算法判断某二叉树是否是完全二叉树。【青岛海洋大学 1999 六(15 分)】

（2)编程，判断一棵二叉链表表示的二叉树是否是完全二叉树。【南京航空航天大学 2001 十(10 分)】

（3)编写算法判断一棵二叉树 BT 是否是完全二叉树。【北方交通大学 1997 八 (20 分)】

（4）假设二元树用左右链表示，试编写一算法，判别给定二元树是否为完全二元树？【哈尔滨工业大学 2000 十一 (14 分)】

（5）设二叉树以二叉链表为存储结构，试给出判断一棵二叉树是否为满二叉树的算法，用类 pascal 语言写为函数形式。【南开大学 1997 四 (16 分)】

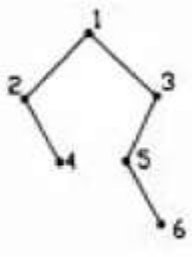
（6）试写一算法判别某二叉树是否是完全二叉树。【北京邮电大学 1994 九 (20 分)】

7．有 n 个结点的完全二叉树存放在一维数组 A[1..n]中，试据此建立一棵用二叉链表表示的二叉树 ，根由 tree 指向。 【南京理工大学 1998 七、1 （6 分)】

8．设任意非空二叉树中结点按层次顺序依次编号为 1,2,…,n(n>0),其存储结构采用下图所示形式，其中 i 表示结点的编号，L(i)的值是 i 的左儿子的编号，R(i)的值是 i 的右儿子的编号。若 L(i),R(i)的值为 0，表示结点 i 无左儿子或右儿子。试设计算法：

（1）求出二叉树的高度。

i	L(i)	R(i)	D(i)
1	2	3	
2	0	4	
3	5	0	
4	0	0	
5	0	6	
6	0	0	



（2）求出每个结点的层号（根结点层号为 1），并填入 D(i)中。(可采用任何高级语言，但要注明你所采用的语言名称)。【山东大学 1992 三、 (13 分)】

9．已知深度为 h 的二叉树采用顺序存储结构已存放于数组 BT[1:2^h-1]中，请写一非递归算法，产生该二叉树的二叉链表结构。设二叉链表中链结点的构造为(lchild,data,rchild),根结点所在链结点的指针由 T 给出。【北京航空航天大学 1999 七、(15 分)】

10. 二叉树的动态二叉链表结构中的每个结点有三个字段：data，lchild，rchild。其

中指针 lchild

和 rchild 的类型为 bitre。静态二叉链表是用数组作为存储空间，每个数组元素存储二叉树的一个结点，也有三个字段：data，lchild，rchild。所不同的是，lchild 和 rdhild 为 integer 型，分别用于存储左右孩子的下标，如果没有左右孩子，则相应的值为 0。例如，下面左图所示的二叉树的静态二叉链表如右图所示。

编写算法由二叉树的动态二叉链表构造出相应的静态二叉链表 a[1..n]，并写出其调用形式和有关的类型描述。其中 n 为一个确定的整数。【合肥工业大学 2000 五、3 （8 分）】

11．假设以双亲表示法作树的存储结构，写出双亲表示的类型说明，并编写求给定的树的深度的算法。(注:已知树中结点数)【清华大学 1994 七、 (15 分)】

12．试编写算法求出二叉树的深度。二叉树的存储结构为如下说明的二叉链表:

```
TYPE  btre=↑bnode
      bnode=RECORD data:datatype; lch,rch:btre  END;
```

【北京轻工业学院 1997 一(15 分)】【南京航空航天大学 1997 十(10)】【北京理工大学 2000 四 3(4)】

13．二叉树采用二叉链表存储：

（1）编写计算整个二叉树高度的算法（二叉树的高度也叫二叉树的深度）。

（2）编写计算二叉树最大宽度的算法（二叉树的最大宽度是指二叉树所有层中结点个数的最大值）。

【西北大学 2001 四】

14． 以孩子兄弟链表为存储结构，请设计递归和非递归算法求树的深度。【北方交通大学 1999 五(18 分)】

类似本题的另外叙述有：

（1）设 T 是一棵 n 元树，Tb 是 T 的孩子兄弟表示（二叉链表）的二叉树，试编程，由 Tb 计算 T 的高度（要求用非递归方法实现）。【南京航空航天大学 2000 九】

15．设一棵二叉树的结点结构为 (LLINK,INFO,RLINK),ROOT 为指向该二叉树根结点的指针，p 和 q 分别为指向该二叉树中任意两个结点的指针，试编写一算法 ANCESTOR（ROOT，p,q,r），该算法找到 p 和 q 的最近共同祖先结点 r。【吉林大学 2000 二、3 （12 分）】【中山大学 1994 六（15 分）】

16．已知一棵二叉树按顺序方式存储在数组 A[1..n]中。设计算法，求出下标分别为 i 和 j 的两个结点的最近的公共祖先结点的值。【武汉大学 2000 五 1】

17．设计这样的二叉树，用它可以表示父子、夫妻和兄弟三种关系，并编写一个查找任意父亲结点的所有儿子的过程。【燕山大学 2001 四、5 （8 分）】

18．在二叉树中查找值为 x 的结点，试编写算法（用 C 语言）打印值为 x 的结点的所有祖先，假设值为 x 的结点不多于一个，最后试分析该算法的时间复杂度（若不加分析，直接

写出结果，按零分算)。

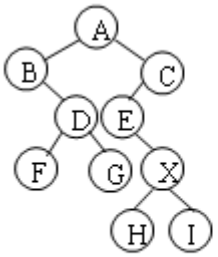
【上海交通大学 1998 五】

类似本题的另外叙述有：

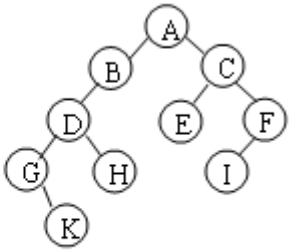
（1）在二叉树中查找值为 x 的结点，请编写一算法用以打印值为 x 的结点的所有祖先，假设值为 x 的结点不多于 1 个。注：采用非递归算法。【西安电子科技大学 1996 六(10 分)】

（2）设二叉树中结点的数据域的值互不相同，试设计一个算法将数据域值为 x 的结点的所有祖先结点的数据域打印出来。【北方交通大学 1996 八(20 分)】

（3）设二叉树根指针为 t,且树中结点值各不相同，写出算法 disp_f(t,x),查找树中值为 t 的结点，并显示出其所有祖先结点的值。【首都经贸大学 1998 三、4 （20 分)】



18(4)题图



19题图

（4）若一棵二叉树中没有数据域值相同的结点，设计算法打印数据域值为 x 的所有祖先结点的数据域。如果根结点的数据域值为 x 或不存在数据域值为 x 的结点，则什么也不打印。例如右图所示的二叉树，则打印结点序列为 A、C、E。

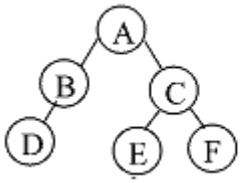
【北京工业大学 1995 五、 (16 分)】

19. 利用栈的基本操作写出先序遍历二叉树的非递归算法要求进栈的元素最少，并指出下列(最右图)二叉树中需进栈的元素。 【山东科技大学 2002 四、 (10 分)】

20. 设一棵完全二叉树使用顺序存储在数组 bt[1..n]中,请写出进行非递归的前序遍历算法。【西安电子科技大学 1998 四(8 分)】

21. 若二叉树用以下存储结构表示，试给出求前序遍历的算法：

TYPE Tree:=ARRAY[1..max] OF RECORD data:char;parent:integer; END;



下标	1	2	3	4	5	6
数据	D	C	A	F	B	E
父母	-5	3	0	2	-3	-2

【北京邮电大学 2002 五、4 （15 分)】

22. 设计算法返回二叉树 T 的先序序列的最后一个结点的指针，要求采用非递归形式，且不许用栈。

【合肥工业大学 1999 五、2 （8 分)】

23. 已知一棵高度为 K 具有 n 个结点的二叉树，按顺序方式存储：

（1）编写用先根遍历树中每个结点的递归算法；

（2）编写将树中最大序号叶子结点的祖先结点全部打印输出的算法。【东北大学 1997 六(20 分)】。

24. 对于二叉树的链接实现,完成非递归的中序遍历过程。 【中山大学 1999 五、 (15 分)】

类似本题的另外叙述有：

（1）写出中序遍历二叉树的非递归算法及递推算法。【大连海事大学 1996 六、2 （10 分)】。

（2）设计一个中序遍历算法，应用栈来存储树结点，要求结点仅能进栈和出栈一次。（本题指中序遍历二叉树)【西安电子科技大学 1999 计应用 四 (10 分)】

（3）用非递归方式写出二叉树中序遍历算法。【山东科技大学 2002 六、2 (9 分)】

25．已知二叉树用下面的顺序存储结构，写出中序遍历该二叉树的算法。

```
TYPE  ARRAY [1..maxn] OF RECORD data:char;   //存储结点值
                Lc,Rc;integer; END; //存左孩子右孩子的下标，0 表示无左、右孩子。
```

	1	2	3	4	5	6	7	8	9
data	A	B	C	D	E	F	G	H	I
Lc	2	4	0	0	0	8	0	0	0
Rc	3	5	6	0	7	9	0	0	0

如树 T=A（B（D，E,(#,G)），C（#，F（H，I））存储如上图。【北京邮电大学 1999 九 (10 分)】

26．试给出二叉树的自下而上、自右而左的层次遍历算法。【吉林大学 2001 二 、2 （8 分)】

27．在一棵以二叉链表表示的二叉树上，试写出用按层次顺序遍历二叉树的方法，统计树中具有度为 1 的结点数目的算法。二叉链表的类型定义为：

```
TYPE bitreptr=^bnodetp;
     bnodetp=RECORD data:char; lchild,rchild:bitreptr END;
```

【同济大学 2000 三、2 （12 分)】

类似本题的另外叙述有：

（1）请设计算法按层次顺序遍历二叉树。【北方交通大学 2001 四、 (20 分)】

（2)试以二叉链表作存储结构，编写按层次顺序遍历二叉树的算法。【上海交通大学 1999 三(12 分)】

（3）已知一棵以二叉链表作存储结构的二叉树，编写按层次顺序（同一层自左至右）遍历二叉树的算法。

【燕山大学 1999 十、1 （8 分)】

（4)设二叉树用二叉链表存储，试编写按层输出二叉树结点的算法。【北京理工大学 2001 九、2（8 分)】

（5）写出按层次顺序打印任意二叉树 T 中结点的程序。二叉树采用双链结构，结点形式为（LSON，DATA，RSON）可采用任何你熟识的算法语言，设 T 指向二叉树的根结点。【山东大学 1993 二 (12 分)】

28．设一棵二叉树以二叉链表为存贮结构，结点结构为(lchild, data,rchild)，设计一个算法将二叉树中所有结点的左，右子树相互交换。【福州大学 1998 四、2 (10 分)】

类似本题的另外叙述有：

（1）设 t 为一棵二叉树的根结点地址指针，试设计一个非递归的算法完成把二叉树中每个结点的左右孩子位置交换。【东北大学 1996 五、 (14 分)】

（2）写一个将二叉树中每个结点的左右孩子交换的算法。（统考生做）【南京航空航天大学 1999 九(10 分)】

29．设 T 是一棵满二叉树，编写一个将 T 的先序遍历序列转换为后序遍历序列的递归算法。

【东北大学 2001 三 (15 分)】

30．已知一棵二叉树的中序序列和后序序列，写一个建立该二叉树的二叉链表存储结构

的算法。

【东北大学 1999 六、3 （12 分)】

31．设二叉树采用二叉链表作为存储结构。试用类 PASCAL 语言实现按前序遍历顺序输出二叉树中结点的非递归算法。要求定义所用结构。设栈已经定义：inits(S)，empty（S） push（S，P)，pop（S)，top（S）分别为栈初始化，判栈空，入栈，出栈，看栈顶等操作。【北京工业大学 1997 二、1（10 分)】

32．已知深度为 h 的二叉树以一维数组 BT(1:2^h-1)作为其存储结构。请写一算法，求该二叉树中叶结点的个数。【北京航空航天大学 1996】

33．设某二叉树结点结构为:

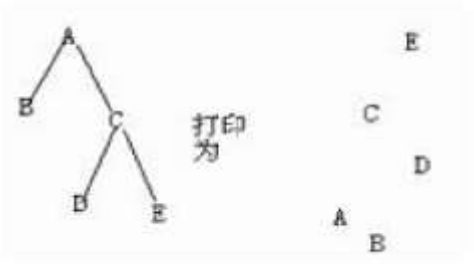
```
TYPE bitreptr=^bnodetp;
     bnodetp=RECORD data:integer; lchild,rchild:bitreptr END;
```

试编写算法，计算每层中结点 data 域数值大于 50 的结点个数，并输出这些结点的 data 域的数值和序号。

【北京工业大学 1998 九(10 分)】

34．编写递归程序将二叉树逆时针旋转 90 度打印出来。如右图：（要求用类 PASCAL 语言，并描述结构)。

【北京工业大学 1999 二 、 (6 分)】



35．二叉树排序方法如下：

（1）将第一个数据放在树根。

（2）将随后读入的数据与树根中的数据相比较，若比树根大，则置于右子树，反之则置于左子树，建成一棵二叉树；

（3）利用中序遍历打印排序结果。

试用PASCAL或C语言编写二叉树的排序程序,并分析其算法复杂性。【浙江大学 1995 九 (15 分)】

36．已知一二叉树中结点的左右孩子为 left 和 right，p 指向二叉树的某一结点。请用 C 或 PASCAL 编一个非递归函数 postfirst(p),求 p 所对应子树的第一个后序遍历结点。【浙江大学 1998 六(10 分)】

37．已知二叉树 T 的结点在先根次序下的排列为 A[1]，A[2]，…，A[n]，在中根次序下的排列为 B[1]，B[2]，…，B[n]，其中，A 和 B 是一维数组，数组元素的值为 T 中相应的结点的 INFO 字段的值，并假定二叉树 T 中结点的 INFO 字段的值互不相同，n>=0。试解答：

（1）证明由 A[1：n]和 B[1：n]能唯一的确定二叉树 T 的结构；

（2）给出建造二叉树 T 的算法，要求所建造的二叉树以 LLINK/RLINK 链接结构表示，且该算法是非递归算法；

(3) 分析你所给算法的时间复杂性，该过程包括如何确定基本运算如何推导出期望复杂性和最坏复杂性。【吉林大学 1997 四 (20 分) 1998 二】

38．已知一具有 n 个结点的二叉树的中序遍历序列与后序遍历序列分别存放于数组 IN[1：

n]和 POST[1：n]中，（设该二叉树各结点的数据值均不相同）。请写一建立该二叉树的二叉链表结构的非递归算法。该二叉链表的链结点结构为(lchild,data,rchild)，其中 data 为数据域，lchild 与 rhild 分别为指向该结点左、右孩子的指针域（当孩子结点不存在时，相应指针域为空，用 nil 表示）。

【北京航空航天大学 1998 六 (15 分)】

39．试写出复制一棵二叉树的算法。二叉树采用标准链接结构。【山东大学 2000 二 (10 分)】。

类似本题的另外叙述有：

（1）已知二叉树 T,试写出复制该二叉树的算法(t→T)

（1）(8 分)递归算法（2）(12 分)非递归算法【北方交通大学 1993 七(20 分)】

（2）算法题（共 20 分，每题 10 分）

（1）试写出一递归函数，判别两棵树是否相等。

（2）试写出一递归函数，复制一棵二叉树。【山东工业大学 1997 八、 (20 分)】

40．假设一维数组 H[1：n]存放森林 F 的每个结点的地址，且序列 H[1]，H[2]，…，H[n]正好是森林 F 在先根次序下结点地址的排列；E[1:n]是一维数组，且当 1<=i<=n 时，E[i]是 H[i]所指结点的次数（即儿子结点的个数）。试给出一个算法，该算法计算森林 F 的树形个数，并计算森林 F 的最后一个树形的根结点地址。【吉林大学 1995 五(15 分)】

41．请设计一个算法，要求该算法把二叉树的叶子结点按从左到右的顺序连成一个单链表，表头指针为 head。 二叉树按二叉链表方式存储，链接时用叶子结点的右指针域来存放单链表指针。分析你的算法的时、空复杂度。【华南师范大学 1999 六、2 （13 分)】

类似本题的另外叙述有：

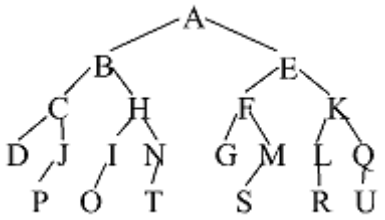
（1）已知二叉树的链表存储结构定义如下：

```
TYPE bitreptr=^bitrenode;
    bitrenode=RECORD data:char; lchild,rchild:bitreptr END;
```

编写一个递归算法，利用叶结点中空的右链指针域 rchild，将所有叶结点自左至右链接成一个单链表，算法返回最左叶结点的地址（链头）。 【清华大学 1997 三、 (10 分)】

42．设二叉树以二叉链表示。使用类 PASCAL 语言编一过程，输出二叉树中各结点的数据及其所在的层数（已知一棵二叉树按中序遍历时各结点被访问的次序和这棵二叉树按后序遍历时各结点被访问的次序，是否唯一确定这棵二叉树的结构？为什么？若已知一棵二叉树按先序遍历时各结点被访问的次序和这棵二叉树按后序遍历时各结点访问的次序，能否唯一确定这棵二叉树的结构？为什么？）

【南开大学 1997 四(15 分) 1998 三(12 分)】



43．写一非递归遍历算法，使右图树遍历输出顺序为字母顺序。

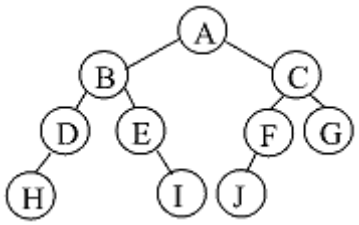
【中国人民大学 2000 三、1 （10 分)】

44．二叉树结点的平衡因子（bf）定义为该结点的左子树高度与右子树高度之差。设二叉树结点结构为：(lchild,data,bf,rchild),
lchild,rchild 是左右儿子指针；data 是数据元素；bf 是平衡因子，编写递归算法计算二叉树中各个结点的平衡因子。【石油大学 1998 四、 (18 分)】

类似本题的另外叙述有：

(1)设二叉树结点结构为(left,data,bf,right)。定义二叉树结点 T 的平衡因子 bf(T)=hl-hr,写一递归算法确定二叉树 tree 中各结点的平衡因子 bf,同时返回二叉树中非叶结点个数。【东南大学 1996 四(15 分)】

45. 已知二叉树 T 采用二叉链表结构存储，每个结点有三个字段：data,Lchild 和 Rchild 。设计算法求出 T 的顺序存储结构 A[1..n]，并给出初始调用形式。要求：如某位置为空，将其置为 null；如超出下标范围 n，则报错；最后返回实际的最大下标.右图所示为 n=15 时一个二叉树及所对应的输出结果示例（空缺表示 null)。输出结果（表结构的值和最大下标)：=12（最大下标为 12)【合肥工业大学 2001 五、5 （8 分)】



1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	B	C	D	E	F	G	H			I	J			

46. 设两棵二叉树的的根结点地址分别为 p 和 q，采用二叉链表的形式存储这两棵树上所有的结点。请编写程序，判断它们是否相似。【上海交通大学 2000 十二(8 分)】

类似本题的另外叙述有：

（1）编写一个函数或过程判定两棵二叉树是否相似，所谓两棵二叉树 s 和 t 相似，即是要么它们都为空或都只有一个结点，要么它们的左右子树都相似。【厦门大学 2000 四、1 （9 分)】

（2）设计判断两棵二叉树是否相似的算法。【中国矿业大学 2000 四(10 分)】

47. 编写递归算法，依据树的双亲表示法及其根结点创建树的孩子－兄弟链表存储结构。要求写算法以前先写出这两种存储结构的类型说明。【清华大学 1995 六(20 分)】

48. 已知二叉树以二叉链表存储，编写算法完成：对于树中每一个元素值为 x 的结点，删去以它为根的子树，并释放相应的空间。【北京轻工业学院 1998 二(14 分)】

类似本题的另外叙述有：

（1）设 T 是一棵给定的查找树,试编写一个在树中删除根结点为 a 的子树的程序,要求在删除的过程中释放该子树所有结点所占有的存储空间,这里假设树 T 中结点所占有的存储空间是通过动态存储分配取得的,其结点的形式为:(lchild,data,rchild)【复旦大学 1999 七、(15 分)】

49. 试为二叉树写出一个建立三叉链表的算法，并在此三叉链表中删去每一个元素值为 x 的结点，以及以它为根的子树，且释放相应存储空间。二叉树的三叉链表的描述为：

```
TYPE bitreptr=^nodetp;
       nodetp=RECORD data:char; lchild,rchild,parent:bitreptr END;
VAR  bt:bitreptr;
```

【同济大学 1998 四(14 分)】

50. 设一棵二叉树的根结点指针为 T，C 为计数变量，初值为 0，试写出对此二叉树中结点计数的算法：BTLC（T，C）。【北京科技大学 1999 十、2（10 分) 2000 十、2（10)】

51.试编写算法,对一棵以孩子—兄弟链表表示的树统计叶子的个数。【北京轻工业学院 2000 四(15 分)】

52.设计算法:统计一棵二叉树中所有叶结点的数目及非叶结点的数目。【南开大学 2000 三、1】

53．用类 PASCAL 语言编写一非递归算法，求二叉树上叶子结点的数量。二叉树用二叉链表存贮，左指针定义为 lchild,右指针定义为 rchild。【燕山大学 2000 七、2（8 分）】

类似本题的另外叙述有：

（1）用递归方法求已知二叉树的叶结点个数。【天津大学 1999 七】

54．一棵二叉树以二叉链表来表示，求其指定的某一层 k(k>1)上的叶子结点的个数。

【上海大学 1999 三、1(18 分)】

55．二叉树采用二叉链表方式存放，对二叉树从 1 开始进行连续编号，要求每个结点的编号大于其左、右孩子的编号，同一个结点的左右孩子中，其左孩子的编号小于其右孩子的编号，请回答采用什么次序的遍历方式实现编号？并给出在二叉树中结点的数据域部分填写实现如上要求编号的非递归算法。

【西北大学 2001 六】

56．设一棵二叉树中各结点的值互不相同，其前序序列和中序序列分别存于两个一维数组 pre[1..n]和 mid[1..n]中，试遍写算法建立该二叉树的二叉链表。【南京航空航天大学 1999 十(10 分)】

类似本题的另外叙述有：

（1）已知一棵二叉树的先序遍历序列和中序遍历序列分别存于两个一维数组中，试编写算法建立该二叉树的二叉链表。【上海交通大学 1999 四(12 分)】

（2）已知一棵二叉树的前序序列和中序序列分别存于两个一维数组 PRE[1..n]和 INO[1..n]中，请编写算法来建立该二叉树的二叉链表。【西安电子科技大学 1999 软件 三(8 分)】

（3）已知一棵二叉树的前序序列和中序序列，可唯一地确定该二叉树。试编写据此思想构造二叉树的算法。【北方交通大学 1995 七(20 分)】

57．已知二叉树的中序遍历序列为 GFBEANHM，后序遍历的结点序列为 GEBFHNMA。

（1）画出此二叉树的形态。（2）写出根据二叉树的中序和后序遍历的结点序列，建立它的二叉链表存储结构的递归算法。【北京邮电大学 1992 四 (20 分)】

58．假设二叉树采用链式存储结构进行存储，root^为根结点，p^为任一给定的结点，请写出求从根结点到 p^之间路径的非递归算法。【西安电子科技大学 2000 软件 三(9 分)】

59．设二叉树的结点具有如下的结构：(lchild,info,rchild)，指针变量 BT 指向该树的根结点，试设计一个算法打印出由根结点出发到达叶结点的所有路径。

【北方交通大学 1994 八(16 分)】【中国人民大学 2000 三、2（10 分）】

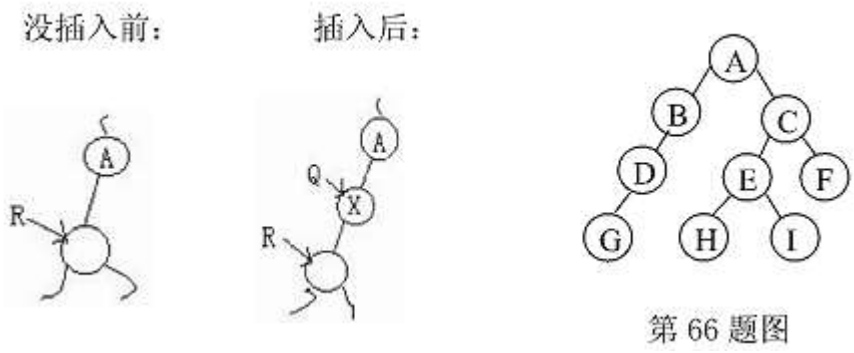
60．设二叉树的结点结构是（Lc,data,Rc），其中 Lc、Rc 分别为指向左、右子树根的指针，data 是字符型数据。试写出算法，求任意二叉树中第一条最长的路径长度，并输出此路径上各结点的值。

【北京邮电大学 1997 八(20 分)】

61.设 t 是一棵按后序遍历方式构成的线索二叉树的根结点指针，试设计一个非递归的算法，把一个地址为 x 的新结点插到 t 树中，已知地址为 y 的结点右侧作为结点 y 的右孩子，并使插入后的二叉树仍为后序线索二叉树。【东北大学 1996 七 (15 分)】

62．请用类 C 或用类 PASCAL 语言编写算法。请编写在中序全线索二叉树 T 中的结点 P 下插入一棵根为 X 的中序全线索二叉树的算法。如果 P 左右孩子都存在，则插入失败并返回 FALSE；如果 P 没有左孩子，则 X 作为 P 的左孩子插入；否则 X 作为 P 的右孩子插入。插入完成后要求二叉树保持中序全线索并返回 TRUE。【上海大学 2002 七、1 （10 分）】

63．有中序穿索树 T，结点形式为：(LL,LT,D,RT,RL)，试编写非递归算法找到数据域为 A 的结点，并在其左子树中插入已知新结点 X：插入方式如下：



第 66 题图

注意：可能 A 有左孩子或无左孩子，插入后考虑穿索的状态应作何修改。【上海大学 1998 六(17 分)】

64. 编写一算法，利用叶子结点中的空指针域将所有叶子结点链接为一个带有头结点的双链表，算法返回头结点的地址。【东北大学 1999 四(13 分)】

65. 编写程序段，利用中序全线索树求其中任意结点 p^的前序后继结点，结果仍用 p 指出。要求先描述结构和算法思路。设线索树不带头结点，其中序序列第一结点的左标志和最后结点的右标志皆为 0 （非线索），对应指针皆为空。【北京工业大学 2000 七(10 分)】

66. 已知一个二叉树如下图，修改结点（node）的连接方式，以致可以不借助辅助堆栈实现中序遍历的非递归方法。画出修改后的结点连接图并写出其实现中序遍历的非递归算法。【浙江大学 2002 五（10 分）】

67. 已知指针 p 指向带表头的中根次序穿线二叉树中的某结点，试写一算法 FFA（p，q），该算法寻找结点 p 的父亲结点 q。设穿线二叉树的结点结构、表头结点结构和空树结构分别为（LTAG,LLINK,INFO,RLINK,RTAG）,且规定穿线树的最左下结点的 LLINK 域和最右下结点的 RLINK 域指向表头。【吉林大学 1999 二 、1 （16 分)】

68. 给出中序线索树的结点结构并画出一个具有头结点的中序线索树，使其树结点至少应有 6 个。写一算法在不使用栈和递归的情况下前序遍历一中序线索树，并分析其时间复杂性。

【东南大学 1993 三(20 分) 1997 三(18 分) 1998 六(14 分)】

69. 设有二叉树 BT，每个结点包括 ltag、lchild、data、rchild、rtag 五个字段，依次为左标志、左儿子、数据、右儿子、右标志。给出将二叉树 BT 建成前序（即先序）线索二叉树的递归算法。

【四川联合大学 2000 三】【东南大学 1999 六(15 分)】

70. 写出中序线索二叉树的线索化过程（已知二叉树 T)。

【山东大学 2000 五、2 (10 分)】【长沙铁道学院 1997 五、6 (10 分)】

71. 已知一中序线索二叉树,写一算法完成对它的中序扫描。【山东大学 2001 软件与理论三(8 分)】

72. 已知中序线索二叉树 T 右子树不空。设计算法，将 S 所指的结点作为 T 的右子树中的一个叶子结点插入进去，并使之成为 T 的右子树的（中序序列）第一个结点（同时要修改相应的线索关系)。

【合肥工业大学 2001 五、2（8 分)】

73. 写出算法，求出中序线索二叉树中给定值为 x 的结点之后继结点，返回该后继结点的指针。线索树中结点结构为：(ltag,lc,data,rc,rtag)。其中，data 存放结点的值；lc，rc 为指向左、右孩子或该结点前驱或后继的指针；ltag，rtag 为标志域，各值为：0，则 lc，rc 为指向左、右孩子的指针；值为 1，则 lc，rc 为指向某前驱后继结点的指针。【北京邮电大学 1996 八(20 分)】

74. 设后序线索树中结点构造为(Ltag,Lchild,Data,Rchild,Rtag)。其中:Ltag,Rtag 值为

0 时,Lchild、Rchild 分别为儿子指针；否则分别为直接前驱,直接后继的线索。请写出在后序线索树上找给定结点 p^ 的直接前驱 q 的算法。【武汉交通科技大学 1996 四、1（13 分)】

75. 用算法说明在对称序穿线树中，如何对任意给定的结点直接找出该结点的对称序后继。【山东大学 1999 六、 3 (10 分)】

76. 写出在中序线索二叉树里；找指定结点在后序下的前驱结点的算法。【河海大学 1998 七(10 分)】

77. 设中序穿线二叉树的结点由五个域构成：info:给出结点的数据场之值。LL：当 LT 为 1 时，则给出该结点的左儿子之地址，当 LT 为 0 时，则给出按中序遍历的前驱结点的地址。LT:标志域，为 1 或为 0。RL：当 RT 为 1 时，则给出该结点的右儿子的地址；当 RT 为 0 时，则给出按中序遍历的后继结点地址。RT: 标志域为 0 或为 1。

请编写程序，在具有上述结点结构的中序穿线二叉树上，求某一结点 p 的按后序遍历次序的后继结点的地址 q，设该中序穿线二叉树的根结点地址为 r。另外，请注意必须满足：（1）额外空间的使用只能为 0(1)，(2)程序为非递归。【上海交通大学 2000 十(20 分)】

78. 写出按后序序列遍历中序线索树的算法。【东南大学 2000 六(15 分)】

79. 给定一组项及其权值，假定项都存放于二叉树的树叶结点，则具有最小带权外部路径长度的树称为 huffman 树。（1）给出构造 huffman 树 的算法。（2）给定项及相应的权如下表:画出执行上述算法后得到的 huffman 树。(3)用 c 语言编写构造 huffman 树的程序.【浙江大学 2000 七 (18 分)】

序号	1	2	3	4	5	6	7	8	9
向	A	B	C	D	E	F	G	H	I
	15	6	7	12	25	4	6	1	15

80、二叉树 T 的中序遍历序列和层次遍历序列分别是 BAFDGCE 和 ABCDEFG，试画出该二叉树（3 分)，并写出由二叉树的中序遍历序列和层次遍历序列确定二叉树的算法（5 分)。【烟台大学 2004 四、1(8 分)】

第 6 章　树和二叉树（答案）

一、选择题

1.D	2.B	3.C	4.D	5.D	6.A	7.1C	7.2A	7.3C	7.4A	7.5C	8.B
9.C	10.D	11.B	12.E	13.D	14.D	15.C	16.B	17.C	18.C	19.B	20.D
21.A	22.A	23.C	24.C	25.C	26.C	27.C	28.C	29.B	30.C	31.D	32.B
33.A	34.D	35.B	36.B	37.C	38.B	39.B	40.B	41.1F	41.2B	42.C	43.B
44.C	45.C	46.B	47.D	48.B	49.C	50.A	51.C	52.C	53.C	54.D	55.C
56.B	57.A	58.D	59.D	60.B	61.1B	61.2A	61.3G	62.B	63.B	64.D	65.D
66.1C	66.2D	66.3F	66.4H	66.5I							

部分答案解释如下。

12. 由二叉树结点的公式：n=n0+n1+n2=n0+n1+(n0-1)=2n0+n1-1， 因为 n=1001,所以 1002=2n0+n1,在完全二叉树树中，n1 只能取 0 或 1,在本题中只能取 0，故 n=501，因此选 E。

42.前序序列是“根左右”，后序序列是“左右根”，若要这两个序列相反，只有单支树，所以本题的 A 和 B 均对，单支树的特点是只有一个叶子结点，故 C 是最合适的，选 C。A 或 B

都不全。由本题可解答 44 题。

47. 左子树为空的二叉树的根结点的左线索为空（无前驱），先序序列的最后结点的右线索为空（无后继），共 2 个空链域。

52. 线索二叉树是利用二叉树的空链域加上线索，n 个结点的二叉树有 n+1 个空链域。

二、判断题

1.×	2.×	3.×	4. √	5. √	6. √	7.√	8.×	9. √	10.×	11.×	12.×
13.×	14.√	15.×	16.×	17.√	18.√	19.×	20.√	21.×	22.√	23.×	24.×
25.√	26.×	27.×	28.×	29.√	30.×	31.×	32.√	33.×	34.×	35.×	36.√
37.√	38.×	39.×	40.×	41.(3)	42.√	43.√	44.×	45.√	46.×	47.×	48.×
49.√	50.√										

部分答案解释如下。

6. 只有在确定何序（前序、中序、后序或层次）遍历后，遍历结果才唯一。

19. 任何结点至多只有左子树的二叉树的遍历就不需要栈。

24. 只对完全二叉树适用，编号为 i 的结点的左儿子的编号为 2i(2i<=n)，右儿子是 2i+1（2i+1<=n）

37. 其中序前驱是其左子树上按中序遍历的最右边的结点（叶子或无右子女），该结点无右孩子。

38 . 新插入的结点都是叶子结点。

42. 在二叉树上，对有左右子女的结点，其中序前驱是其左子树上按中序遍历的最右边的结点（该结点的后继指针指向祖先），中序后继是其右子树上按中序遍历的最左边的结点（该结点的前驱指针指向祖先）。

44. 非空二叉树中序遍历第一个结点无前驱，最后一个结点无后继，这两个结点的前驱线索和后继线索为空指针。

三. 填空题

1.(1)根结点(2)左子树(3)右子树　　2.(1)双亲链表表示法(2)孩子链表表示法(3)孩子兄弟表示法

3. p->lchild==null && p->rchlid==null　　4.(1) ++a*b3*4-cd　(2)18　　5.平衡因子

6. 9　　7. 12　　8.(1)2^{k-1}　(2)2^k-1　　9.(1)2^{H-1}　(2)2^H-1　(3)H=⌊$\log_2$N⌋+1

10. 用顺序存储二叉树时，要按完全二叉树的形式存储，非完全二叉树存储时，要加“虚结点”。设编号为 i 和 j 的结点在顺序存储中的下标为 s 和 t ，则结点 i 和 j 在同一层上的条件是⌊$\log_2$s⌋=⌊$\log_2$t⌋。

11. ⌊$\log_2$i⌋=⌊$\log_2$j⌋　　12.(1)0　(2)(n-1)/2　(3)(n+1)/2　(4) ⌊$\log_2$n⌋ +1

13.n

14. N2+1　　15.(1) 2^{K+1}-1 (2) k+1　　16.⌊N/2⌋　　17. 2^{k-2}　　18. 64

19. 99　　20. 11　　21.(1) n1-1 (2)n2+n3

22.(1)2^{k-2}+1（第 k 层 1 个结点，总结点个数是 2^{H-1}，其双亲是 $2^{H-1}/2=2^{k-2}$）(2) ⌊$\log_2$i⌋+1

23.69

24. 4　　25.3^{h-1}　　26. ⌊n/2⌋　　27.⌈$\log_2$k⌉+1

28.(1)完全二叉树 (2)单枝树，树中任一结点（除最后一个结点是叶子外），只有左子女或只有右子女。

29. N+1　30. (1) 128(第七层满，加第八层 1 个)　(2) 7
31. 0 至多个。任意二叉树，度为 1 的结点个数没限制。只有完全二叉树，度为 1 的结点个数才至多为 1。
32. 21　33. (1)2　(2) n-1　(3) 1　(4) n　(5) 1　(6) n-1
34. (1) FEGHDCB　(2)BEF（该二叉树转换成森林，含三棵树，其第一棵树的先根次序是 BEF）
35. (1)先序（2）中序　36. (1)EACBDGF （2）2　37. 任何结点至多只有右子女的二叉树。
38. (1)a　(2) dbe　(3) hfcg　39. (1) .D.G.B.A.E.H.C.F.　(2) ...GD.B...HE..FCA
40. DGEBFCA　41. (1)5　(2)略　42. 二叉排序树　43. 二叉树　44. 前序
45. (1) 先根次序（2）中根次序　46. 双亲的右子树中最左下的叶子结点　47. 2
48. (n+1)/2
49. 31（x 的后继是经 x 的双亲 y 的右子树中最左下的叶结点）　50. (1)前驱　(2)后继
51. (1)1　(2)y^.lchild　(3)0　(4)x　(5)1　(6) y　(7)x(编者注：本题按中序线索化)
52. 带权路径长度最小的二叉树，又称最优二叉树　53. 69　54. (1)6 (2)261
55. (1)80 (2)001（不唯一）56. $2n_0-1$
57. 本题①是表达式求值，②是在二叉排序树中删除值为 x 的结点。首先查找 x，若没有 x，则结束。否则分成四种情况讨论：x 结点有左右子树；只有左子树；只有右子树和本身是叶子。
(1)Postoder_eval(t^.Lchild)　(2) Postorder_eval(t^.Rchild)　(3)ERROR(无此运算符)(4)A
(5)tempA^.Lchild　(6)tempA=NULL　(7)q^.Rchild　(8)q　(9)tempA^.Rchild
(10)tempA^.Item<r^.Item
58. (1) **IF** t=NIL THEN num:=0 **ELSE** num:=num(t^.l)+num(t^.r)+1
(2) **IF** (t=NIL) AND (m≤n) OR (t<>NIL) AND (m>n) THEN all:=false
ELSE BEGIN chk(t^.l,2*m);chk (t^.r,2*m+1);END
59. (1)p->rchild　(2)p->lchild　(3)p->lchild　(4)ADDQ(Q,p->lchild)
(5)ADDQ(Q,p->rchild)
60. (1)t->rchild!=null　(2)t->rchild!=null　(3)NO++　(4)count(t->lchild)
(5)count(t->rchild)
61. (1)p　(2)0　(3)height(p->lchild)　(4)0　(5)height(p->rchild)　(6)lh+1
(7)rh+1　(8)0
62. (1)p<>NIL　(2)addx(p)　(3)addx(tree)　(4)r^.rchild
63. (1)stack[tp]=t　(2) p=stack[tp--]　(3)p
(4)++tp
64. ① 本算法将二叉树的左右子树交换
② (1)new (s)　//初始化，申请结点　(2) s^.next=NIL　//s 是带头结点的链栈
(3)s^.next^.data　//取栈顶元素　(4)s^.next:= p^.next　//栈顶指针下移
(5)dispose(p)　//回收空间　(6)p^.next:=s^.next　//将新结点入链栈

(7)push(s,p^.rchild) //先沿树的左分支向下，将 p 的右子女入栈保存

(8)NOT empty(s)　(9) finishe:=true //已完成　(10)finish=true （或 s^.next=NIL）

65.(1)new(t)　(2)2*i≤n　(3)t^.lchild,2*i　(4)2*i+1≤n　(5)t^.rchild,2*i+1 (6)1

66.(1)Push(s,p)　(2)K=2　(3)p->data=ch　(4)BT=p　(5) ins>>ch

67.(1)result;　(2)p:=p^.link;　(3) q:=q^.pre （（２）（３）顺序可变）

68.(1)top++　(2) stack[top]=p->rchild　(3)top++ (4)stack[top]=p->lchild

69.(1)(i<=j) AND (x<=y)　(2)A[i]<>B[k]　(3)k-x

(4)creatBT(i+1,i+L,x,k-1,s^.lchild)　(5) creatBT(i+L+1,j,k+1,y,s^.rchild)

70. (1)push(s,bt)　（２）pop(s)　（３）push(s,p^.rchild) // p 的右子树进栈

71. (1) p=p->lchild // 沿左子树向下　（２）p=p->rchild

72. (1)0　（２）hl>hr　(3)hr=hl

73. (1)top>0　(2)t*2 // 沿左分枝向下　（３）top-1 // 退栈

74. (1)p:=p^.lchild　（２）（３）p:=S.data[s.top]^.rchild　(4)s.top=0

75. (1)*ppos // 根结点　（２）rpos=ipos　(3)rpos－ipos　(4)ipos　(5)ppos+1

76. (1)top>0　(2)stack[top]:=nd^.right　(3)nd^.left<>NIL　(4)top:=top+1 (左子树非空)

77. (1) p<>thr // 未循环结束　（２）p->ltag=0　(3)p->lchild

(4)p->rtag=1 && p->rchild!=thr　(5) p=p->rchild　(6)p=p->rchild

78. 若 p^.rtag=1,则 p^.rchild 为后继，否则 p 的后继是 p 的右子树中最左下的结点

(1)q=p^.rchild　(2)q^.ltag=0　(3) q^.lchild

79.（１）tree->lchild　(2)null　(3)pre->rchild

(4)pre->rtag=1　(5) pre->right=tree;　(6) tree->right (注（４）和（５）顺序可换)

80.（１）node->rflag==0　(2)*x=bt　(3) *x=node->right

四．应用题

1．树的孩子兄弟链表表示法和二叉树二叉链表表示法，本质是一样的，只是解释不同，也就是说树（树是森林的特例，即森林中只有一棵树的特殊情况）可用二叉树唯一表示，并可使用二叉树的一些算法去解决树和森林中的问题。

树和二叉树的区别有三：一是二叉树的度至多为 2，树无此限制；二是二叉树有左右子树之分，即使在只有一个分枝的情况下， 也必须指出是左子树还是右子树，树无此限制；三是二叉树允许为空，树一般不允许为空（个别书上允许为空）。

2.树和二叉树逻辑上都是树形结构，区别有以上题 1 所述三点。二叉树不是树的特例。

3．线性表属于约束最强的线性结构，在非空线性表中，只有一个“第一个”元素，也只有一个“最后一个”元素；除第一个元素外，每个元素有唯一前驱；除最后一个元素外，每个元素有唯一后继。树是一种层次结构，有且只有一个根结点，每个结点可以有多个子女，但只有一个双亲（根无双亲），从这个意义上说存在一（双亲）对多（子女）的关系。广义表中的元素既可以是原子，也可以是子表，子表可以为它表共享。从表中套表意义上说，广义表也是层次结构。从逻辑上讲，树和广义表均属非线性结构。但在以下意义上，又蜕变为线性结构。如度为 1 的树，以及广义表中的元素都是原
子时。另外，广义表从元素之间的关系可看成前驱和后继，也符
合线性表，但这时元素有原子，也有子表，即元素并不属于同一

数据对象。

4．方法有二。一是对该算术表达式（二叉树）进行后序遍历，得到表达式的后序遍历序列，再按后缀表达式求值；二是递归求出左子树表达式的值，再递归求出右子树表达式的值，最后按根结点运算符（+、-、*、/ 等）进行最后求值。

5．该算术表达式转化的二叉树如右图所示。 第 5 题图

6．n（n>0）个结点的 d 度树共有 nd 个链域，除根结点外，每个结点均有一个指针所指，故该树的空链域有 nd-(n-1)=n(d-1)+1 个。

7.证明：设二叉树度为 0 和 2 的结点数及总的结点数分别为 n0,n2 和 n,则 n=n0+n2 …(1)

再设二叉树的分支数为 B, 除根结点外，每个结点都有一个分支所指，则 n=B+1… … …(2)

度为零的结点是叶子，没有分支，而度为 2 的结点有两个分支，因此（2）式可写为

n=2*n2+1 ……(3)

由（1）、（3）得 n2=n0-1,代入（1），并由（1）和（2）得 B=2*(n0-1)。 证毕。

8．(1)k^{h-1}(h 为层数)

（2）因为该树每层上均有 K^{h-1} 个结点，从根开始编号为 1，则结点 i 的从右向左数第 2 个孩子的结点编号为 ki。设 n 为结点 i 的子女，则关系式(i-1)k+2<=n<=ik+1 成立，因 i 是整数，故结点 n 的双亲 i 的编号为⌊n-2)/k⌋+1。

(3) 结点 n(n>1)的前一结点编号为 n-1（其最右边子女编号是(n-1)*k+1），故结点 n 的第 i 个孩子的编号是(n-1)*k+1+i。

(4) 根据以上分析，结点 n 有右兄弟的条件是，它不是双亲的从右数的第一子女，即(n-1)%k!=0，其右兄弟编号是 n+1。

9．最低高度二叉树的特点是，除最下层结点个数不满外，其余各层的结点数都应达到各层的最大值。设 n 个结点的二叉树的最低高度是 h，则 n 应满足 2^{h-1}<n<=2^h-1 关系式。解此不等式，并考虑 h 是整数，则有 h=⌊logn⌋+1，即任一结点个数为 n 的二叉树的高度至少为 0(logn)。

10．2^n-1(本题等价于高度为 n 的满二叉树有多少叶子结点，从根结点到各叶子结点的单枝树是不同的二叉树。)

11．235。由于本题求二叉树的结点数最多是多少，第 7 层共有 2^{7-1}=64 个结点，已知有 10 个叶子，其余 54 个结点均为分支结点。它在第八层上有 108 个叶子结点。所以该二叉树的结点数最多可达(2^7-1+108)=235。(注意；本题并未明说完全二叉树的高度，但根据题意，只能 8 层。)

12．1023（=2^{10}-1）

13.证明：设度为 1 和 2 的结点数是 n1 和 n2,则二叉树结点数 n 为 n=m+n1+n2…… (1)

由于二叉树根结点没有分枝所指，度为 1 和 2 的结点各有 1 个和 2 个分枝，度为 0 的结点没有分枝，故二叉树的结点数 n 与分枝数 B 有如下关系

n=B+1=n1+2*n2+1…….(2)

由（1）和（2），得 n2=m-1。即 n 个结点的二叉树，若叶子结点数是 m，则非叶子结点中有（m-1）个度为 2，其余度为 1。

14．根据顺序存储的完全二叉树的性质，编号为 i 的结点的双亲的编号是⌊i/2⌋，故 A[i]和 A[j]的最近公共祖先可如下求出：

```
while(i/2!=j/2)
    if(i>j) i=i/2; else  j=j/2;
```

退出 **while** 后，若 i/2=0,则最近公共祖先为根结点，否则最近公共祖先是 i/2。

15. N 个结点的 K 叉树，最大高度 N（只有一个叶结点的任意 k 叉树）。设最小高度为 H，第 i(1<=i<=H)层的结点数 K^{i-1}，则 $N=1+k+k^2+\cdots+k^{H-1}$，由此得 $H=\lfloor \log_K(N(K-1)+1) \rfloor$

16. 结点个数在 20 到 40 的满二叉树且结点数是素数的数是 31，其叶子数是 16。

17. 设分枝结点和叶子结点数分别是为 n_k 和 n_0，因此有 $n=n_0+n_k$ (1)

另外从树的分枝数 B 与结点的关系有 $n=B+1=K*n_k+1$ (2)

由（1）和（2）有 $n_0=n-n_k=(n(K-1)+1)/K$

18. 用顺序存储结构存储 n 个结点的完全二叉树。编号为 i 的结点，其双亲编号是$\lfloor i/2 \rfloor$(i=1 时无双亲)，其左子女是 2i(若 2i<=n,否则 i 无左子女)，右子女是 2i+1(若 2i+1<=n,否则无右子女)。

19. 根据完全二叉树的性质，最后一个结点（编号为 n）的双亲结点的编号是$\lfloor n/2 \rfloor$，这是最后一个分枝结点，在它之后是第一个终端（叶子）结点，故序号最小的叶子结点的下标是$\lfloor n/2 \rfloor+1$。

20. 按前序遍历对顶点编号，即根结点从 1 开始，对前序遍历序列的结点从小到大编号。

21. 设树的结点数为 n，分枝数为 B，则下面二式成立

$n=n_0+n_1+n_2+\cdots+n_m$ (1)

$n=B+1=n_1+2n_2+\cdots+mn_m$ (2)

由(1)和(2)得叶子结点数 $n_0=1+$

22. $\lceil \log_2 n \rceil+1$ 23. 15

24. 该结论不成立。对于任一 a∈A,可在 B 中找到最近祖先 f。a 在 f 的左子树上。对于从 f 到根结点路径上所有 b∈B，有可能 f 在 b 的右子树上，因而 a 也就在 b 的右子树上，这时 a＞b，因此 a＜b 不成立。同理可以证明 b<c 不成立。而对于任何 a∈A,c∈C 均有 a<c。

25. n 个结点的 m 次树，共有 n*m 个指针。除根结点外，其余 n-1 个结点均有指针所指，故空指针数为 n*m-(n-1)=n*(m-1)+1。证毕。

26. 证明 设度为 1 和 2 及叶子结点数分别为 n_0，n_1 和 n_2，则二叉树结点数 n 为 $n=n_0+n_1+n_2$ (1)

再看二叉树的分支数，除根结点外，其余结点都有一个分支进入，设 B 为分支总数，则 n=B+1。度为 1 和 2 的结点各有 1 个和 2 个分支，度为 0 的结点没有分支，故 $n=n_1+2n_2+1$ (2)

由（1）和（2），得 $n_0=n_2+1$。

27. 参见题 26.

28. 设完全二叉树中叶子结点数为 n，则根据完全二叉树的性质，度为 2 的结点数是 n-1，而完全二叉树中，度为 1 的结点数至多为 1，所以具有 n 个叶子结点的完全二叉树结点数是 n+(n-1)+1=2n 或 2n-1（有或无度为 1 的结点）。由于具有 2n(或 2n-1)个结点的完全二叉树的深度是$\lfloor \log_2(2n) \rfloor+1$($\lfloor \log_2(2n-1) \rfloor+1$，即$\lceil \log_2 n \rceil+1$,故 n 个叶结点的非满的完全二叉树的高度是$\lceil \log_2 n \rceil+1$。（最下层结点数>=2）。

29. (1)根据二叉树度为 2 结点个数等于叶子结点个数减 1 的性质，故具有 n 个叶子结点且非叶子结点均有左左子树的二叉树的结点数是 2n-1。

(2)证明：当 i=1 时，$2^{-(1-1)}=2^0=1$，公式成立。设当 i=n-1 时公式成立，证明当 i=n 时公式仍成立。

设某叶子结点的层号为 t，当将该结点变为内部结点，从而再增加两个叶子结点时，这

两个叶子结点的层号都是 t+1，对于公式的变化，是减少了一个原来的叶子结点，增加了两个新叶子结点，反映到公式中，因为 $2^{-(t-1)}=2^{-(t+1-1)}+2^{-(t+1-1)}$，所以结果不变，这就证明当 i=n 时公式仍成立。证毕.

30. 结点数的最大值 2^h-1（满二叉树），最小值 2h-1（第一层根结点，其余每层均两个结点）。

31.（1）k(u-1)+1+i　　(2) $\lfloor(v-2)/k\rfloor+1$　(参见第 8 题推导)

32. 该二叉树是按前序遍历顺序编号，以根结点为编号 1，前序遍历的顺序是“根左右”。

33.（1）设 n=1，则 e=0+2*1=2（只有一个根结点时，有两个外部结点），公式成立。

设有 n 个结点时，公式成立，即

$$E_n=I_n+2n \qquad (1)$$

现在要证明，当有 n+1 个结点时公式成立。

增加一个内部结点，设路径长度为 l，则

$$I_{n+1}=I_n+l \qquad (2)$$

该内部结点，其实是从一个外部结点变来的，即这时相当于也增加了一个外部结点（原外部结点变成内部结点时，增加两个外部结点），则

$$E_{n+1}=E_n+l+2 \qquad (3)$$

由（1）和（2），则（3）推导为

$$E_{n+1}=I_n+2n+l+2=I_{n+1}-l+2n+l+2$$
$$=I_{n+1}+2(n+1)$$

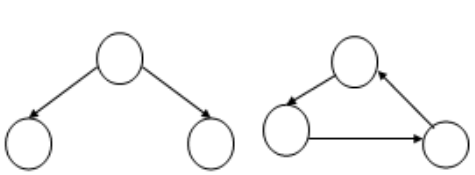
故命题成立

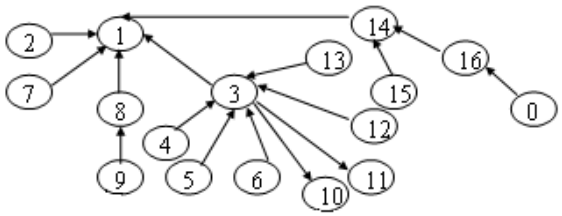
（2）成功查找的平均比较次数 s=I/n

不成功查找的平均比较次数 u=（E-n）/（n+1）=（I+n）/n+1

由以上二式，有 s=（1+1/n）*u-1 。

34.





35. 题图

该有向图只有一个顶点入度为 0，其余顶点入度均为 1，

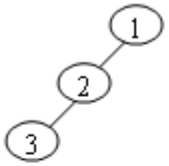
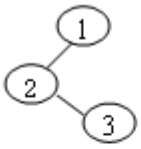
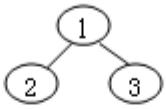
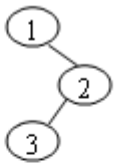
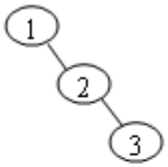
它不是有向树。

35. 题图

36. 参见 26 题

37. 由于二叉树前序遍历序列和中序遍历序列可唯一确定一棵二叉树，因此，若入栈序列为 1，2，3，…，n，相当于前序遍历序列是 1，2，3，…，n，出栈序列就是该前序遍历对应的二叉树的中序序列的数目。因为中序遍历的实质就是一个结点进栈和出栈的过程，二叉树的形态确定了结点进栈和出栈的顺序，也就确定了结点的中序序列。

下图以入栈序列 1，2，3，（解释为二叉树的前序序列）为例，说明不同形态的二叉树在中序遍历时栈的状态和访问结点次序的关系：

				
栈状态　访问 空 1 1 2 1 2 3 1 2　3 1　2 空　1	栈状态　访问 空 1 1 2 1　2 1 3 1　3 空　1	栈状态　访问 空 1 1 2 1　2 空　1 3 空　3	栈状态　访问 空 1 空　1 2 2 3 2　3 空　2	栈状态　访问 空 1 空　1 2 空　2 3 空　3

38．给定二叉树结点的前序序列和对称序（中序）序列，可以唯一确定该二叉树。因为前序序列的第一个元素是根结点，该元素将二叉树中序序列分成两部分，左边（设 l 个元素）表示左子树，若左边无元素，则说明左子树为空；右边（设 r 个元素）是右子树，若为空，则右子树为空。根据前序遍历中“根—左子树—右子树”的顺序，则由从第二元素开始的 l 个结点序列和中序序列根左边的 l 个结点序列构造左子树，由前序序列最后 r 个元素序列与中序序列根右边的 r 个元素序列构造右子树。

由二叉树的前序序列和后序序列不能唯一确定一棵二叉树，因无法确定左右子树两部分。例如，任何结点只有左子树的二叉树和任何结点只有右子树的二叉树，其前序序列相同，后序序列相同，但却是两棵不同的二叉树。

39．前序遍历是“根左右”，中序遍历是“左根右”，后序遍历是“左右根”。三种遍历中只是访问“根”结点的时机不同，对左右子树均是按左右顺序来遍历的，因此所有叶子都按相同的相对位置出现。

40．在第 38 题，已经说明由二叉树的前序序列和中序序列可以确定一棵二叉树，现在来证明由二叉树的中序序列和后序序列，也可以唯一确定一棵二叉树。

当 n=1 时，只有一个根结点，由中序序列和后序序列可以确定这棵二叉树。

设当 n=m-1 时结论成立，现证明当 n=m 时结论成立。

设中序序列为 S1，S2，…，Sm,后序序列是 P1,P2,…，Pm。因后序序列最后一个元素 Pm 是根，则在中序序列中可找到与 Pm 相等的结点（设二叉树中各结点互不相同）Si(1≤i≤m)，因中序序列是由中序遍历而得，所以 Si 是根结点，S1，S2，…，Si-1 是左子树的中序序列，而 Si+1,Si+2,…,Sm 是右子树的中序序列。

若 i=1，则 S1 是根，这时二叉树的左子树为空，右子树的结点数是 m-1,则{S2，S3，…，Sm}和{P1，P2，…，Pm-1}可以唯一确定右子树，从而也确定了二叉树。

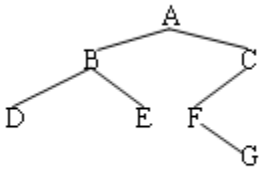
若 i=m,则 Sm 是根，这时二叉树的右子树为空，左子树的结点数是 m-1，则{S1，S2，…，Sm-1}和{P1，P2，…，Pm-1}唯一确定左子树，从而也确定了二叉树。

最后，当 1<i<m 时，Si 把中序序列分成{S1，S2，…，Si-1}和{Si+1,Si+2,…，Sm}。由于后序遍历是“左子树—右子树—根结点”，所以{P1,P2,…,Pi-1}和{Pi,Pi+1,…Pm-1}是二叉树的左子树和右子树的后序遍历序列。因而由{S1，S2，…，Si-1}和{P1，P2，…，Pi-1}可唯一确定二叉树的左子树，由{Si+1,Si+2,…，Sm}和{Pi,Pi+1,…，Pm-1}可唯一确定二叉树的右子树 。

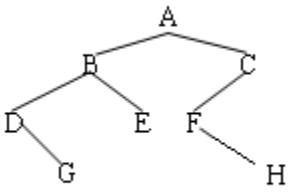
由中序序列 DBEAFGC 和后序序列 DEBGFCA 构造的二叉树如右图：

41．证明请参见第 40 题和第 38 题

由前序序列 ABDGECFH 和中序序列 DGBEAFHC 构造的二叉树如图：



第40题图



第41题图

42．参见第 38 题

43．先序遍历二叉树的顺序是“根—左子树—右子树”，中序遍历“左子树—根—右子树”，后序遍历顺序是：“左子树—右子树－根＂，根据以上原则，本题解答如下：

（1） 若先序序列与后序序列相同，则或为空树，或为只有根结点的二叉树

（2） 若中序序列与后序序列相同，则或为空树，或为任一结点至多只有左子树的二叉树.

（3） 若先序序列与中序序列相同，则或为空树，或为任一结点至多只有右子树的二叉树.

（4） 若中序序列与层次遍历序列相同，则或为空树，或为任一结点至多只有右子树的二叉树

由中序序列 DBEAFIHCG 和后序序列 DEBHIFGCA 确定的二叉树略。

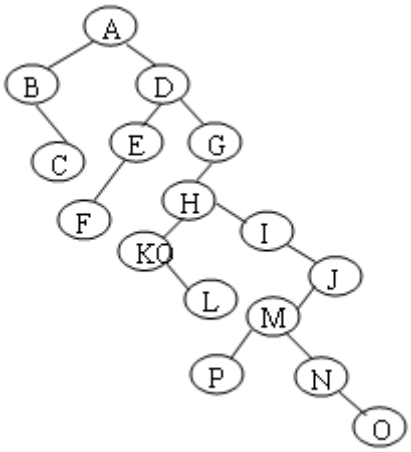
44．森林转为二叉树的三步：

（1）连线（将兄弟结点相连，各树的根看作兄弟）；

（2）切线（保留最左边子女为独生子女，将其它子女分枝切掉）；

（3）旋转（以最左边树的根为轴，顺时针向下旋转 45 度）。

其实经过（1）和（2），已转为二叉树，执行（3）只是为了与平时的二叉树的画法一致。

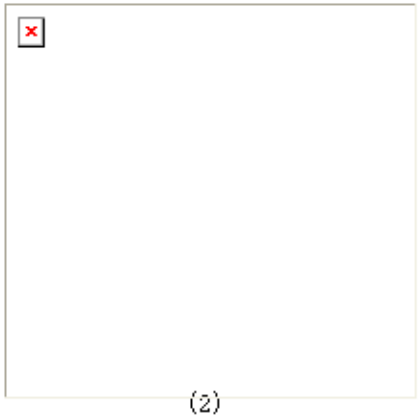


45．（1）①tree[p]·l→p　② tree[p]·r→p　③ p=0

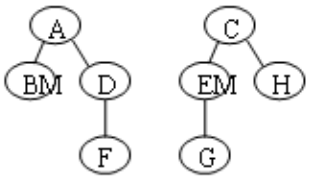
（2）框（A）移至Ⅲ处，成为前序遍历。

46.

(1)

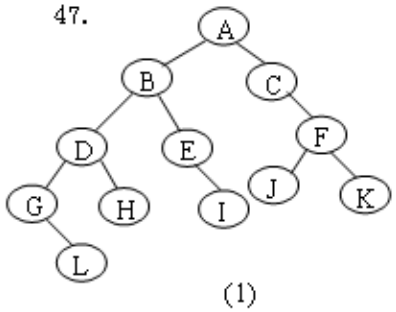


(2)

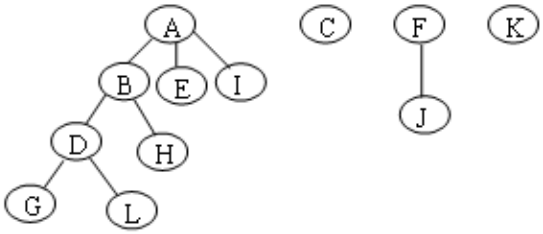


(3)

47.

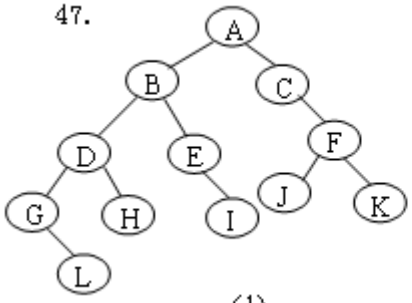


(1)

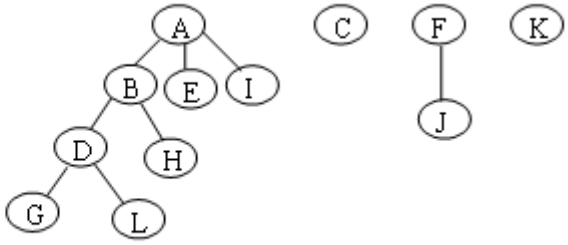


(2)

47.

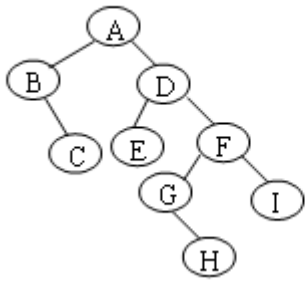


(1)



(2)

48. (1)



（2）设二叉树的前序遍历序列为 P1，P2，…，Pm，中序遍历序列为 S1,S2,…，Sm。因为前序遍历是“根左右”，中序遍历是“左根右”，则前序遍历序列中第一个结点是根结点（P1）。到中序序列中查询到 Si=P1,根据中序遍历时根结点将中序序列分成两部分的原则，有：

若 i=1，即 S1=P1，则这时的二叉树没有左子树；否则，S1,S2,…,Si-1 是左子树的中序遍历序列,用该序列和前序序列 P2,P3，…，Pi 去构造该二叉树的左子树。

若 i=m,即 Sm=P1,则这时的二叉树没有右子树；否则,Si+1,Si+2,…，Sm 是右子树的中序遍历序列，用该序列和前序序列中 Pi+1,Pi+2,…,Pm,去构造该二叉树的右子树。算法描

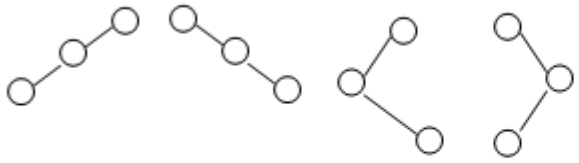
述请参见下面算法设计第 56 题。

（3）若前序序列是 abcd，并非由这四个字母的任意组合（4!=24）都能构造出二叉树。因为以 abcd 为输入序列，通过栈只能得到 1/(n+1)*2n!/(n!*n!)=14 种，即以 abcd 为前序序列的二叉树的数目是 14。任取以 abcd 作为中序遍历序列，并不全能与前序的 abcd 序列构成二叉树。例如：若取中序列 dcab 就不能。

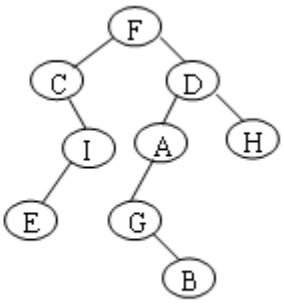
该 14 棵二叉树的形态及中序序列略。

49．不能。因 DABC 并不是 ABCD 的合法出栈序列，参照第 37、48（3）的解释。

50．先序序列是“根左右” 后序序列是“左右根”，可见对任意结点，若至多只有左子女或至多只有右子女，均可使前序序列与后序序列相反，图示如下：



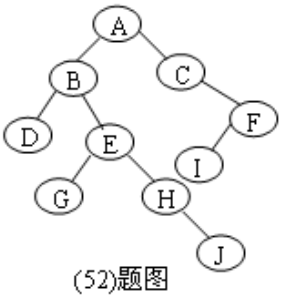
50题



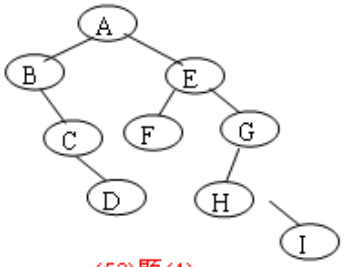
51 题

51．

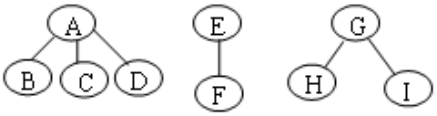
52．按层次遍历，第一个结点（若树不空）为根，该结点在中序序列中把序列分成左右两部分—左子树和右子树。若左子树不空，层次序列中第二个结点左子树的根；若左子树为空，则层次序列中第二个结点右子树的根。对右子树也作类似的分析。层次序列的特点是：从左到右每个结点或是当前情况下子树的根或是叶子。



(52)题图

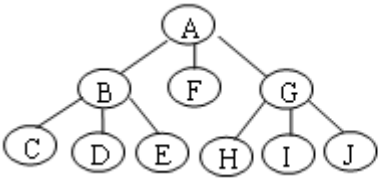


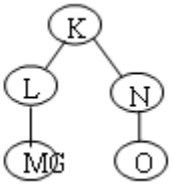
(52)题(1)



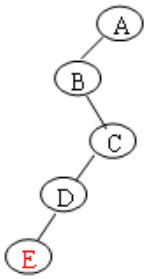
(52题(1)对应森林)

53．森林的先序序列和后序序列对应其转换的二叉树的先序序列和中序序列，应先据此构造二叉树，再构造出森林。



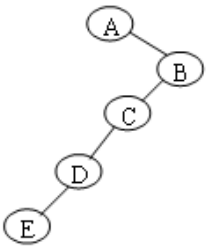


54.



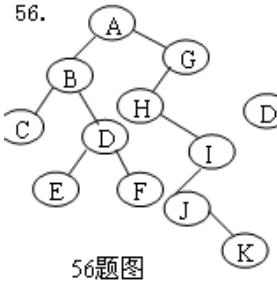
(54)题图

53题森林

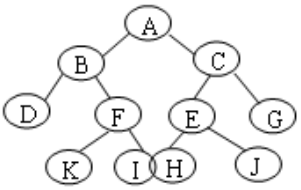


55. HIDJKEBLFGCA

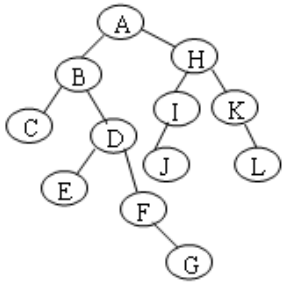
56.



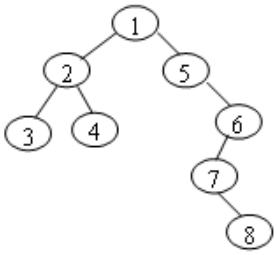
56题图



56 (1)



56 (2)



56 (3)

57. M 叉树的前序和后序遍历分别与它转换成的二叉树的先序和中序遍历对应。

58. 前序遍历是“根左右”，中序遍历是“左根右”，后序遍历是“左右根”。若将“根”去掉，三种遍历就剩“左右”。三种遍历中的差别就是访问根结点的时机不同。二叉树是递归定义的，对左右子树均是按左右顺序来遍历的，因此所有叶子结点间的先后关系都是相同的。

59. 本题的核心是三种遍历的顺序：“根左右”-“左根右”-“左右根”，但对本题的解答必须先定义结点间相互关系的“左右”。本解答中将 N 是 M 的左子女，当作 N 在 M 的左边，而 N 是 M 的右子女，当作 N 在 M 的右边。若定义 P 是 M 和 N 的最近公共祖先，N 在 P 的左子树中，M 在 P 的右子树中，称 N 在 M 的左边，那时的答案是不一样的。

	先根遍历时n先被访问	中根遍历时n先被访问	后根遍历时n先被访问
N在M的左边		√	√
N在M的右边			√
N是M的祖先			
N是M的子孙			√

60. HIDJKEBLFGCA

61.

62. 后序遍历的顺序是“左子树—右子树—根结点”。因此，二叉树最左下的叶子结点是遍历的第一个结点。下面的语句段说明了这一过程（设 p 是二叉树根结点的指针）。

```
if(p!=null)
  {while (p->lchild!=null || p->rchild!=null)
     {while(p->lchild!=null) p=p->lchild;
      if(p->rchild!=null) p=p->rchild;  } }
return(p);    //返回后序序列第一个结点的指针;
```

63. 采用前序和后序两个序列来判断二叉树上结点 n1 必定是结点 n2 的祖先。

在前序序列中某结点的祖先都排在其前。若结点 n1 是 n2 的祖先，则 n1 必定在 n2 之前。而在后序序列中，某结点的祖先排在其后，即若结点 n1 是 n2 的祖先，则 n1 必在 n2 之后。

根据这条规则来判断若结点 n1 在前序序列中在 n2 之前，在后序序列中又在 n2 之后，则它必是结点 n2 的祖先。

64.(1)　　（2） 前序序列：ABCEDFHGIJ　　（3）后序线索树

中序序列：ECBHFDJIGA

后序序列：ECHFJIGDBA

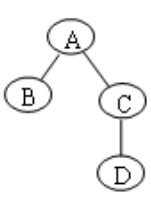
65. 最后一个递归调用语句所保留的参数没有意义。这类递归因其在算法最后，通常被称为“尾递归”， 可不用栈且将其（递归）消除。 如中序遍历递归算法中，最后的递归语句 inorder (bt->rchild)可改为下列语句段：

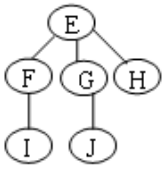
```
bt=bt->rchild;
while (bt->rchild!=null)
 {inorder (bt ->lchild); printf(“%c”,bt ->data);//访问根结点，假定结点数据域为字符
  bt=bt ->rchild; }
```

66.在二叉链表表示的二叉树中， 引入线索的目的主要是便于查找结点的前驱和后继。因为若知道各结点的后继，二叉树的遍历就变成非常简单。二叉链表结构查结点的左右子女非常方便，但其前驱和后继是在遍历中形成的。为了将非线性结构二叉树的结点排成线性序列，利用结点的空链域，左链为空时用作前驱指针，右链为空时作为后继指针。再引入左右标记 ltag 和 rtag ，规定 ltag=0，lchild 指向左子女,ltag=1 时,lchild 指向前驱；当 rtag=0 时 ,rchild 指向右子女，rtag=1 时，rchild 指向后继。这样，在线索二叉树（特别是中序线索二叉树）上遍历就消除了递归，也不使用栈（后序线索二叉树查后继仍需要栈。）

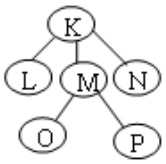
67.

67题(1)



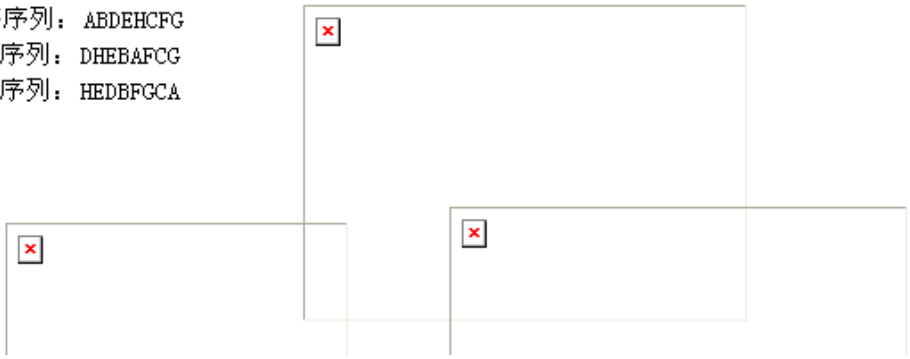


67题(2)



(3)后根遍历森林，结点序列为：
BDCAIFJGHELOPMNK

68. （1）前序序列：ABDEHCFG
（2）中序序列：DHEBAFCG
（3）后序序列：HEDBFGCA



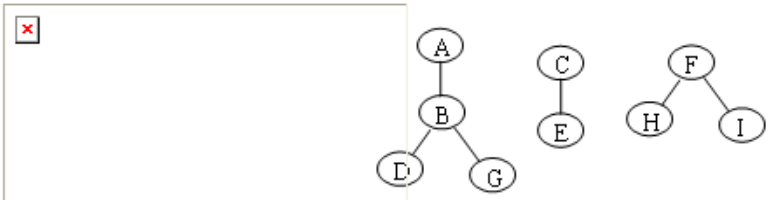
69. （1）

69. （1）

（2）BiTree INORDER-PRIOR(N,X) //在中序线索二叉树上查找结点 N 的前驱结点 X

```
{if(n->ltag==1){X=N->lchild; return (X);}
 else {p=N->lchild;
       while (p->rtag==0) p=p->rchild;
       X=p;return(p);}  }
```

70.



70题（3）

71.

前序序列：A B C E D F H G I J
中序序列： E C B H F D J I G A
后序序列： E C H F J I G D B
A

71题（2）

72. 后序线索树中结点的后继，要么是其右线索（当结点的 rtag=1 时），要么是其双亲结点右子树中最左下的叶子结点。所以，只有当二叉树只有根或树中任一结点均无右子树时，进行遍历才不用栈，其遍历程序段如下：

```
while（p->ltag==0）p==p->lchild; //找最左下叶子结点
while（p->rchild!=null）{visit(p->data); //访问结点；
 p=p->rchild;} //沿线索向上
```

对前序线索二叉树，当二叉树非空时，若其结点有左子女（ltag=0），左子女是后继；否则，若结点有右子女（rtag=0），则右子女是后继；若无右子女（rtag=1），右子女是线索，也指向后继。所以，对任何前序线索二叉树进行前序遍历均不需使用栈。

73. 左右子树均不空的二叉树先序线索化后，空指针域为 1 个（最后访问结点的右链为空）。

74.
```
if(p->ltag==0) return(p->lchild);//左子女不空，左子女为直接后继结点
else return(p->rchild);          //左子女空，右子女（或右线索）为后继
```

75. 后序线索树中结点的后继（根结点无后继），要么是其右线索（当结点的 rtag=1 时），要么是其双亲结点右子树中最左下的叶子结点。对中序线索二叉树某结点，若其左标记等于 1,则左子女为线索，指向直接前驱；否则，其前驱是其左子树上按中序遍历的最后一个结点。

76. 树的后根遍历（对应二叉树的中序遍历）全线索链表

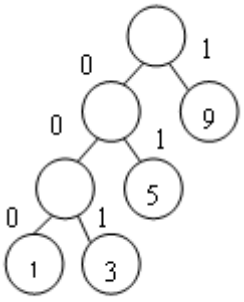
Data	A	B	C	D	E	F	G	H	I	J	K
Ltag	0	1	0	0	0	1	0	1	1	1	1
Fch	2	null	5	7	8	5	11	2	8	9	3
Rtag	1	0	0	1	0	1	1	0	0	1	1
Nsib	null	3	4	1	6	3	4	9	10	5	7

77.

虽然哈夫曼树的带权路径长度是唯一的，但形态不唯一。本题中各字母编码如下：c1:0110 c2:10 c3:0010 c4:0111 c5:000 c6:010 c7:11 c8:0011

78. 字符 A，B，C，D 出现的次数为 9,1,5,3。其哈夫曼编码如下 A:1，B:000，C:01，D:001

79.（2）wpl=(2+3)*5+6*4+(9+14+15)*3+(16+17)*2=229



78题哈夫曼编码树

3）编码为：15:111，3:10101，14:110，2:10100，6:1011，9:100，16:00，17:01

(4) 常用哈夫曼树为通讯用的字符编码，本题中集合的数值解释为字符发生的频率（次数）。由哈夫曼树构造出哈夫曼编码。译码时，进行编码的“匹配”，即从左往右扫描对方发来的“编码串”，用字符编码去匹配，得到原来的元素（本题中的数）。

80. 首先确定是否需加“虚权值”(即权值为 0),对 m 个权值，建 k 叉树，若（m-1）%(k-1)=0,则不需要加“虚权值”，否则，第一次归并时需（m-1）%(k-1)+1 个权值归并。建立 k 叉树的过程如下：

（1）将 m 个权值看作 m 棵只有根结点的 k 叉树的集合 F={T1，T2，…，Tm}。

（2）从 F 中选 k（若需加虚权值，则第一次选（m-1）%(k-1)+1）个权值最小的树作子树，构成一棵 k 叉树，k 叉树根结点的权值为所选的 k 个树根结点权值之和，在 F 中删除这 k 棵子树，并将新 k 叉树加入到 F 中。

（3）从 F 中选 k 个权值最小的树作子树，构成一棵 k 叉树，其根结点权值等于所选的 k 棵树根结点权值之和，在 F 中删除这 k 棵树，并将新得到的树加到 F 中。

(4) 重复（3），直到 F 中只有一棵树为止，这就是最优的 k 叉树。

对本题 10 个权值，构造最优三叉树。

因（10-1）%(3-1)=1，

所以第一次用 2 个权值合并。

最小加权路径长度：

（1+4）*4+(9+16)*3+(25+36+49+64+81)*2+100*1=705

	字符	weight	Parent	lch	rch
1	A	3	0	0	0
2	B	12	0	0	0
3	C	7	0	0	0
4	D	4	0	0	0
5	E	2	0	0	0
6	F	8	0	0	0
7	G	11	0	0	0
8			0	0	0
9			0	0	0
10			0	0	0
11			0	0	0
12			0	0	0
13			0	0	0

	字符	weight	parent	lch	rch
1	A	3	8	0	0
2	B	12	12	0	0
3	C	7	10	0	0
4	D	4	9	0	0
5	E	2	8	0	0
6	F	8	10	0	0
7	G	11	11	0	0
8		5	9	5	1
9		9	11	4	8
10		15	12	3	6
11		20	13	9	7
12		27	13	2	10
13		47	0	11	12

81．初态图　　终态图

82．前缀码是一编码不是任何其它编码前缀的编码。例如，0 和 01 就不是前缀码，因为编码 0 是编码 01 的前缀。仅从编码来看，0 和 01 是前缀码，但因历史的原因，它不被称为前缀码，而是把一编码不是另一编码前缀的编码称为前缀码。

利用二叉树可以构造前缀码，例如，以 A，B，C，D 为叶子可构成二叉树，将左分枝解释为 0，右分枝解释成 1，从根结点到叶子结点的 0、1 串就是叶子的前缀码。用哈夫曼树可构造出最优二叉树，使编码长度最短。

83．哈夫曼树只有度为 0 的叶子结点和度为 2 的分枝结点，设数量分别为 n0 和 n2，则树的结点数 n 为 n=n0+n2。 另根据二叉树性质：任意二叉树中度为 0 的结点数 n0 和度为 2 的结点数 n2 间的关系是 n2=n0-1,代入上式，则 n=n0+n2=2n0-1。

84．（1）T 树的最大深度 Kmax=6（除根外，每层均是两个结点）

T 树的最小深度 Kmin=4（具有 6 个叶子的完全二叉树是其中的一种形态）

（2）非叶子结点数是 5。(n2=n0-1)　（3）哈夫曼树见下图，其带权路径长度 wpl=51

85.（1）错误，循环结束条件 top=0 不能满足，因为在 top>1 情况下，执行 top:=top-1
(2)错误　（3）错误　（4）正确　（5）结点的深度与其右孩子深度相同，比左孩子深度少 1。

五．算法设计题

1．[题目分析]以二叉树表示算术表达式，根结点用于存储运算符。若能先分别求出左子树和右子树表示的子表达式的值，最后就可以根据根结点的运算符的要求，计算出表达式的最后结果。

```
typedef struct node
   {ElemType  data;  float val;
    char optr;  //只取 '+',  '-',  '*', '/'
    struct node *lchild, *rchild }BiNode,*BiTree;
float PostEval(BiTree bt)   // 以后序遍历算法求以二叉树表示的算术表达式的值
  {float lv,rv;
   if(bt!=null)
   {lv=PostEval(bt->lchild); // 求左子树表示的子表达式的值
    rv=PostEval(bt->rchild); // 求右子树表示的子表达式的值
    switch(bt->optr)
   {case  '+' : value=lv+rv; break;
    case   '-' : value=lv-rv;break;
    case   '*' : value=lv*rv;break;
    case   '/' : value=lv/rv;
  } } return(value); }
```

2.[题目分析] 本题是将符号算术表达式用二叉树表示的逆问题，即将二叉树表示的表达式还原成原表达式。二叉树的中序遍历序列与原算术表达式基本相同，差别仅在于二叉树表示中消除了括号。将中序序列加上括号就恢复原貌。当根结点运算符优先级高于左子树(或右子树）根结点运算符时，就需要加括号。

```
    int Precede(char optr1,optr2)
     // 比较运算符级别高低，optr1 级别高于 optr2 时返回 1，相等时返回 0，低于时返回-1
      {switch(optr1)
          {case'+':case'-':if(optr2=='+'||optr2=='-')return(0);else return(-1);
           case'*':case'/':if(optr1=='*'||optr2=='/')return(0);else return(1);
          }  }
    void InorderExp (BiTree bt)
       //输出二叉树表示的算术表达式，设二叉树的数据域是运算符或变量名
       {int bracket;
        if(bt)
         {if(bt->lchild!=null)
           {bracket=Precede(bt->data,bt->lchild->data)//比较双亲与左子女运算符优先级
            if(bracket==1) printf( '(' );
            InorderExp(bt->lchild);          //输出左子女表示的算术表达式
            if(bracket==1)printf( ')' );    //加上右括号
```

```
        }
        printf(bt->data);                    //输出根结点
        if(bt->rchild!=null)                 //输出右子树表示的算术表达式
          {bracket=Precede(bt->data,bt->rchild->data)
           if (bracket==1)printf(“(”);   //右子女级别低，加括号
           InorderExp (bt->rchild);
           if(bracket==1)printf(“)”);
        } }
      }//结束 Inorder Exp
```

3.[题目分析]首先通过对二叉树后序遍历形成后缀表达式，这可通过全局变量的字符数组存放后缀表达式；接着对后缀表达式求值，借助于一栈存放运算结果。从左到右扫描后缀表达式，遇操作数就压入栈中，遇运算符就从栈中弹出两个操作数，作运算符要求的运算，并把运算结果压入栈中，如此下去，直到后缀表达式结束，这时栈中只有一个数，这就是表达式的值。

```
char ar[maxsize];//maxsize 是后缀表达式所能达到的最大长度
int i=1;
void  PostOrder(BiTree  t )//后序遍历二叉树 t,以得到后缀表达式
 {if(t)
  {PostOrder(t->lchild); PostOrder(b->rchild); ar[i++]=b->data; }
 }//结束 PostOrder
void  EXPVALUE()
//对二叉树表示的算术表达式，进行后缀表达式的求值
{ar[i]=‘\0’;       //给后缀表达式加上结束标记
 char value[];      //存放操作数及部分运算结果
 i=1; ch=ar[i++];
 while(ch!=‘\0’)
   {switch(ch)
      {case ch in op: opnd1=pop(value);opnd2=pop(value); //处理运算符
                      push(operate(opnd2,ch,opnd1));break;
       default:       push(value,ch);                    //处理操作数，压入栈中
      }
      ch=ar[i++];                  //读入后缀表达式
   } printf(value[1]);             //栈中只剩下一个操作数，即运算结束。
 } //结束 EXPVALUE
```

[算法讨论] 根据题意，操作数是单字母变量，存放运算结果的栈也用了字符数组。实际上，操作数既可能是变量，也可以是常量。运算中，两个操作数（opnd1 和 opnd2）也不会直接运算，即两个操作数要从字符转换成数（如‘3’是字符，而数值 3=‘3’-‘0’）。数在压入字符栈也必须转换，算法中的 operate 也是一个需要编写的函数，可参见上面算法设计题 1，其细节不再深入讨论。

4.[题目分析]当森林（树）以孩子兄弟表示法存储时，若结点没有孩子（fch=null），则它必是叶子，总的叶子结点个数是孩子子树（fch）上的叶子数和兄弟（nsib）子树上叶结点个数之和。

```
typedef  struct  node
```

```
      {ElemType  data;//数据域
       struct  node *fch, *nsib;//孩子与兄弟域 }*Tree;
  int Leaves (Tree t)
     //计算以孩子-兄弟表示法存储的森林的叶子数
    {if(t)
     if(t->fch==null)                //若结点无孩子，则该结点必是叶子
        return(1+Leaves(t->nsib));  //返回叶子结点和其兄弟子树中的叶子结点数
     else return (Leaves(t->fch)+Leaves(t->nsib)); //孩子子树和兄弟子树中叶子数之和
    }//结束 Leaves
```

5. [题目分析]由指示结点 i 左儿子和右儿子的两个一维数组 L[i]和 R[i]，很容易建立指示结点 i 的双亲的一维数组 T[i]，根据 T 数组，判断结点 U 是否是结点 V 后代的算法，转为判断结点 V 是否是结点 U 的祖先的问题。

```
  int Generation (int U,V,N,L[],R[],T[])
    //L[]和 R[]是含有 N 个元素且指示二叉树结点 i 左儿子和右儿子的一维数组，
    //本算法据此建立结点 i 的双亲数组 T，并判断结点 U 是否是结点 V 的后代。
    {for(i=1;i<=N;i++)  T[i]=0;  //T 数组初始化
     for (i=1;i<=N;i++)          //根据 L 和 R 填写 T
      if(L[i]!=0) T[L[i]]=i;     //若结点 i 的左子女是 L，则结点 L 的双亲是结点 i
       for(i=1;i<=N;i++)
         if (R[i]!=0) T[R[i]]=i;   //i 的右子女是 r，则 r 的双亲是 i
       int parent=U;               //判断 U 是否是 V 的后代
       while (parent!=V && parent!=0) parent=T[parent];
       if (parent==V){printf(“结点 U 是结点 V 的后代”);return(1);}
       else{ printf(“结点 U 不是结点 V 的后代”);return(0);}
      }结束 Generation
```

6. [题目分析]二叉树是递归定义的，以递归方式建立最简单。判定是否是完全二叉树，可以使用队列，在遍历中利用完全二叉树“若某结点无左子女就不应有右子女”的原则进行判断。

```
  BiTree Creat()       //建立二叉树的二叉链表形式的存储结构
  {ElemType x; BiTree bt;
    scanf(“%d”,&x);  //本题假定结点数据域为整型
    if(x==0) bt=null;
    else if(x>0)
           {bt=(BiNode *)malloc(sizeof(BiNode));
            bt->data=x; bt->lchild=creat(); bt->rchild=creat();
           }
          else error(“输入错误”);
    return(bt);
   }//结束 BiTree
  int JudgeComplete(BiTree bt)  //判断二叉树是否是完全二叉树,如是，返回 1，否则，返回 0
       {int tag=0; BiTree p=bt, Q[]; // Q 是队列，元素是二叉树结点指针，容量足够大
```

```
        if(p==null) return (1);
        QueueInit(Q); QueueIn(Q,p);  //初始化队列，根结点指针入队
        while (!QueueEmpty(Q))
          {p=QueueOut(Q);                                    //出队
           if (p->lchild && !tag) QueueIn(Q,p->lchild);  //左子女入队
           else {if (p->lchild) return 0;                  //前边已有结点为空，
本结点不空
                 else tag=1;                                  //首次出现结点为空
           if (p->rchild && !tag) QueueIn(Q,p->rchild);  //右子女入队
           else if (p->rchild) return 0; else tag=1;
           } //while
        return 1;  } //JudgeComplete
```

[算法讨论]完全二叉树证明还有其它方法。判断时易犯的错误是证明其左子树和右子数都是完全二叉树，由此推出整棵二叉树必是完全二叉树的错误结论。

7. BiTree Creat(ElemType A[],int i)

//n 个结点的完全二叉树存于一维数组 A 中，本算法据此建立以二叉链表表示的完全二叉树

```
{BiTree tree;
if (i<=n){tree=(BiTree)malloc(sizeof(BiNode)); tree->data=A[i];
          if(2*i>n) tree->lchild=null; else tree->lchild=Creat(A,2*i);
          if(2*i+1>n) tree->rchild=null;else tree->rchild=Creat(A,2*i+1);   }
return (tree);  }//Creat
```

[算法讨论]初始调用时,i=1。

8. [题目分析]二叉树高度可递归计算如下：若二叉树为空，则高度为零，否则，二叉树的高度等于左右子树高度的大者加 1。这里二叉树为空的标记不是 null 而是 0。设根结点层号为 1，则每个结点的层号等于其双亲层号加 1。

现将二叉树的存储结构定义如下:

```
typedef struct node
    {int L[];//编号为 i 的结点的左儿子
     int R[];//编号为 i 的结点的右儿子
     int D[];//编号为 i 的结点的层号
     int i;  //存储结点的顺序号（下标）
    }tnode;
```

（1）

```
int Height(tnode t,int i)//求二叉树高度，调用时 i=1
    {int lh,rh;
     if (i==0) return (0);
     else{lh=Height(t,t.L[i]); rh=Height(t,t.R[i]);
          if(lh>rh) return(lh+1); else return(rh+1);
          }
    }//结束 Height
```

（2）

```
int Level（tnode t）//求二叉树各结点的层号，已知编号为 1 的结点是根，且层号为 1
    {t.D[1]=1;
     for(i=1;i<=n;i++) {depth=t.D[i];          //取出根结点层号
```

```
    if(t.L[i]!=0)    t.D[t.L[i]]=depth+1;    //i 结点左儿子层号
    if(t.R[i]!=0)    t.D[t.R[i]]=depth+1;  }//i 结点右儿子层号
}结束 level
```

9.[题目分析]二叉树采用顺序存储结构（一维数组）是按完全二叉树的形状存储的，不是完全二叉树的二叉树顺序存储时，要加“虚结点”。数组中的第一个元素是根结点。本题中采用队列结构。

```
typedef struct
  {BiTree bt;          //二叉树结点指针
   int num; }tnode    // num 是结点在一维数组中的编号
tnode  Q[maxsize];    //循环队列,容量足够大
void  creat(BiTree T,ElemType BT[ ])
    //深度 h 的二叉树存于一维数组 BT[1:2^h-1]中,本算法生成该二叉树的二叉链表存储结构
  {tnode tq;          //tq 是队列元素
   int len=2^h-1;     //数组长度
   T=(BiTree)malloc(sizeof(BiNode)); //申请结点
   T->data=BT[1];                    //根结点数据
   tq.bt=T; tq.num=1;
   Q[1]=tq;                  //根入队列
   front=0;rear=1;           //循环队列头、尾指针
   while(front!=rear)        //当队列不空时循环
    {front=(front+1) % maxsize ;
     tq=Q[front] ; p=tq.bt; i=tq.num ;              //出队，取出结点及编号
     if (BT[2*i]==‘#’||2*i>len) p->lchild=null;  //左子树为空，‘#’表示虚结点
     else   //建立左子女结点并入队列
      {p->lchild=(BiTree) malloc(sizeof(BiNode)); //申请结点空间
       p->lchild→data=BT[2*i];                    // 左子女数据
       tq.bt=p->lchild; tq.num=2*i; rear=(rear+1) % maxsize ;//计算队尾位置
       Q[rear]=tq;             //左子女结点及其编号入队
       }
     if(BT[2*i+1]==‘#’ || 2*i+1>len) p->rchild=null; //右子树为空
      else    //建立右子女结点并入队列
        {p->rchild=(BiTree)malloc(sizeof (BiNode); //申请结点空间
         p->rchild->data=BT[2*i+1]; tq.bt=p->rchild; tq.num=2*i+1;
         rear=(rear+1)%maxsize; Q[rear]=tq;  //计算队尾位置,右子女及其编号入队
         }
     } //while
  }//结束 creat
```

[算法讨论] 本题中的虚结点用‘#’表示。应根据二叉树的结点数据的类型而定。

10.[题目分析]本题静态链表中结点是按动态二叉链表的前序遍历顺序存放的。首先对动态二叉链表的二叉树进行前序遍历，填写静态链表的“下标”和 data 域，再对动态二叉链表的二叉树进行层次遍历，设队列 Q，填写静态链表的 lchild 域和 rchild 域。

```
typedef struct node          //静态链表结点结构
  {ElemType data;            //结点数据
   int row,lchild,rchild ; //下标，左右子女
  }component;
component st[];  //st 容量足够大
struct node {BiTree t; int idx; }qbt;
static int num=0;
void PreOrder(BiTree bt);
// 前序遍历二叉树，填写静态链表的“下标”和 data 域
 {if (bt)
   {st[++num].data=bt->data; st[num].row=num;
    PreOrder(bt->lchild); PreOrder(bt->rchild);
   } }
int Locate(ElemType x)
   //在静态链表中查二叉树结点的下标
   {for (i=1;i<=num;i++) if (st[i].data==x) return (i);
    }
void DynaToST (BiTree bt)  //将二叉树的动态二叉链表结构转为静态链表结构
 {int i=0;        //i 为静态链表 st 的下标
  if (bt!=null)
   {QueueInit(Q);  //Q 是队列，容量足够大，队列元素是 qbt
    qbt.t=bt; qbt.idx=1; QueueIn(Q,qbt); st[1].data=bt->data;
    while(!QueueEmpty(Q))
      {qbt=QueueOut(Q);        //出队列
       p=qbt.t; i=qbt.idx;     //二叉树结点及在静态链表中的下标
       if (p->lchild!=null)    //若左子女存在,查其在静态链表中的下标, 填
   lchild 域值
           {lch=Locate(p->lchild->data); st[i].lchild=lch;
            qbt.t=p->lchild; qbt.idx=lch;  QueueIn(Q,qbt); }
       else st[i].lchild=0;   //无左子女，其 lchild 域填 0
       if (p->rchild!=null)    //若右子女存在,查其在静态链表中的下标，填
   rchild 域值
            {rch=Locate(p->->rchild->data); st[i].rchild=rch;
             qbt.t=p->rchild; qbt.idx=rch;  QueueIn(Q,qbt); }
       else st[i].rchild=0;   //无左子女，其 lchild 域填 0
      }//while
   }//结束 DynaToST
```

11. [题目分析] 由于以双亲表示法作树的存储结构，找结点的双亲容易。因此我们可求出每一结点的层次，取其最大层次就是树有深度。对每一结点，找其双亲，双亲的双亲，直至（根）结点双亲为 0 为止。

```
int Depth(Ptree t)    //求以双亲表示法为存储结构的树的深度，Ptree 的定义参看
教材
{int maxdepth=0;
 for(i=1;i<=t.n;i++)
```

```
{temp=0; f=i;
 while(f>0) {temp++; f=t.nodes[f].parent; }      // 深度加 1,并取新的双亲
 if(temp>maxdepth)  maxdepth=temp; //最大深度更新
 }
 return(maxdepth);//返回树的深度
} //结束 Depth
```

12. [题目分析] 二叉树是递归定义的，其运算最好采取递归方式

```
int Height(btre bt)//求二叉树 bt 的深度
{int hl,hr;
 if (bt==null) return(0);
   else {hl=Height(bt->lch); hr=Height(bt->rch);
         if(hl>hr) return (hl+1); else return(hr+1);
     }  }
```

13. [题目分析] 求二叉树高度的算法见上题。求最大宽度可采用层次遍历的方法，记下各层结点数，每层遍历完毕，若结点数大于原先最大宽度，则修改最大宽度。

```
int Width(BiTree bt)//求二叉树 bt 的最大宽度
{if (bt==null) return (0);  //空二叉树宽度为 0
 else
 {BiTree Q[];//Q 是队列，元素为二叉树结点指针，容量足够大
   front=1;rear=1;last=1;//front 队头指针,rear 队尾指针,last 同层最右结点在队列中的位置
   temp=0; maxw=0;        //temp 记局部宽度, maxw 记最大宽度
   Q[rear]=bt;            //根结点入队列
   while(front<=last)
   {p=Q[front++]; temp++; //同层元素数加 1
    if (p->lchild!=null)  Q[++rear]=p->lchild;   //左子女入队
       if (p->rchild!=null)  Q[++rear]=p->rchild;   //右子女入队
    if (front>last)       //一层结束,
     {last=rear;
         if(temp>maxw) maxw=temp;//last 指向下层最右元素, 更新当前最大宽度
      temp=0;
      }//if
    }//while
   return (maxw);
 }//结束 width
```

14. [题目分析]由孩子兄弟链表表示的树，求高度的递归模型是：若树为空，高度为零；若第一子女为空，高度为 1 和兄弟子树的高度的大者；否则，高度为第一子女树高度加 1 和兄弟子树高度的大者。其非递归算法使用队列，逐层遍历树，取得树的高度。

```
int Height(CSTree bt)      //递归求以孩子兄弟链表表示的树的深度
{int hc,hs;
 if (bt==null) return (0);
 else if (!bt->firstchild) return (1+height(bt->nextsibling);//子女空，查兄弟的深度
        else  // 结点既有第一子女又有兄弟，高度取子女高度+1 和兄弟子树高度的大
```

```
者
        {hc=height(bt->firstchild);  //第一子女树高
         hs=height(bt->nextsibling); //兄弟树高
         if(hc+1>hs)return(hc+1);  else return (hs);
         }
 }//结束 height
int height(CSTree t)      //非递归遍历求以孩子兄弟链表表示的树的深度
{if(t==null)  return(0);
 else{int front=1,rear=1;  //front,rear 是队头队尾元素的指针
      int last=1,h=0;      //last 指向树中同层结点中最后一个结点，h 是树的高度
      Q[rear]=t;           //Q 是以树中结点为元素的队列
      while(front<=last)
       {t=Q[front++];      //队头出列
        while(t!=null)     //层次遍历
         {if (t->firstchild) Q[++rear]=t->firstchild; //第一子女入队
          t=t->nextsibling; //同层兄弟指针后移
          }
         if(front>last)      //本层结束，深度加 1（初始深度为 0）
            {h++;last=rear;} //last 再移到指向当前层最右一个结点
       }//while(front<=last)
      }//else
    }//Height
```

15.[题目分析]后序遍历最后访问根结点，即在递归算法中，根是压在栈底的。采用后序非递归算法，栈中存放二叉树结点的指针，当访问到某结点时，栈中所有元素均为该结点的祖先。本题要找 p 和 q 的最近共同祖先结点 r，不失一般性，设 p 在 q 的左边。后序遍历必然先遍历到结点 p，栈中元素均为 p 的祖先。将栈拷入另一辅助栈中。再继续遍历到结点 q 时，将栈中元素从栈顶开始逐个到辅助栈中去匹配，第一个匹配（即相等）的元素就是结点 p 和 q 的最近公共祖先。

```
typedef struct
  {BiTree t;int tag;//tag=0  表示结点的左子女已被访问，tag=1 表示结点的右子女已被访问
  }stack;
stack s[],s1[];//栈，容量够大
BiTree Ancestor(BiTree ROOT,p,q,r)//求二叉树上结点 p 和 q 的最近的共同祖先结点 r。
 {top=0; bt=ROOT;
  while(bt!=null ||top>0)
  {while(bt!=null && bt!=p && bt!=q)                //结点入栈
    {s[++top].t=bt; s[top].tag=0; bt=bt->lchild;} //沿左分枝向下
   if(bt==p)  //不失一般性，假定 p 在 q 的左侧,遇结点 p 时，栈中元素均为 p 的祖先结点
     {for(i=1;i<=top;i++) s1[i]=s[i]; top1=top; }//将栈 s 的元素转入辅助栈 s1 保存
   if(bt==q)  //找到 q 结点。
      for(i=top;i>0;i--)//；将栈中元素的树结点到 s1 去匹配
```

```
        {pp=s[i].t;
         for (j=top1;j>0;j--)
          if(s1[j].t==pp) {printf("p 和 q 的最近共同的祖先已找到"); return (pp);}
        }
     while(top!=0 && s[top].tag==1) top--;       //退栈
     if (top!=0) {s[top].tag=1;bt=s[top].t->rchild;}  //沿右分枝向下遍历
    }//结束 while(bt!=null ||top>0)
    return(null);// q 、p 无公共祖先
   } //结束 Ancestor
```

16.[题目分析]二叉树顺序存储，是按完全二叉树的格式存储，利用完全二叉树双亲结点与子女结点编号间的关系，求下标为 i 和 j 的两结点的双亲，双亲的双亲，等等，直至找到最近的公共祖先。

```
void Ancestor(ElemType A[],int n,i,j)
//二叉树顺序存储在数组 A[1..n]中，本算法求下标分别为 i 和 j 的结点的最近公共祖先结点的值。
  {while(i!=j)
     if(i>j) i=i/2;       //下标为 i 的结点的双亲结点的下标
     else j=j/2;          //下标为 j 的结点的双亲结点的下标
     printf("所查结点的最近公共祖先的下标是%d，值是%d",i,A[i]);//设元素类型整型。
  }// Ancestor
```

17.[题目分析]用二叉树表示出父子，夫妻和兄弟三种关系，可以用根结点表示父（祖先），根结点的左子女表示妻，妻的右子女表示子。这种二叉树可以看成类似树的孩子兄弟链表表示法；根结点是父，根无右子女，左子女表示妻，妻的右子女（右子女的右子女等）均可看成兄弟（即父的所有儿子），兄弟结点又成为新的父，其左子女是兄弟（父的儿子）妻，妻的右子女（右子女的右子女等）又为儿子的儿子等等。首先递归查找某父亲结点，若查找成功，则其左子女是妻，妻的右子女及右子女的右子女等均为父亲的儿子。

```
    BiTree Search(BiTree t,ElemType father)//在二叉树上查找值为 father 的结点
    {if(t==null) return (null);              //二叉树上无 father 结点
     else if(t->data==father) return(t);     //查找成功
     p=Search(t->lchild,father); p=Search(t->rchild,father); }
    }//结束 Search
    void PrintSons(BiTree t,ElemType p)  //在二叉树上查找结点值为 p 的所有的儿子
     {p=Serach(t,p);                     //在二叉树 t 上查找父结点 p
      if(p!=null)                        //存在父结点
        {q=p->lchild; q=q->rchild;       //先指向其妻结点，再找到第一个儿子
         while(q!=null) {printf(q->data); q=q->rchild;} //输出父的所有儿子
        }
      }//结束 PrintSons
```

18.[题目分析]后序遍历最后访问根结点，当访问到值为 x 的结点时，栈中所有元素均为该结点的祖先。

```
   void Search(BiTree bt,ElemType x) //在二叉树 bt 中，查找值为 x 的结点，并打印其所有祖先
    {typedef struct
```

```
{BiTree t; int tag; }stack;//tag=0 表示左子女被访问，tag=1 表示右子女被访问
 stack s[];  //栈容量足够大
 top=0;
 while(bt!=null||top>0)
   {while(bt!=null && bt->data!=x)                    //结点入栈
      {s[++top].t=bt; s[top].tag=0; bt=bt->lchild;} //沿左分枝向下
    if(bt->data==x){ printf("所查结点的所有祖先结点的值为:\n");   //找到
x
      for(i=1;i<=top;i++) printf(s[i].t->data);  return;  } //输出祖先值后
结束
    while(top!=0 && s[top].tag==1) top--;            //退栈（空遍历）
    if(top!=0) {s[top].tag=1;bt=s[top].t->rchild;}  //沿右分枝向下遍历
   }// while(bt!=null||top>0)
 }结束 search
```

因为查找的过程就是后序遍历的过程，使用的栈的深度不超过树的深度，算法复杂度为 O(logn)。

19.[题目分析] 先序遍历二叉树的非递归算法，要求进栈元素少，意味着空指针不进栈。

```
void PreOrder(Bitree  bt)//对二叉数 bt 进行非递归遍历
{int top=0; Bitree s[]; //top 是栈 s 的栈顶指针，栈中元素是树结点指针，栈容量
足够大
 while(bt!=null || top>0)
 {while(bt!=null)
   {printf(bt->data);  //访问根结点
    if(bt->rchlid) s[++top]=bt->rchild;  //若有右子女，则右子女进栈
    bt=bt->lchild; }
  if (top>0) bt=s[top--];
  }
```

本题中的二叉树中需进栈的元素有 C,H,K,F。

20.[题目分析]二叉树的顺序存储是按完全二叉树的顺序存储格式，双亲与子女结点下标间有确定关系。对顺序存储结构的二叉树进行遍历，与二叉链表类似。在顺序存储结构下，判二叉树为空时，用结点下标大于 n（完全二叉树）或 0（对一般二叉树的“虚结点”)。本题是完全二叉树。

```
void PreOrder(ElemType bt,int n)//对以顺序结构存储的完全二叉树 bt 进行前序遍
历
 {int top=0,s[];  //top 是栈 s 的栈顶指针，栈容量足够大
  while(i<=n||top>0)
  {while(i<=n)
    { printf(bt[i]);                    //访问根结点；
      if (2*i+1<=n) s[++top]=2*i+1;     //右子女的下标位置进栈
      i=2*i; }                          //沿左子女向下
     if(top>0) i=s[top--]; }            //取出栈顶元素
 }//结束 PreOrder
```

21.[题目分析] 本题使用的存储结构是一种双亲表示法，对每个结点，直接给出其双亲（的下标），而且用正或负表示该结点是双亲的右子女或左子女。该类结构不适于直接进行前序

遍历（即若直接前序遍历，算法要很复杂），较好的办法是将这类结构转为结点及其左右子女的顺序存储结构，即

```
Tree2=ARRAY[1..max] OF RECORD data: char;          //结点数据
                              lc,rc: integer; END;//结点的左右子女在数组中的下标
 void Change (Tree t, Tree2 bt, int *root) //先将 t 转为如上定义类型的变量 bt;
  {for(i=1;i<=max;i++) {bt[i].lc=bt[i].rc=0;} //先将结点的左右子女初始化为 0
   for(i=1;i<=max;i++)          //填入结点数据，和结点左右子女的信息
    {bt[i].data=t[i].data;
     if(t[i].parent<0) bt[-t[i].parent].lc=i;       //左子女
     else if(t[i].parent>0) bt[t[i].parent].rc=i;  //右子女
      else  *root=i;   //root 记住根结点
  } }//change
void PreOrder(Tree2 bt)          //对二叉树进行前序遍历
 {int *root,top=0;  int s[];  //s 是栈
  change(t,bt,root); int i=*root;
  while(i!=0||top>0)
    {while (i!=0)
       {printf (bt[i].data); if(bt[i].rc!=0) s[++top]=bt[i].rc; //右子女进栈
        i=bt[i].lc;
        }
     if (top>0) i=s[top--];
    } }//结束 preorder
  [算法讨论]本题的前序递归算法如下
void PreOrder(int root)//root 是二叉树根结点在顺序存储中的下标，本算法前序遍历二叉树 bt
 {if(root!=0){printf(bt[root].data);//访问根结点
              PreOrder(bt[root].lc);//前序遍历左子树
              PreOrder(bt[root].rc);//前序遍历右子树
  } }//结束 preorder，初始调用时，root 是根结点的下标
```

这类问题的求解方法值得注意。当给定数据存储结构不合适时，可由已给结构来构造好的数据结构，以便于运算。象上面第 5 题也是这样，先根据 L 和 R 数组，构造一个结点的双亲的数组 T。

22.[题目分析]二叉树先序序列的最后一个结点，若二叉树有右子树，则是右子树中最右下的叶子结点；若无右子树，仅有左子树，则是左子树最右下的叶子结点；若二叉树无左右子树，则返回根结点。

```
BiTree LastNode(BiTree bt)//返回二叉树 bt 先序序列的最后一个结点的指针
 {BiTree p=bt;
  if(bt==null) return(null);
  else while(p)
        if (p->rchild) p=p->rchild;       //若右子树不空，沿右子树向下
        else if (p->lchild) p=p->lchild; //右子树空，左子树不空，沿左子树向下
             else return(p);              //p 即为所求
  }//结束 lastnode
```

23.[题目分析]高度为 K 的二叉树，按顺序方式存储，要占用 2^K-1 个存储单元，与实际结点个数 n 关系不大，对不是完全二叉树的二叉树，要增加“虚结点”，使其在形态上成为完全二叉树。

```
int m=2^K - 1;  //全局变量
void PreOrder(ElemType bt[],i )
//递归遍历以顺序方式存储的二叉树 bt， i 是根结点下标（初始调用时为 1)。
  {if (i<=m)        //设虚结点以 0 表示
     {printf(bt[i]);   //访问根结点
      if(2*i<=m && bt[2*i]!=0) PreOrder(bt,2*i);       //先序遍历左子树
      if(2*i+1<=m && bt[2*i+1]!=0) PreOrder(bt,2*i+1);// 先序遍历右子树
  }  }//结束 PreOrder
```

二叉树中最大序号的叶子结点,是在顺序存储方式下编号最大的结点

```
void Ancesstor(ElemType bt[])     //打印最大序号叶子结点的全部祖先
{c=m; while(bt[c]==0) c--;        //找最大序号叶子结点,该结点存储时在最后
 f=c/2;        //c 的双亲结点 f
 while(f!=0)  //从结点 c 的双亲结点直到根结点，路径上所有结点均为祖先结点
   {printf(bt[f]); f=f/2; }//逆序输出，最老的祖先最后输出
 } //结束
```

24.

```
void InOrder(BiTree bt)
 {BiTree s[],p=bt;  //s 是元素为二叉树结点指针的栈，容量足够大
  int top=0;
  while(p || top>0)
     {while(p) {s[++top]=p; bt=p->lchild;}                  //中序遍历左子树
      if(top>0){p=s[top--]; printf(p->data); p=p->rchild;} //退栈，访问，转右子树
     }   }
```

25.[题目分析] 二叉树用顺序方式存储，其遍历方法与用二叉链表方式存储类似。判空时，在二叉链表方式下用结点指针是否等于 null，在顺序存储方式下， 一是下标是否是“虚结点”，二是下标值是否超过最大值（高为 H 的二叉树要有 2^H-1 个存储单元，与实际结点个数关系不大)。当然，顺序存储方式下，要告诉根结点的下标。

```
void InOrder(int i) //对顺序存储的二叉树进行中序遍历，i 是根结点的下标
{if(i!=0)
  {InOrder(ar[i].Lc);   //中序遍历左子树
   printf(ar[i].data);  //访问根结点
   InOrder(ar[i].Rc);   //中序遍历左子树
  }  } // 结束 InOrder
```

26.[题目分析] 借助队列和栈，最后弹出栈中元素实现对二叉树按自下至上，自右至左的层次遍历

```
void InvertLevel(biTree bt)  // 对二叉树按自下至上，自右至左的进行层次遍历
 {if(bt!=null)
   {StackInit(s); //栈初始化，栈中存放二叉树结点的指针
    QueueInit(Q); //队列初始化。队列中存放二叉树结点的指针
    QueueIn(Q,bt);
    while(!QueueEmpty(Q))                    //从上而下层次遍历
```

```
        {p=QueueOut(Q); push(s,p);              //出队, 入栈
         if(p->lchild) QueueIn(Q,p->lchild);  //若左子女不空，则入队列
         if(p->rchild) QueueIn(Q,p->rchild);} //若右子女不空,则入队列
      while(!StackEmpty(s)) {p=pop(s); printf(p->data);} //自下而上,从右到左的层次遍历
     }//if(bt!=null)
    } //结束 InvertLevel
27. int Level(BiTree bt) //层次遍历二叉树，并统计度为 1 的结点的个数
    {int num=0; //num 统计度为 1 的结点的个数
     if(bt){QueueInit(Q); QueueIn(Q,bt); //Q 是以二叉树结点指针为元素的队列
            while(!QueueEmpty(Q))
             {p=QueueOut(Q); printf(p->data);      //出队,访问结点
              if(p->lchild && !p->rchild ||!p->lchild && p->rchild)num++;//度为 1 的结点
              if(p->lchild) QueueIn(Q,p->lchild); //非空左子女入队
              if(p->rchild) QueueIn(Q,p->rchild); //非空右子女入队
             }  }//if(bt)
     return(num); }//返回度为 1 的结点的个数
28. void exchange(BiTree bt)//将二叉树 bt 所有结点的左右子树交换
    {if(bt){BiTree s;
            s=bt->lchild; bt->lchild=bt->rchild; bt->rchild=s; //左右子女交换
            exchange(bt->lchild);  //交换左子树上所有结点的左右子树
            exchange(bt->rchild);  //交换右子树上所有结点的左右子树
           } }
```

[算法讨论]将上述算法中两个递归调用语句放在前面，将交换语句放在最后，则是以后序遍历方式交换所有结点的左右子树。中序遍历不适合本题。下面是本题（1）要求的非递归算法

```
（1）void exchange(BiTree  t)   //交换二叉树中各结点的左右孩子的非递归算法
    {int top=0; BiTree s[],p;   //s 是二叉树的结点指针的栈，容量足够大
     if(bt)
     {s[++top]=t;
      while(top>0)
       {t=s[top--];
        if(t->lchild||t->rchild){p=t->lchild;t->lchild=t->rchild;t->rchild=p;}
//交换左右
        if(t->lchild) s[++top]=t->lchild; //左子女入栈
        if(t->rchild) s[++top]=t->rchild; //右子女入栈
       }//while(top>0)
     }//if(bt)   }// 结束 exchange
```

29.[题目分析]对一般二叉树，仅根据一个先序、中序、后序遍历，不能确定另一个遍历序列。但对于满二叉树，任一结点的左右子树均含有数量相等的结点，根据此性质，可将任一遍历序列转为另一遍历序列（即任一遍历序列均可确定一棵二叉树）。

```
    void PreToPost(ElemType pre[] ,post[],int l1,h1,l2,h2)
      //将满二叉树的先序序列转为后序序列，l1,h1,l2,h2 是序列初始和最后结点的下
```

标。

```
{if(h1>=l1)
  {post[h2]=pre[l1];     //根结点
   half=(h1-l1)/2;       //左或右子树的结点数
    PreToPost(pre,post,l1+1,l1+half,l2,l2+half-1)  //将左子树先序序列转为后序序列
    PreToPost(pre,post,l1+half+1,h1,l2+half,h2-1)  //将右子树先序序列转为后序序列
  }  }//PreToPost
```

30. BiTree IntoPost(ElemType in[],post[],int l1,h1,l2,h2)

//in和post是二叉树的中序序列和后序序列,l1,h1,l2,h2分别是两序列第一和最后结点的下标

```
{BiTree bt=(BiTree)malloc(sizeof(BiNode));//申请结点
 bt->data=post[h2];//后序遍历序列最后一个元素是根结点数据
 for(i=l1;i<=h1;i++) if(in[i]==post[h2])break;//在中序序列中查找根结点
 if(i==l1) bt->lchild=null;  //处理左子树
    else bt->lchild=IntoPost(in,post,l1,i-1,l2,l2+i-l1-1);
 if(i==h1) bt->rchild=null;  //处理右子树
    else bt->rchild=IntoPost(in,post,i+1,h1,l2+i-l1,h2-1);
return(bt);  }
```

31. TYPE bitreptr=^binode;

```
      binode=RECORD data:ElemType; lchild,rchlid:bitreptr END;
PROC PreOrder(bt:bitreptr);  //非递归前序遍列二叉树
VAR S:ARRAY[1..max] OF bitreptr;    //max 是栈容量，足够大
inits(S);//栈初始化
WHILE (bt<>NIL) OR (NOT empty(S)) DO
[WHILE (bt<>NIL )DO
 [write(bt↑data); push(S,bt->rchild); bt:=bt↑.lchild;]//访问结点,右子女进栈
 WHILE (NOT empty(S) AND top(S)=NIL) bt:=pop(S);// 退栈
 IF NOT empty(S) THEN bt:=pop(S);
 ] ENDP;
```

[算法讨论]若不要求使用 top(S)，以上算法还可简化。

32. [题目分析]二叉树采取顺序结构存储，是按完全二叉树格式存储的。对非完全二叉树要补上“虚结点”。由于不是完全二叉树，在顺序结构存储中对叶子结点的判定是根据其左右子女为0。叶子和双亲结点下标间的关系满足完全二叉树的性质。

```
int Leaves(int h)  //求深度为h以顺序结构存储的二叉树的叶子结点数
{int BT[]; int len=2^h-1, count=0; //BT是二叉树结点值一维数组，容量为2^h
 for (i=1;i<=len;i++) //数组元素从下标1开始存放
   if (BT[i]!=0)      //假定二叉树结点值是整数，“虚结点”用0填充
     if(i*2)>len) count++;                    //第i个结点没子女，肯定是叶子
      else if(BT[2*i]==0 && 2*i+1<=len && BT[2*i+1]==0) count++; //无左右子女的结点是叶子
```

```
 return (count)
} //结束 Leaves
```

33.[题目分析] 计算每层中结点值大于 50 的结点个数，应按层次遍历。设一队列 Q，用 front 和 rear 分别指向队头和队尾元素，last 指向各层最右结点的位置。存放值大于 50 的结点结构为

```
typedef struct {int level,value,idx; }node;//元素所在层号、值和本层中的序号
node a[],s;
void ValueGT50(BiTree bt)//查各层中结点值大于 50 的结点个数，输出其值及序号
 {if(bt!=null)
   {int front=0,last=1,rear=1,level=1,i=0,num=0;//num 记>50 的结点个数
    BiTree Q[];Q[1]=bt;//根结点入队
    while(front<=last)
     {bt=Q[++front];
      if(bt->data>50){s.level=level;      s.idx=++i;      s.value=bt->data;
a[++num]=s;}
      if(bt->lchild!=null) Q[++rear]=bt->lchild;//左子女入队列
      if(bt->rchild!=null) Q[++rear]=bt->rchild;//右子女入队列
      if(front==last) {last=rear; level++; i=0;} //本层最后一个结点已处理完
     } //初始化下层：last 指向下层最右结点，层号加 1，下层>50 的序号初始为 0
     }//while
   for(i=1;i<=num;i++) //输出 data 域数值大于 50 的结点的层号、data 域的数值和
序号
   printf("层号=%3d,本层序号=%3d，值=%3d", a[i].level,a[i].idx,a[i].value);
}//算法 ValueGT50 结束
```

34.

```
PROC print(bt:BiTree; xy:integer)
 //将二叉树逆时针旋转 90 度打印，xy 是根结点基准坐标，调用时 xy=40
IF bt<>NIL THEN [ print(bt↑.rchild,xy+5);      //中序遍历右子树
                  writeln(bt->data:xy);        //访问根结点
                  print(bt↑.lchild,xy+5);]     //中序遍历左子树
 ENDP;
```

35.

```
BiTree creat()//生成并中序输出二叉排序树
{scanf("%c",&ch) //ch 是二叉排序树结点值的类型变量，假定是字符变量
 BiTree bst=null,f=null;
 while(ch!='#')                    //'#'是输入结束标记
  {s=(BiTree)malloc(sizeof(BiNode)); //申请结点
   s->data=ch; s->lchild=s->rchild=null;
   if (bst==null) bst=s;            //根结点
   else                             //查找插入结点
    {p=bst;
     while(p)
      if (ch>p->data) {f=p; p=p->rchild;}          //沿右分枝查,f 是双亲
      else {f=p; p=p->lchild;}                     //沿左分枝查
    if(f->data<ch) f->rchild=s; else f->lchild=s;}//将 s 结点插入树中
   scanf("%c",&ch);  //读入下一数据
```

```
    } //while (ch!= '#' )
  return(bst); } //结束 creat
 void InOrder(BiTree bst)      //bst 是二叉排序树，中序遍历输出二叉排序树
  {if(bst)
     {InOrder (bst->lchild); printf(bst->data); InOrder(bst->rchild); }
  }//结束 InOrder
```

36.[题目分析]二叉树结点 p 所对应子树的第一个后序遍历结点 q 的求法如下：若 p 有左子树，则 q 是 p 的左子树上最左下的叶子结点；若 p 无左子树，仅有右子树，则 q 是 p 的右子树上最左下的叶子结点。

```
    BiTree PostFirst(p)
     {BiTree q=p;
      if (!q) return(null); else
       while(q->lchild || q->rchild); //找最左下的叶子结点
         if(q->lchild) q=q->lchild;   //优先沿左分枝向下去查“最左下”的叶子结点
           else q=q->rchild;          //沿右分枝去查“最左下”的叶子结点
      return(q);
     }
```

[算法讨论]题目“求 p 所对应子树的第一个后序遍历结点”，蕴涵 p 是子树的根。若 p 是叶子结点，求其后继要通过双亲。

37.(1)由先序序列 A[1..n]和中序序列 B[1..n],可以确定一棵二叉树,详见本章四第 38 题.

（2）

```
    //由二叉树前序序列A[1..n]和中序序列B[1..n]建立二叉树,l1,h1,和l2,h2分别为先序序列和
    //中序序列第一和最后结点的下标.初始调用时 l1=l2=1,h1=h2=n 。
    {typedef struct {int l1,h1,l2,h2; BiTree t; }node;
     BiTree bt;
     int top=0,i; node s[],p;   //s 为栈,容量足够大
     bt=(BiTree)malloc(sizeof(BiNode)); //申请结点空间
     p.l1=l1; p.h1=h1; p.l2=l2; p.h2=h2; p.t=bt; s[++top]=p; //初始化
     while(top>0)
       {p=s[top--]; bt=p.t; l1=p.l1; h1=p.h1; l2=p.l2 ;h2=p.h2;//取出栈顶数据
        for(i=l2;i<=h2;i++) if(B[i]==A[l1]) break;    //到中序序列中查根结点的值
        bt->data=A[l1];                    //A[l1]为根结点的值
        if(i==l2) bt->lchild=null;         //bt 无左子树
        else   //将建立左子树的数据入栈
          {bt->lchild=(BiTree)malloc(sizeof(BiNode)); p.t=bt->lchild;
           p.l1=l1+1; p.h1=l1+i-l2; p.l2=l2; p.h2=i-1; s[++top]=p;  }
        if(i==h2) bt->rchild=null; //bt 无右子树
        else {bt->rchild=(BiTree)malloc(sizeof(BiNode)); p.t=bt->rchild;
              p.l1=l1+i-l2+1; p.h1=h1; p.l2=i+1; p.h2=h2; s[++top]=p; }//右子树数据入栈
       }//while
    }结束 PreInCreat
```

(3)当二叉树为单支树时，栈深 n;，当二叉树左右子树高相等时，栈深 logn。时间复杂度 O(n)。

38.[题目分析]知二叉树中序序列与后序序列，第 30 题以递归算法建立了二叉树，本题是非递归算法。

```
void InPostCreat(ElemType IN[],POST[],int l1,h1,l2,h2)
//由二叉树的中序序列 IN[]和后序序列 POST[]建立二叉树，l1,h1 和 l2,h2 分别是中序序列和
//后序序列第一和最后元素的下标，初始调用时，l1=l2=1,h1=h2=n。
{typedef struct {int l1,h1,l2,h2; BiTree t; }node;
node s[],p;//s 为栈，容量足够大
BiTree bt=(BiTree)malloc(sizeof(BiNode)); int top=0,i;
p.l1=l1; p.h1=h1; p.l2=l2; p.h2=h2; p.t=bt; s[++top]=p;//初始化
while(top>0)
 {p=s[top--]; bt=p.t; l1=p.l1; h1=p.h1; l2=p.l2; h2=p.h2;//取出栈顶数据
  for(i=l1;i<=h1;i++) if(IN[i]==POST[h2]) break;//在中序序列中查等于 POST[h2]的结点
  bt->data=POST[h2];      //根结点的值
  if(i==l1) bt->lchild=null;     //bt 无左子树
  else    //将建立左子树的数据入栈
    {bt->lchild=(BiTree)malloc(sizeof(BiNode)); p.t=bt->lchild;
     p.l1=l1; p.h1=i-1; p.l2=l2; p.h2=l2+i-l1-1; s[++top]=p; }
  if(i==h1) bt->rchild=null;     //bt 无右子树
    else {bt->rchild=(BiTree)malloc(sizeof(BiNode)); p.t=bt->rchild;
          p.l1=i+1; p.h1=h1; p.l2=l2+i-l1; p.h2=h2-1; s[++top]=p;  }//右子树数据入栈
  }// while(top>0)
}结束 InPostCreat
```

39. BiTree Copy(BiTree t)//复制二叉树 t

```
{BiTree bt;
 if (t==null) bt=null;
 else{bt=(BiTree)malloc(sizeof(BiNode)); bt->data=t->data;
      bt->lchild=Copy(t->lchild);
      bt->rchild=Copy(t->rchild);
      }
 return(bt); }//结束 Copy
```

40.[题目分析]森林在先根次序遍历时，首先遍历第一棵子树的根，接着是第一棵子树的结点；之后是第二棵树，……，最后一棵树。本题中 E[i]是 H[i]所指结点的次数，次数就是结点的分支个数 B，而分支数 B 与树的结点数 N 的关系是 N=B+1（除根结点外，任何一个结点都有一个分支所指）。所以，从 E[i]的第一个单元开始，将值累加，当累加到第 i 个单元，其值正好等于 i-1 时，就是第一棵树。接着，用相同方法，将其它树分开，进行到第 n 个单元，将所有树分开。例如，上面应用题第 47 题(2)的森林可按本题图示如下。从左往右将次数加到下标 8（=B+1）时，正好结束第一棵树。

i	1	2	3	4	5	6	7	8	9	10	11	12
H[i]	A	B	D	G	L	H	E	I	C	F	J	K
E[i]	3	2	2	0	0	0	0	0	0	1	0	0

```
void Forest(ElemType H[],int E[],int,n)
  // H[i]是森林 F 在先根次序下结点的地址排列，E[i]是 H[i]所指结点的次数，本算法计算森林
  //F 的树形个数，并计算森林 F 的最后一个树形的根结点地址
  {int i=1,sum=0,j=0,m=0; //sum 记一棵树的分支数，j 记树的棵数，m 记一棵树的结点数
    int tree[];   //tree 记每棵树先序遍历最后一个结点在 H[i]中的地址
    while (i<=n)  //n 是森林中结点个数，题目已给出
     {sum+=E[i]; m++;
      if (sum+1==m && i<=n) //记树先序最后结点的地址，为下步初始化
         {sum=0; m=0; tree[++j]=i;}
      i++;
      }//while
     if (j==1)return (1);     //只有一棵树时，第一个结点是根
       else return(tree[j-1]+1)
  }//forest
```

41.[题目分析]叶子结点只有在遍历中才能知道，这里使用中序递归遍历。设置前驱结点指针 pre，初始为空。第一个叶子结点由指针 head 指向，遍历到叶子结点时，就将它前驱的 rchild 指针指向它，最后叶子结点的 rchild 为空。

```
LinkedList head,pre=null;  //全局变量
LinkedList InOrder(BiTree bt)
  //中序遍历二叉树 bt，将叶子结点从左到右链成一个单链表，表头指针为 head
  {if(bt){InOrder(bt->lchild);           //中序遍历左子树
          if(bt->lchild==null && bt->rchild==null) //叶子结点
             if(pre==null) {head=bt; pre=bt;} //处理第一个叶子结点
             else{pre->rchild=bt; pre=bt; }   //将叶子结点链入链表
          InOrder(bt->rchild);                 //中序遍历左子树
          pre->rchild=null;                    //设置链表尾
         }
    return(head);  } //InOrder
```

时间复杂度为 O(n),辅助变量使用 head 和 pre,栈空间复杂度 O(n)

42. 层次遍历，参见上面算法第 33 题。中序遍历序列和后序序列可确定一棵二叉树，详见上面应用题第 40 题。先序序列和后序序列无法确定一棵二叉树，因为无法将二叉树分成左右子树。如先序序列 AB 和后序序列 BA，A 是根，但 B 可以是左子女，也可以是右子女。

43.[题目分析]此树看作度为 2 的有序树，先将根结点入队列。当队列不空，重复以下动作：结点出队；若结点有两个子女，先将第二（右）子女入队列，并转向第一子女；若结点有一个子女，则入队列；如此下去，直到碰到叶子结点或只有一个子女的结点，再重复上述动作，直至队列为空。定义树结构如下：

```
typedef struct node
  {ElemType data; struct node *firstchild, *secondchild; }*Tree;
```

```
void TreeTravers(Tree t)  //按字母顺序输出树中各结点的值
{Tree p,Q[];              //Q 为队列，元素是树结点指针，容量是够大
 if (t)
   {QueueInit(Q); QueueIn(Q,t); //根结点入队
    while(!QueueEmpty(Q))
     {p=QueueOut(Q);           //出队
      while (p->firstchild && p->secondchild)      //当有双子女时
        {QueueIn(Q,p->secondchild);printf(p->data);//访问结点
         p=p->firstchild;}// 沿第一子女向下
      if (p->firstchild){printf(p->data); QueueIn(Q,p->firstchild)} //一个子女树入队
      if (!p->firstchild && !p->secondchild) printf(p->data);//访问叶子结点
      }//while(!QueueEmpty(Q))
     }//if
}//算法结束
```

44.[题目分析] 由定义，结点的平衡因子 bf 等于结点的左子树高度与右子树高度之差，设计一遍历算法，在遍历结点时，求结点的左子树和右子树的高度，然后得到结点的平衡因子。

```
int Height(BiTree bt)//求二叉树 bt 的深度
  {int hl,hr;
   if (bt==null) return(0);
   else {hl=Height(bt->lchild); hr=Height(bt->rchild);
         if(hl>hr) return (hl+1); else return(hr+1);
     }  }// Height
void  Balance(BiTree bt)
  //计算二叉树 bt 各结点的平衡因子
{if (bt)
 {Balance(bt->lchild); //后序遍历左子树
  Balance(bt->rchild); //后序遍历右子树
  hl=Height(bt->lchild); hr=Height(bt->rchild);//求左右子树的高度
  bt->bf=hl-hr; //结点的平衡因子 bf
  }   }//算法结束
```

45.[题目分析]本题应采用层次遍历方式。若树不空，则二叉树根结点入队，然后当队列不空且队列长不超过 n，重复如下操作：出队，若出队元素不为空，则记住其下标，且将其左右子女入队列；若出队元素为空，当作虚结点，也将其“虚子女”入队列。为节省空间，就用树 T 的顺序存储结构 A[1..n]作队列,队头指针 front,队尾指针 rear,元素最大下标 last.

```
void  Traverse(BiTree bt ,int n)
  // 求二叉树 bt 的顺序存储结构 A[1..n]， 下标超过 n 报错，给出实际的最大下标
 {BiTree A[], p;
  if(bt!=null)
    {int front=0,rear=1,last=1; A[1]=bt;
     while(front<=rear)
       {p=A[++front]; if(p) last=front; // 出队;用 last 记住最后一个结点的下标
        rear=2*front;//计算结点（包括虚结点）“左子女”下标
```

```
            if (p)        //二叉树的实际结点
             {if(rear>n)    printf(“%c 结点无左子女”); else A[rear]=p->lchild;
              if(rear+1>n) printf(“%c 结点无右子女”); else A[rear+1]=p->rchild;
             }
            else //p 是虚结点
               { if(rear<=n) A[rear]=null; if(rear+1<=n) A[rear+1]=null; }
          }// while(front<=rear)
         printf(“实际的最大下标是%d”,last);
       }//if(bt!=null)   }//Traverse
```

46.[题目分析]两棵空二叉树或仅有根结点的二叉树相似；对非空二叉树，可判左右子树是否相似，采用递归算法。

```
int Similar(BiTree p,q) //判断二叉树 p 和 q 是否相似
   {if(p==null && q==null) return (1);
    else if(!p && q || p && !q) return (0);
         else          return(Similar(p->lchild,q->lchild)          &&
Similar(p->rchild,q->rchild))
   }//结束 Similar
```

47.[题目分析] 根据树的双亲表示法创建树的孩子兄弟链表表示法,首先给出根结点在双亲表示法中的下标，但找根结点的子女要遍历双亲表示法的整个静态链表，根结点的第一个子女是孩子兄弟表示法中的孩子，其它子女结点作兄弟。对双亲表示法中的任一结点，均递归建立其孩子兄弟链表子树。

```
 CSTree PtreeToCstree (PTree pt,int root)
  //本算法将双亲表示法的树 pt 转为孩子兄弟链表表示的树，root 是根结点在双亲表示法中的下标。
  {CSTree child , sibling; int firstchild;
   CSTree cst=(CSTree)malloc(sizeof(CSNode)); //申请结点空间
   cst->data=pt.nodes[root].data; cst->firstchild=null; cst->nextsibling=null;//根结点
   firstchild=1;
   for(i=1;i<=pt.n;i++)  //查找 root 的孩子
     if(pt.nodes[i].parent==root)
       {child=PtreetoCstree(pt,i);
        if(firstchild)                                  {cst->firstchild=child;
firstchild=0;sibling=cst->firstchild;}
        else //child 不是 root 的第一个孩子，作兄弟处理
         {sibling->nextsibling=child; sibling=sibling->nextsibling;}
        }//if
    }//end for
  return cst;  }//结束 PtreetoCstree
```

48.[题目分析] 删除以元素值 x 为根的子树，只要能删除其左右子树，就可以释放值为 x 的根结点，因此宜采用后序遍历。删除值为 x 结点，意味着应将其父结点的左(右)子女指针置空，用层次遍历易于找到某结点的父结点。本题要求删除树中每一个元素值为 x 的结点的子树，因此要遍历完整棵二叉树。

```
   void  DeleteXTree(BiTree  bt)  //删除以 bt 为根的子树
```

```
{DeleteXTree(bt->lchild); DeleteXTree(bt->rchild);//删除 bt 的左子树、右子树
 free(bt);  }// DeleteXTree                          //释放被删结点所占的存储空间
void  Search(B:Tree bt,ElemType x)
  //在二叉树上查找所有以 x 为元素值的结点，并删除以其为根的子树
 {BiTree Q[];//Q 是存放二叉树结点指针的队列，容量足够大
  if(bt)
   {if(bt->data==x) {DeleteXTree(bt); exit(0);}//若根结点的值为 x，则删除整棵树
    {QueueInit(Q); QueueIn(Q,bt);
     while(!QueueEmpty(Q))
       {p=QueueOut(Q);
        if(p->lchild)                  // 若左子女非空
          if(p->lchild->data==x)     //左子女结点值为 x,应删除当前结点的左子树
            {DeleteXTree(p->lchild); p->lchild=null;} //父结点的左子女置空
          else Enqueue (Q,p->lchild);// 左子女入队列
        if(p->rchild)                  // 若右子女非空
          if(p->rchild->data==x)     //右子女结点值为 x,应删除当前结点的右子树
            {DeleteXTree(p->rchild);   p->rchild=null;} //父结点的右子女置空
          else Enqueue (Q,p->rchild);// 右子女入队列
       }//while
    }//if(bt)  }//search
```

49.[题目分析]在二叉树上建立三叉链表，若二叉树已用二叉链表表示，则可象 48 题那样，给每个结点加上指向双亲的指针（根结点的双亲指针为空）。至于删除元素值为 x 的结点以及以 x 为根的子树，与 48 题完全一样，请参照 48 题。下面给出建立用三叉链表表示的二叉树的算法。二叉树按完全二叉树格式输入，对非完全二叉树，要补上“虚结点”。按完全二叉树双亲和子女存储下标间的关系，完成双亲和子女间指针的链接。‘#’是输入结束标志，‘$’是虚结点标志。

```
BiTree creat()//生成三叉链表的二叉树（题目给出 PASCAL 定义，下面的用类 C 书写）
{BiTree p,Q[], root;  //Q 是二叉树结点指针的一维数组，容量足够大
 char ch; int rear=0; //一维数组最后元素的下标
 scanf(&ch);
 while(ch!=‘#’)
 {p=null;
  if(ch!=‘$’){p=(BiTree)malloc(sizeof(nodetp));
              p->data=ch; p->lchild=p->rchild=null; }
  Q[++rear]=p;  //元素或虚结点
  if(p){if(rear==1) {root=p;root->parent=null; } //根结点
        else{Q[rear]->parent=Q[rear/2]; /双亲结点和子女结点用指针链上
             if (rear%2==0) Q[rear/2]->lchild=Q[rear]; else Q[rear/2]->rchild=Q[rear];
            }
```

```
    scanf(“%c”,&ch);
   }//while
  return(root); }//结束 creat
50.int BTLC(BiTree T,int *c)//对二叉树 T 的结点计数
   {if(T)
      {*c++;
       BTLC(T->lchild,&c); //统计左子树结点
       BTLC(T->rchild,&c); //统计右子树结点
       } }//结束 Count,调用时*c=0
51. int Count(CSTree t)//统计以孩子兄弟链表表示的树的叶子结点个数
    {if(t==null) return(0);
     else if(t->firstlchild==null)  //左子女为空，结点必为叶子
             return(1+Count(t->nextsibling));  //（叶子）+兄弟子树上的叶子结点
          else return(Count(t->firstchild)+Count(t->nextsibling));//子女子树+兄弟子树
     }//Count
52.void Count(BiTree bt,int *n0,*n) //统计二叉树 bt 上叶子结点数 n0 和非叶子结点数 n
   {if(bt)
      {if (bt->lchild==null && bt->rchild==null) *n0++; //叶子结点
       else  *n++; //非叶结点
       Count(bt->lchild,&n0,&n);
       Count(bt->rchild,&n0,&n);
      }   }//结束 Count
53. int Count (BiTree  bt) // 非递归遍历求二叉树上的叶子结点个数
    {int num=0;
     BiTree s[]; //s 是栈，栈中元素是二叉树结点指针，栈容量足够大
     whlie(bt!=null || top>0)
       {while(bt!=null) {push(s,bt); bt=bt->lchild; }  //沿左分支向下
        if(!StackEmpty(s))
         {bt=pop(s); if(bt->lchild==null && bt->rchild==null) num++; //叶子结点
          bt=bt->rchild;
         }
        } return(num);
    }//结束 Count
54.[题目分析]对二叉树的某层上的结点进行运算，采用队列结构按层次遍历最适宜。
   int LeafKlevel(BiTree bt, int k) //求二叉树 bt 的第 k(k>1) 层上叶子结点个数
    {if(bt==null || k<1) return(0);
     BiTree p=bt,Q[];              //Q 是队列，元素是二叉树结点指针,容量足够大
     int front=0,rear=1,leaf=0;  //front 和 rear 是队头和队尾指针, leaf 是叶子结点数
     int last=1,level=1; Q[1]=p; //last 是二叉树同层最右结点的指针，level 是二叉树的层数
     while(front<=rear)
```

```
{p=Q[++front];
  if(level==k && !p->lchild && !p->rchild) leaf++; //叶子结点
 if(p->lchild) Q[++rear]=p->lchild;                //左子女入队
  if(p->rchild) Q[++rear]=p->rchild;               //右子女入队
  if(front==last) {level++;       //二叉树同层最右结点已处理,层数增 1
                   last=rear; }   //last 移到指向下层最右一元素
if(level>k) return (leaf);              //层数大于 k 后退出运行
  }//while  }//结束 LeafKLevel
```

55.[题目分析]按题目要求，每个结点的编号大于其左右孩子的编号，结点左孩子的编号小于右孩子的编号。由此看出，从小到大按“左右根”顺序，这正是后序遍序的顺序，故对二叉树进行后序遍历，在遍历中对结点进行编号，现将二叉树结点结构定义如下：

```
typedef struct node
{ElemType data; int num; struct node *lchild, *rchild; }Bnode,*Btree;
void PostOrder(Btree t)
//对二叉树从 1 开始编号，结点编号大于其左右子女结点编号，结点的左子女编号小于其右子女编号
{typedef struct {Btree t; int tag;  }node;
 Btree p=t; node sn,s[];    //s 为栈，容量足够大
 int  k=0,top=0; //k 为结点编号，top 为栈顶指针
 while(p!=null || top>0)
  {while(p) {sn.t=p; sn.tag=0; s[++top]=sn; p=p->lchild;} //沿左分枝向下
   while(top>0 && s[top].tag==1){sn=s[top--];sn.t->num=++k;}//左右孩子已遍历,结点赋编号
   if (top>0) {s[top].tag=1; p=s[top].t->rchild;}
  }//while(p!=null || top>0)
}结束 PostOrder
```

56. [题目分析]非递归算法参考上面第 37 题。下面给出递归算法。

```
void PreInCreat(BiTree root,ElemType pre[],in[],int l1,h1,l2,h2)
 //根据二叉树前序序列 pre 和中序序列 in 建立二叉树。l1,h1,l2,h2 是序列第一和最后元素下标。
{root=(BiTree)malloc(sizeof(BiNode)); //申请结点
 root->data=pre[l1];  //pre[l1]是根
 for(i=l2;i<=h2;i++) if(in[i]==pre[l1]) break; //在中序序列中，根结点将树分成左右子树
 if(i==l2) root->lchild=null; //无左子树
 else PreInCreat(root->lchild,pre,in,l1+1,l1+(i-l2),l2,i-1); //递归建立左子树
 if(i==h2) root->rchild=null;  //无右子树
 else PreInCreat((root)->rchild,pre,in,l1+(i-l2)+1,h1,i+1,h2) //递归建立右子树
}//结束 PreInCreat
```

57（1）略 (2) 根据中序和后序序列，建立二叉树的递归算法见上面第 30 题，非递归算法见第 38 题。

58.[题目分析]采用后序非递归遍历二叉树，栈中保留从根结点到当前结点的路径上的所有

结点。

```
void PrintPath(BiTree bt,p) //打印从根结点 bt 到结点 p 之间路径上的所有结点
 {BiTree q=bt,s[]; //s 是元素为二叉树结点指针的栈，容量足够大
  int top=0; tag[];//tag 是数组，元素值为 0 或 1，访问左、右子树的标志，tag 和 s 同步
  if (q==p) {printf(q->data); return;} //根结点就是所找结点
  while(q!=null || top>0)
   {while(q!=null)  //左子女入栈，并置标记
      if（q==p）    //找到结点 p，栈中元素均为结点 p 的祖先
        {printf(“从根结点到 p 结点的路径为\n”);
         for(i=1;i<=top;i++) printf(s[i]->data); printf(q->data); return;
         }
      else {s[++top]=q; tag[top]=0; q=q—>lchild;} //沿左分支向下
    while(top>0 && tag[top]==1)) top--; //本题不要求输出遍历序列，这里只退栈
    if (top>0) {q=s[top]; q=q->rchild; tag[top]=1; } //沿右分支向下
   }//while(q!=null || top>0)
 }//结束算法 PrintPath
```

59.[题目分析]上题是打印从根结点到某结点 p 的路径上所有祖先结点，本题是打印由根结点到叶子结点的所有路径。只要在上题基础上把 q 是否等于 p 的判断改为 q 是否是叶子结点即可。其语句段如下：

```
if(q->lchild==null&&q->rchild==null) //q 为叶子结点
  {printf(“从根结点到 p 结点的路径为\n”);
   for(i=1;i<=top;i++) printf(s[i]->data);//输出从根到叶子路径上，叶子 q 的所有祖先
    printf(q->data); }
```

60.[题目分析]因为后序遍历栈中保留当前结点的祖先的信息，用一变量保存栈的最高栈顶指针，每当退栈时，栈顶指针高于保存最高栈顶指针的值时，则将该栈倒入辅助栈中，辅助栈始终保存最长路径长度上的结点，直至后序遍历完毕，则辅助栈中内容即为所求。

```
void LongestPath(BiTree bt)//求二叉树中的第一条最长路径长度
 {BiTree p=bt,l[],s[]; //l, s 是栈，元素是二叉树结点指针，l 中保留当前最长路径中的结点
  int i，top=0,tag[],longest=0;
  while(p || top>0)
   { while(p) {s[++top]=p; tag[top]=0; p=p->Lc;} //沿左分枝向下
     if(tag[top]==1)    //当前结点的右分枝已遍历
      {if(!s[top]->Lc && !s[top]->Rc)  //只有到叶子结点时，才查看路径长度
       if(top>longest) {for(i=1;i<=top;i++) l[i]=s[i]; longest=top; top--;}
            //保留当前最长路径到 l 栈，记住最高栈顶指针，退栈
       }
     else if(top>0) {tag[top]=1; p=s[top].Rc;}   //沿右子分枝向下
   }//while(p!=null||top>0)
 }//结束 LongestPath
```

61.[题目分析]在线索二叉树上插入结点,破坏了与被插入结点的线索。因此,插入结点时,必须修复线索。在结点 y 的右侧插入结点 x，因为是后序线索树，要区分结点 y 有无左子树

的情况。

```
void TreeInsert(BiTree t,y,x)//在二叉树 t 的结点 y 的右侧，插入结点 x
 {if(y->ltag==0)  //y 有左子女
   {p=y->lchild;  if (p->rtag==1) p->rchild=x; //x 是 y 的左子女的后序后继
    x->ltag=1; x->lchild=p;          //x 的左线索是 y 的左子女
   }
  else  //y 无左子女
   {x->ltag=1; x->lchild=y->lchild;//y 的左线索成为 x 的左线索
    if(y->lchild->rtag==1)          //若 y 的后序前驱的右标记为 1
       y->lchild->rchild=x;         //则将 y 的后序前驱的后继改为 x
   }
  x->rtag=1; x->rchild=y; y->rtag=0; y->rchild=x; //x 作 y 的右子树
 }//结束  TreeInsert
```

62.[题目分析]在中序全线索化 T 树的 P 结点上，插入以 X 为根的中序全线索化二叉树，要对 X 有无左右子树进行讨论，并修改 X 左(右)子树上最左结点和最右结点的线索。在中序线索树上查找某结点 p 的前驱的规律是：若 p->ltag=1，则 p->lchild 就指向前驱，否则，其前驱是 p 的左子树上按中序遍历的最后一个结点；查找某结点 p 的后继的规律是：若 p->rtag=1，则 p->rchild 就指向后继，否则，其后继是 p 的右子树上按中序遍历的第一个结点。

```
 int TreeThrInsert(BiThrTree T,P,X)
    //在中序全线索二叉树 T 的结点 P 上，插入以 X 为根的中序全线索二叉树，返回插入结果信息。
  {if(P->ltag==0 && P->rtag==0) {printf("P 有左右子女，插入失败\n");
return(0); }
   if(P->ltag==0)    //P 有左子女，将 X 插为 P 的右子女
    {if(X->ltag==1) X->lchild=P;  //若 X 无左子树， X 的左线索(前驱)是 P
     else  //寻找 X 的左子树上最左（下）边的结点
       {q=X->lchild;
        while(q->ltag==0) q=q->lchild;
        q->lchild=P;
        }
     if(X->rtag==1)    //修改 X 的右线索
       X->rchild=P->rchild;  //将 P 的右线索改为 X 的右线索
     else  //找 X 右子树最右面的结点
       {q=X->rchild; while(q->rtag==0) q=q->rchild;
        q->rchild=P->rchild;
        }
      P->rtag=0;P->rchild=X;   //将 X 作为 P 的右子树
    } //结束将 X 插入为 P 的右子树
  else    //P 有右子女，将 X 插入为 P 的左子女
    {if(X->ltag==1)    //X 无左子女
       X->lchild=P->lchild;  //将 P 的左线索改为 X 的左线索
     else    //X 有左子女，找 X 左子树上最左边的结点
       {q=X->lchild;
```

```
        while(q->ltag==0) q=q->lchild;
        q->lchild=P->lchild;
        }
      if(X->rtag==1) X->rchild=P;    //若 X 无右子树，则 X 的右线索是 P
      else       //X 有右子树，查其右子树中最右边的结点，将该结点的后继修改为 P
        {q=X->rchild;
        while(q->rtag==0) q=q->rchild;
        q->rchild=P;
        }
      P->ltag=0; //最后将 P 的左标记置 0
      P->lchild=X; //P 的左子女链接到 X
    } //结束将 X 插入为 P 的左子女
  }  //结束 Tree Thrfnsert
```

63.[题目分析]在中序线索树中，非递归查找数据域为 A 的结点（设该结点存在，其指针为 P）并将数据域为 x 的 Q 结点插入到左子树中。若 P 无左子女，则 Q 成为 P 的左子女,原 P 的左线索成为 Q 的左线索，Q 的右线索为 P；若 P 有左子树，设 P 左子树中最右结点的右线索是结点 Q，结点 Q 的右线索是 P。

```
  void InThrInsert(BiThrTree T,Q; ElemType A)
    //在中序线索二叉树 T 中,查找其数据域为 A 的结点，并在该结点的左子树上插入结点 Q
  {BiThrTree P=T;
   while(P)
    {while(P->LT==0 && P->data!=A) P=P->LL; //沿左子树向下
     if (P->data==A) break;     //找到数据域为 A 的结点,退出循环
     while(P->RT==1) P=P->RL;  //还没找到数据域为 A 的结点沿右线索找后继
     P=P->RL;    //沿右子树向下
     }
  if(P->LT==1)  //P 没有左子树，Q 结点插入作 P 的左子女
    {Q->LL=P->LL; Q->LT=1;  //将 P 的左线索作为 Q 的左线索
     }
  else  //P 有左子树，应修改 P 的左子树最右结点的线索
     {Q->LL=P->LL;Q->LT=0; //Q 成为 P 的左子女
      s=Q->LL;              //s 指向原 P 的左子女
      while(s->RT==0) s=s->RL; //查找 P 的左子树最右边的结点
      s->RL=Q;                 //原 P 左子树上最右结点的右线索是新插入结点 Q
      }
    P->LT=0;P->LL=Q; //修改 P 的标记和指针
    Q->RT=1;Q->RL=P; //将 Q 链为 P 的左子女,其中序后继是 P;
   }//结束 InThrInsert
```

64.[题目分析]“双链”就利用二叉树结点的左右指针，重新定义左指针为指向前驱的指针，右指针是指向后继的指针，链表在遍历中建立，下面采用中序遍历二叉树。

```
  BiTree head=null,pre; //全局变量链表头指针 head,pre
  void CreatLeafList(BiTree bt) //将 BiTree 树中所有叶子结点链成带头结点的双链表，
```

```
{if (bt)        //若 bt 不空
{CreatLeafList (bt->lchild) ;                //中序遍历左子树
 if(bt->lchild==null && bt->rchild==null)  //叶子结点
   if(head==null)//第一个叶子结点
     {head=(BiTree)malloc(sizeof(BiNode)); //生成头结点
      head->lchild=null; head->rchild=bt;  //头结点的左链为空,右链指向第一个叶子结点
      bt->lchild=head; pre=bt;  //第一个叶子结点左链指向头结点，pre 指向当前叶子结点
     }
  else    //已不是第一个叶子结点
    {pre->rchild=bt; bt->lchild=pre; pre=bt;} //当前叶子结点链入双链表
 CreatLeafList(bt->rchild);  //中序遍历右子树
 pre->rchild=null;     //最后一个叶子结点的右链置空（链表结束标记）
}//if(bt)   }//结束 CreatLeafList;
```

65.[题目分析]求中序全线索树任意结点 p 的前序后继，其规则如下：若 p 有左子女，则左子女就是其前序后继；若 p 无左子女而有右子女，则 p 的右子女就是 p 的前序后继；若 p 无左右子女，这时沿 p 的右线索往上，直到 p 的右标志为 0（非线索），这时若 p 的右子女为空，则表示这是中序遍历最后一个结点，故指定结点无前序后继，否则，该结点就是指定结点的前序后继。程序段如下：

```
if(p->ltag==0 && p->lchild!=null) return(p->lchild); //p 的左子女是 p 的前序后继
   else if(p->rtag==0) && p->rchild!=null) return(p->rchild);//p 右子女是其前序后继
     else   //p 无左右子女，应沿右线索向上（找其前序后继），直到某结点右标记为 0
      {while (p->rtag==1) p=p->rchild;
       if (p->rchild) return(p->rchild);else return(null); }//指定结点的前序后继
```

[算法讨论]请注意题目“中序序列第一结点的左标志和最后结点的右标志皆为 0 （非线索），对应指针皆为空”的说明。若无这一说明，只要结点的左标记为 0，其左子女就是其前序后继。最后，当 p 无子女，沿右线索向上找其前序后继时，若最后结点的右标志为 0，但对应指针为空，p 也无前序后继。

66.[题目分析] 不借助辅助堆栈实现中序遍历，必须解决如何查找后继的问题。使用线索树就行。为此，将结点结构修改为(ltag,lchild,data,rchild,rtag)。各字段的含义在上面已多次使用，不再介绍。设二叉树已中序线索化。下面首先编写一查中序后继的函数，接着是中序遍历的非递归算法。

```
BiTree  After(BiThrTree t)  //查中序线索二叉树上结点 t 的后继
{if (t->rtag==1) return(t->rchild);
 p=t->rchild;
 while(p->ltag==0) p=p->lchild; //p 右子树中最左下的结点是 p 的中序后继
 return(p); } //if
void InOrder(BiThrTree bt)
  //非递归中序遍历带头结点的中序线索二叉树 bt
```

```
{p=bt->lchild; //p 指向原二叉树的根结点
if (p!=bt) //二叉树非空
  {while (p->ltag==0) p=p->lchild; //找中序遍历的第一个结点
   while (p!=bt)  //没回到头结点，就一直找后继并遍历
     {visit(*p); p=After(p); }
  }//if  }结束算法 InOrder
```

67.[题目分析]在中序穿线树中找结点的双亲，最简单情况是顺线索就可找到。例如，结点的左子女的右线索和右子女的左线索都指向双亲。但对于有左右子女的结点来说，则要利用中序穿线树中线索“向上”指向祖先的特点：若结点 p 是结点 q 右子树中中序遍历最左下的结点，p 的左线索指向 q；若结点 p 是结点 q 左子树上中序遍历最右下的结点，p 的右线索指向是 q。反过来，通过祖先找子女就容易了。另外，若结点 q 的后继是中序穿线树的头结点，则应特殊考虑。

```
void FFA(BiThrTree t,p,q)//在中序穿线树 t 上，求结点 p 的双亲结点 q
{q=p;  //暂存
 while(q->RTAG==0) q=q->RLINK; //找 p 的中序最右下的结点
 q=q->RLINK;           //顺右线索找到 q 的后继（p 的祖先结点）
 if (q==t) q=t->LLINK; //若后继是头结点，则转到根结点
 if (q==p) {printf(“根结点无双亲\n”); return; }
 if (q->LLINK==p) return(q); else q=q->LLINK; //准备到左子树中找 p
 while (q->RLINK!=p) q=q->RLINK; return(q); } //找最右结点的过程中回找到 p
}//结束 FFA
```

[算法讨论]本题也可以先求结点 p 最左下结点的前驱线索，请读者自己写出算法。

68.[题目分析]带头结点的中序线索树，其头结点的 lchild 指向二叉树的根结点，头结点的 rchild 域指向中序遍历的最后一个结点。而二叉树按中序遍历的第一个结点的 lchild 和最后一个结点的 rchild 指向头结点。故从头结点找到根结点后，顺“后继”访问二叉树。在中序线索树中，找前序的后继，已在第 65 题进行了详细的讨论，这里不再赘述。中序线索树在上面的“四、应用题”已画过多个，这里也不重复。

```
void PreorderInThreat(BiTrhTree tbt)
  //前序遍历一中序全线索二叉树 tbt，tbt 是头结点指针
{bt=tbt->lchild;
 while(bt)
    {while(bt->ltag==0){printf(bt->data); bt=bt->lchild;}//沿左分枝向下
     printf(bt->data);   //遍历其左标志为 1 的结点，准备右转
     while(bt->rtag==1 && bt->rchild!=tbt) bt=bt->rchild;//沿右链向上
     if (bt->rchild!=tbt) bt=bt->rchild;//沿右分枝向下
    }
}//结束 PreorderInThreat
```

时间复杂度 O(n)。

69.[题目分析]线索化是在遍历中完成的，因此，对于二叉树进行前序、中序、后序遍历，在“访问根结点”处进行加线索的改造，就可实现前序，中序和后序的线索化

```
BiThrTree pre=null;//设置前驱
void PreOrderThreat(BiThrTree BT)
  //对以线索链表为存储结构的二叉树 BT 进行前序线索化
{if (BT!=null)
```

```
    {if (BT->lchild==null){BT->ltag=1; BT->lchild=pre;}//设置左线索
     if (pre!=null && pre->rtag==1) pre->rchild=BT;     //设置前驱的右线索;
     if (BT->rchild==null) BT->rtag=1;                  //为建立右链作准备
     pre=BT;//前驱后移
     if (BT->ltag==0) PreOrderThreat(BT->lchild);       //左子树前序线索化
  PreOrderThreat(BT->rchild);                           //右子树前序线索化
  }//if (BT!=null)  }结束 PreOrderThreat
70. BiThrTree pre==null;
  void InOrderThreat(BiThrTree T)//对二叉树进行中序线索化
  {if (T)
    {InOrderThreat(T->lchild);   //左子树中序线索化
       if (T->lchild==null) {T->ltag=1; T->lchild=pre; } //左线索为 pre;
       if (pre!=null && pre->rtag==1) pre->rchild=T;}     //给前驱加后继线索
       if (T->rchild==null) T->rtag=1;                    //置右标记，为右线索
    作准备
       pre=BT;//前驱指针后移
       InOrderThreat(T->rchild);                          //右子树中序线索化
     }  }//结束 InOrderThreat
71. void InOrderThreat(BiThrTree thrt)
   //thrt 是指向中序全线索化头结点的指针，本算法中序遍历该二叉树
   {p=thrt->lchild;  //p 指向二叉树的根结点，当二叉树为空时，p 指向 thrt
    whild(p!=thrt)
     {while (p->ltag==0) p=p->lchild;//沿左子女向下
      visit(*p);//访问左子树为空的结点
      while(p->rtag==1 && p->rchild!=thrt){p=p->rchild;visit(*p);}//沿右线索访
   问后继结点
      p=p->rchild;//转向右子树
     }  }//结束 InOrderThread
72.[题目分析]若使新插入的叶子结点 S 成 T 右子树中序序列的第一个结点，则应在 T 的右子树中最左面的结点（设为 p）处插入，使 S 成为结点 p 的左子女。则 S 的前驱是 T，后继是 p.
  void ThrTreeInsert(BiThrTree T,S)
     //在中序线索二叉树 T 的右子树上插入结点 S，使 S 成为 T 右子树中序遍历第一个结点
  {p=T->rchild;            //用 p 去指向 T 的右子树中最左面的结点
  while(p->ltag==0) p=p->lchild;
  S->ltag=1;S->rtag=1;     //S 是叶子，其左右标记均为 1
  S->lchild=T;S->rchild=p;//S 的前驱是根结点 T，后继是结点 p
  p->lchild=S;p->ltag=0;  //将 p 的左子女指向 S ，并修改左标志为 0
  }//结束 ThrTreeInsert
73. BiThrTree InOrder(BiThrTree T,ElemType x)
     //先在带头结点的中序线索二叉树 T 中查找给定值为 x 的结点，假定值为 x 的结点存在
  {p=T->lchild;//设 p 指向二叉树的根结点
```

```
while(p!=T)
  {while(p->ltag==0 && p-data!=x) p=p->lc;
   if(p->data==x)return(p);
   while(p->rtag==1 && p->rc!=T) {p=p->rc; if(p->data== x) return(p);}
   p=p->rc;  }
}//结束 InOrder
```

BiThrTree AfterXNode(BiThrTree T)//在中序线索二叉树 T 中，求给定值为 x 的结点的后继结点

```
{BiThrTree p=InOrde(T,x);          //首先在 T 树上查找给定值为 x 的结点，由 p 指向
 if(p->rtag==1) return(p->rc);    //若 p 的左标志为 1，则 p 的 rc 指针指向其后继
 else {q=p->rc; while(q->ltag==0)q=q->lc; return(q); }
     //结点 p 的右子树中最左面的结点是结点 p 的中序后继
}  }//结束 AfterXnode
```

74. [题目分析]后序遍历是“左-右-根”，因此，若结点有右子女，则右子女是其后序前驱，否则，左子女（或左线索）指向其后序前驱。

BiThrTree PostSucc (BiThrTree T,p)//在后序线索二叉树 T 中，查找指定结点 p 的直接前驱 q

```
{if(p->Rtag==0) q=p->Rchild;//若 p 有右子女，则右子女为其前驱
 else q=p->Lchild;           //若 p 无右子女，左子女或左线索就是 p 的后序前驱
 return (q);
}//结束 PostSucc
```

75. BiThrTree InSucc(BiThrTree T,p) //在对称序穿线树 T 中，查找给定结点 p 的中序后继

```
{if (p->rtag==1) q=p->rchild;   //若 p 的右标志为 1，用其右指针指向后继
  else {q=p->rchild; while(q->ltag==0) q=q->lchild; }//p 的后继为其右子树中最左下的结点
   return (q);
}//结束 InSucc
```

76. [题目分析]在后序序列中，若结点 p 有右子女，则右子女是其前驱，若无右子女而有左子女，则左子女是其前驱。若结点 p 左右子女均无，设其中序左线索指向某祖先结点 f（p 是 f 右子树中按中序遍历的第一个结点），若 f 有左子女，则其左子女是结点 p 在后序下的前驱；若 f 无左子女，则顺其前驱找双亲的双亲，一直继续到双亲有左子女（这时左子女是 p 的前驱）。还有一种情况，若 p 是中序遍历的第一个结点，结点 p 在中序和后序下均无前驱。

BiThrTree InPostPre (BiThrTree t,p)

```
//在中序线索二叉树 t 中，求指定结点 p 在后序下的前驱结点 q
{BiThrTree q;
 if (p->rtag==0) q=p->rchild;        //若 p 有右子女，则右子女是其后序前驱
 else if (p->ltag==0) q=p->lchild;  //若 p 无右子女而有左子女，左子女是其后序前驱。
   else  if(p->lchild==null) q=null;//p 是中序序列第一结点，无后序前驱
     else //顺左线索向上找 p 的祖先，若存在，再找祖先的左子女
       {while(p->ltag==1 && p->lchild!=null) p=p->lchild;
```

```
        if(p->ltag==0) q=p->lchild;  //p 结点的祖先的左子女是其后序前驱
        else  q=null;   //仅右单枝树（p 是叶子），已上到根结点，p 结点无后序前驱
         }
    return(q); }//结束 InPostPre
```

77. [题目分析]在中序穿线二叉树中，求某结点 p 的后序后继 q，需要知道 p 的双亲结点 f 的信息。（中序线索树的性质；若 p 是 f 的左子女，则 f 是 p 的最右子孙的右线索；若 p 是 f 的右子女，则 f 是 p 的最左子孙的左线索。）找到 f 后，若 p 是 f 的右子女，则 p 的后继是 f；若 p 是 f 的左子女，且 f 无右子女，则 p 的后继也是 f；若 p 是 f 的左子女，且 f 有右子女，则 f 的右子树中最左边的叶子结点是 p 的后继。因此，算法思路是，先找 p 的双亲 f，根据 p 是 f 的左/右子女再确定 p 的后序后继。

```
BiThrTree InThrPostSucc(BiThrTree r,p)
  //在中序线索二叉树 r 上，求结点 p（假定存在）的后序后继结点 q）
{if(p==r)return(null)      //若 p 为根结点，后序后继为空
 T=p
 while(T->LT==1) T=T->LL; //找 p 的最左子孙的左线索
 q=T->LL;                 //q 是  p 最左子孙的左线索,是 p 的祖先
 if(q->RL==p) return(q);  //p 是 q 的右子女，则 q 是 p 后序后继。
 T=p;
 while(T->RT==1) T=T->RL; //找 p 的最右子孙的右线索
  q=T->RL;                 //q 是 p 最右子孙的右线索
  if(q->LL=p)              //若 p 是 q 的左子女
    if(q->RT==0) return(q);//若 p 是 q 的左子女且 q 无右子女，则 p 的后序后继是 q
   else      //p 的双亲 q 有右子树，则 q 的右子树上最左下的叶子结点是 p 的后继
    {q=q->RL;
     while(q->LT==1||q->RT==1)  //找 q 右子树中最左下的叶子结点
      {while(q->LT==1) q=q->LL; //向左下
       if(q->RT==1) q=q->RL;    //若 q 无左子女但有右子女，则向右下，直到叶子
结点
       }
     return(q); //q 是的 p 后继
    }
 } //结束 InThrPostSucc
```

[算法讨论] 请注意本题条件：标记为 0 时是线索，而为 1 时是指向子女。

78. [题目分析]第 77 题已讨论了在中序线索树中查找结点 p 的后序后继问题，本题要求在中序线索树上进行后序遍历。因后序遍历是“左右根”，最后访问根结点，即只有从右子树返回时才能访问根结点，为此设一标志 returnflag,当其为 1 时表示从右侧返回，可以访问根结点。为了找当前结点的后继，需知道双亲结点的信息，在中序线索树中，某结点最左子孙的左线索和最右子孙的右线索均指向其双亲，因此设立两个函数 LeftMost 和 RightMost 求结点的最左和最右子孙，为了判定是否是从右子树返回，再设一函数 IsRightChild。

```
BiThrTree LeftMost(BiThrTree t) //求结点 t 最左子孙的左线索
   {BiThrTree p=t;
   while(p->ltag==0) p=p->lchild; //沿左分枝向下
   if (p->lchild!=null) return(p->lchild); else return(null);
   }//LeftMost
```

```
BiThrTree RightMost (BiThrTree t) //求结点 t 最右子孙的右线索
   {BiThrTree p=t;
   while(p->rtag==0) p=p->rchild;   //沿右分枝向下
   if (p->rchild!=null) return (p->rchild); else return(null);
   }//RightMost
int IsRightChild(BiThrTree t,father)   //若 t 是 father 的右孩子，返回 1,否则返回 0
   {father=LeftMost(t);
    if(father &&f ather->rchild==t) return(1); else return(0);
   }//Is RightChild;
void PostOrderInThr (BiThrTree bt) //后序遍历中序线索二叉树 bt
   {BiThrTree father, p=bt;
    int flag;
    while(p!=null)
     {while(p->ltag==0 ) p=p->lchild; // 沿左分枝向下
      if(p->rtag==0) flag=0;//左孩子为线索，右孩子为链，相当从左返回
      else flag=1;          //p 为叶子，相当从右返回
      while(flag==1)
        {visit(*p);//访问结点
         if(IsRightChild(p,father)) {p=father; flag=1;} //修改 p 指向双亲
         else   //p 是左子女，用最右子孙的右线索找双亲
           {p=RightMost(p);
            if(p&&p->rtag==1) flag=1; else flag=0;
            }
        }// while(flag==1)
      if(flag==0 && p!=null) p=p->rchild; //转向当前结点右分枝
    } }//结束 PostOrderInThr
```

79.（1）哈夫曼树的构造过程

① 根据给定的 n 个权值$\{W_1,W_2,W_3,\cdots,W_n\}$构成 n 棵二叉树的集合$F=\{T_1,T_2,\cdots,T_n\}$,其中每棵二叉树$T_i$只有权值为$W_i$的根结点，其左右子树均为空。

② 在 F 中选取两棵根结点的权值最小的树作左右子树构造一棵新二叉树，新二叉树根结点的权值为其左右子树上根结点的权值之和。

③ 在 F 中删除这两棵树，同时将新得到的二叉树加入 F 中。

④ 重复②和③，直到 F 中只剩一棵树为止。这棵树便是哈夫曼树。

（2）含有 n 个叶子结点的哈夫曼树共有 2n-1 个结点，采用静态链表作为存储结构，设置大小为 2n-1 的数组。现将哈夫曼树的结点及树的结构定义如下：

```
typedef struct{float weight ;   //权值
               int parent,lc,rc;//双亲、左、右子女 }node ;
typedef node HufmTree[2*n-1];
void Huffman(int n,float w[n],HufmTree T)
 //构造 n 个叶子结点的哈夫曼树 T，n 个权值已放在 W[n]数组中
{int i,j,p1,p2          //p1,p2 为最小值和次最小值的坐标
float small1,small2;    //small1 和 small2 为权值的最小值和次小值
for(i=0;i<2*n-1;i++)    //置初值，结点的权、左、右子女，双亲
```

```
    {T[i].parent=-1; T[i].lc=-1; T[i].rc=-1;
     if(i<n) T[i].weight=w[i]; else T[i].weight=0;
    }
  for (i=n ;i<2*n-1;i++)          //构造新二叉树
    {p1=p2=0;small1=small2=maxint; //初值
    for(j=0;j<i;j++)
     if(T[j].weight<small1 && T[j].parent==-1)            //最小值
      {p2=p1; small2=small1; p1=j; small1=T[j].weight;}
    else if(T[j].weight<small2 && T[j].parent==-1)       //次小值
        {p2=j;small2=T[j].weight;}
    T[i].weight=T[p1].weight+T[p2].weight;               //合并成一棵新二叉树
    T[i].lc=p1; T[i].rc=p2;                              //置双亲的左右子女
    T[p1].parent=i; T[p2].parent=i;                      //置左、右子女的双亲
  }//for(i=0;i<2*n-1;i++) }//结束 huffman
```

求哈夫曼编码的算法

```
    typedef struct {char bit[n]; int start;}codetype;
    void HuffmanCode(CodeType code[n],HufmTree T) //哈夫曼树 T 已求出，现求其哈夫曼编码
    {int i,j,c,p;
     CodeType cd;
     for(i=0;i<n;i++)
     {cd.start=n;c=i;p=T[i].parent;
       while(p!=-1)
         {cd.start--;
           if(T[p].lc==c) cd.bit[cd.start]='0' //左分枝生成代码'0'
           else cd.bit[cd.start]='1' ;          // 右分枝生成代码'1'
           c=p; p=T[p].parent;                  //双亲变为新子女，再求双亲的双亲
          }
      code[i]=cd;         //成组赋值，求出一个叶子结点的哈夫曼编码
     }//for    }//结束 HuffmanCode
```

80. [题目分析] 二叉树的层次遍历序列的第一个结点是二叉树的根。实际上，层次遍历序列中的每个结点都是“局部根”。确定根后，到二叉树的中序序列中，查到该结点，该结点将二叉树分为“左根右”三部分。若左、右子树均有，则层次序列根结点的后面应是左右子树的根；若中序序列中只有左子树或只有右子树，则在层次序列的根结点后也只有左子树的根或右子树的根。这样，定义一个全局变量指针 R，指向层次序列待处理元素。算法中先处理根结点，将根结点和左右子女的信息入队列。然后，在队列不空的条件下，循环处理二叉树的结点。队列中元素的数据结构定义如下：

```
    typedef struct
       { int lvl;        //层次序列指针，总是指向当前“根结点”在层次序列中的位置
         int l,h;          //中序序列的下上界
         int f;            //层次序列中当前“根结点”的双亲结点的指针
         int lr;           // 1—双亲的左子树  2—双亲的右子树
       }qnode;
  BiTree Creat(datatype in[],level[],int n)
```

```
//由二叉树的层次序列 level[n]和中序序列 in[n]生成二叉树。 n 是二叉树的结点数
{if (n<1) {printf("参数错误\n"); exit(0);}
 qnode s,Q[];        //Q 是元素为 qnode 类型的队列，容量足够大
 init(Q); int R=0;  //R 是层次序列指针，指向当前待处理的结点
 BiTree p=(BiTree)malloc(sizeof(BiNode));          //生成根结点
 p->data=level[0]; p->lchild=null; p->rchild=null; //填写该结点数据
for (i=0; i<n; i++)      //在中序序列中查找根结点，然后，左右子女信息入队列
   if (in[i]==level[0]) break;
if (i==0)  //根结点无左子树，遍历序列的 1—n-1 是右子树
  {p->lchild=null;
   s.lvl=++R; s.l=i+1; s.h=n-1; s.f=p; s.lr=2;  enqueue(Q,s);
  }
else if (i==n-1) //根结点无右子树，遍历序列的 1—n-1 是左子树
          {p->rchild=null;
           s.lvl=++R; s.l=1; s.h=i-1; s.f=p; s.lr=1; enqueue(Q,s);
          }
     else   //根结点有左子树和右子树
       {s.lvl=++R; s.l=0; s.h=i-1; s.f=p; s.lr=1;enqueue(Q,s);
       //左子树有关信息入队列
        s.lvl=++R; s.l=i+1;s.h=n-1;s.f=p; s.lr=2;enqueue(Q,s);
        //右子树有关信息入队列
        }
while (!empty(Q))  //当队列不空，进行循环，构造二叉树的左右子树
 { s=delqueue(Q); father=s.f;
   for (i=s.l; i<=s.h; i++)
      if (in[i]==level[s.lvl]) break;
   p=(bitreptr)malloc(sizeof(binode));                    //申请结点空间
   p->data=level[s.lvl]; p->lchild=null; p->rchild=null;   //填写该结点数据
   if (s.lr==1) father->lchild=p;
    else father->rchild=p;  //让双亲的子女指针指向该结点
   if (i==s.l)
     {p->lchild=null;         //处理无左子女
      s.lvl=++R; s.l=i+1; s.f=p; s.lr=2;  enqueue(Q,s);
      }
   else if (i==s.h)
          {p->rchild=null;     //处理无右子女
           s.lvl=++R; s.h=i-1; s.f=p; s.lr=1; enqueue(Q,s);
           }
        else{s.lvl=++R; s.h=i-1; s.f=p; s.lr=1; enqueue(Q,s);
//左子树有关信息入队列
            s.lvl=++R; s.l=i+1; s.f=p; s.lr=2; enqueue(Q,s);
     //右子树有关信息入队列
            }
  }//结束 while (!empty(Q))
```

```
 return(p);
}//算法结束
```

第七章　图

一、选择题

1．图中有关路径的定义是（　　）。【北方交通大学 2001 一、24 （2 分）】

A．由顶点和相邻顶点序偶构成的边所形成的序列　　B．由不同顶点所形成的序列

C．由不同边所形成的序列　　D．上述定义都不是

2．设无向图的顶点个数为 n，则该图最多有（　）条边。

A．n-1　　B．n(n-1)/2　　C．n(n+1)/2　　D．0　　E．n^2

【清华大学 1998 一、5 （2 分）】【西安电子科技大 1998 一、6 （2 分）】

【北京航空航天大学 1999 一、7 （2 分）】

3．一个 n 个顶点的连通无向图，其边的个数至少为（　　）。【浙江大学 1999 四、4 (4 分)】

A．n-1　　B．n　　C．n+1　　D．nlogn；

4．要连通具有 n 个顶点的有向图，至少需要（　　）条边。【北京航空航天大学 2000 一、6(2 分)】

A．n-1　　B．n　　C．n+1　　D．2n

5．n 个结点的完全有向图含有边的数目（　　）。【中山大学 1998 二、9 （2 分）】

A．n*n　　B．n（n＋1）　　C．n / 2　　D．n*（n－1）

6．一个有 n 个结点的图，最少有（　　）个连通分量，最多有（　　）个连通分量。

A．0　　B．1　　C．n-1　　D．n

【北京邮电大学 2000 二、5 （20/8 分）】

7．在一个无向图中，所有顶点的度数之和等于所有边数（　　）倍，在一个有向图中，所有顶点的入度之和等于所有顶点出度之和的（　　）倍。【哈尔滨工业大学 2001 二、3 （2 分）】

A．1/2　　B．2　　C．1　　D．4

8．用有向无环图描述表达式(A+B)*((A+B)/A)，至少需要顶点的数目为(　)。【中山大学 1999 一、14】

A．5　　B．6　　C．8　　D．9

9．用 DFS 遍历一个无环有向图，并在 DFS 算法退栈返回时打印相应的顶点，则输出的顶点序列是(　)。

A．逆拓扑有序　　B．拓扑有序　　C．无序的　　【中科院软件所 1998】

10．下面结构中最适于表示稀疏无向图的是（　　），适于表示稀疏有向图的是（　　）。

A．邻接矩阵　　B．逆邻接表　　C．邻接多重表　　D．十字链表　　E．邻接表

【北京工业大学 2001 一、3 (2 分)】

11．下列哪一种图的邻接矩阵是对称矩阵？（　　）【北方交通大学 2001 一、11 （2 分）】

A．有向图　　B．无向图　　C．AOV 网　　D．AOE 网

12．　从邻接阵矩 $A=\begin{bmatrix}0&1&0\\1&0&1\\0&1&0\end{bmatrix}$ 可以看出，该图共有（①）个顶点；如果是有向图该图共有（②） 条弧；如果是无向图，则共有（③）条边。【中科院软件所 1999 六、2（3 分）】

①．A．9　　B．3　　C．6　　D．1　　E．以上答案均不正确

②. A. 5　　B. 4　　C. 3　　D. 2　　E. 以上答案均不正确

③. A. 5　　B. 4　　C. 3　　D. 2　　E. 以上答案均不正确

13. 当一个有 N 个顶点的图用邻接矩阵 A 表示时，顶点 Vi 的度是（ ）。【南京理工大学 1998 一、4(2 分)】

A. $\sum_{i=1}^{n} A[i,j]$　　B. $\sum_{j=1}^{n} A[i,j]$　　C. $\sum_{i=1}^{n} A[j,i]$　　D. $\sum_{i=1}^{n} A[i,j] + \sum_{j=1}^{n} A[j,i]$

14. 用相邻矩阵 A 表示图，判定任意两个顶点 Vi 和 Vj 之间是否有长度为 m 的路径相连，则只要检查（　　）的第 i 行第 j 列的元素是否为零即可。【武汉大学 2000 二、7】

A. mA　　B. A　　C. A^m　　D. Am-1

15. 下列说法不正确的是（　　）。【青岛大学 2002 二、9 （2 分)】

A. 图的遍历是从给定的源点出发每一个顶点仅被访问一次　C. 图的深度遍历不适用于有向图

B. 遍历的基本算法有两种：深度遍历和广度遍历　　D. 图的深度遍历是一个递归过程

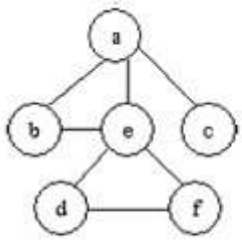
16. 无向图 G=(V,E)，其中：V={a,b,c,d,e,f},E={(a,b),(a,e),(a,c),(b,e),(c,f),(f,d),(e,d)}，对该图进行深度优先遍历，得到的顶点序列正确的是（　　）。【南京理工大学 2001 一、14 （1.5 分)】

A. a,b,e,c,d,f　　B. a,c,f,e,b,d　　C. a,e,b,c,f,d　　D. a,e,d,f,c,b

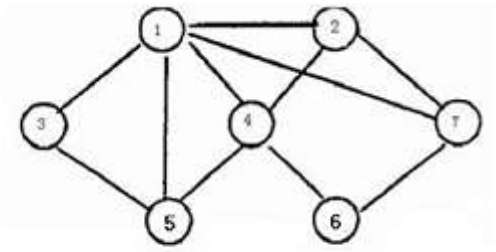
17. 设图如右所示，在下面的 5 个序列中，符合深度优先遍历的序列有多少？（　　）【南京理工大学 2000 一、20 （1.5 分)】

a e b d f c　　a c f d e b　　a e d f c b　　a e f d c b　　a e f d b c

A. 5 个　　B. 4 个　　C. 3 个　　D. 2 个



第 17 题图



第 18 题图

18. 下图中给出由 7 个顶点组成的无向图。从顶点 1 出发，对它进行深度优先遍历得到的序列是(①)，而进行广度优先遍历得到的顶点序列是（ ②)。【中科院软件所 1999 六、2-（1）(2 分)】

①. A. 1354267　　B. 1347652　　C. 1534276　　D. 1247653　　E. 以上答案均不正确

②. A. 1534267　　B. 1726453　　C. 1354276　　D. 1247653　　E. 以上答案均不正确

19. 下面哪一方法可以判断出一个有向图是否有环（回路)：【东北大学 2000 4、2（4 分)】

A. 深度优先遍历　B. 拓扑排序　C. 求最短路径　D. 求关键路径

20. 在图采用邻接表存储时，求最小生成树的 Prim 算法的时间复杂度为(　　)。

A. O(n)　　B. O(n+e)　　C. $O(n^2)$　　D. $O(n^3)$

【合肥工业大学 2001 一、2 (2分)】

21. 下面是求连通网的最小生成树的prim算法:集合VT,ET分别放顶点和边,初始为(1),下面步骤重复n-1次:a:(2);b:(3);最后:(4)。【南京理工大学 1997 一、11_14 (8分)】

(1). A. VT，ET为空　　B. VT为所有顶点，ET为空
C. VT为网中任意一点，ET为空　　D. VT为空，ET为网中所有边

(2). A. 选i属于VT，j不属于VT，且（i，j）上的权最小
B. 选i属于VT，j不属于VT，且（i，j）上的权最大
C. 选i不属于VT，j不属于VT，且（i，j）上的权最小
D. 选i不属于VT，j不属于VT，且（i，j）上的权最大

(3). A. 顶点i加入VT，(i,j) 加入ET　　B. 顶点j加入VT，(i,j) 加入ET
C. 顶点j加入VT，(i,j) 从ET中删去　　D. 顶点i,j加入VT，(i,j) 加入ET

(4). A. ET 中为最小生成树　　B. 不在ET中的边构成最小生成树
C. ET中有n-1条边时为生成树，否则无解　　D. ET中无回路时，为生成树，否则无解

22. (1). 求从指定源点到其余各顶点的迪杰斯特拉（Dijkstra）最短路径算法中弧上权不能为负的原因是在实际应用中无意义;

(2). 利用Dijkstra求每一对不同顶点之间的最短路径的算法时间是$O(n^3)$;（图用邻接矩阵表示）

(3). Floyd求每对不同顶点对的算法中允许弧上的权为负，但不能有权和为负的回路。

上面不正确的是（　　）。【南京理工大学 2000 一、21 (1.5分)】

A. (1),(2),(3)　　B. (1)　　C. (1),(3)　　D. (2),(3)

23. 当各边上的权值( )时，BFS算法可用来解决单源最短路径问题。【中科院计算所2000一、3 (2分)】

A. 均相等　　B. 均互不相等　　C. 不一定相等

24. 求解最短路径的 Floyd 算法的时间复杂度为(　　)。【合肥工业大学 1999 一、2 (2分)】

A. O（n）　　B. O（n+c）　　C. O（n*n）　　D. O（n*n*n）

25. 已知有向图G=(V,E)，其中$V=\{V_1,V_2,V_3,V_4,V_5,V_6,V_7\}$，
$E=\{<V_1,V_2>,<V_1,V_3>,<V_1,V_4>,<V_2,V_5>,<V_3,V_5>,<V_3,V_6>,<V_4,V_6>,<V_5,V_7>,<V_6,V_7>\}$,G的拓扑序列是（　）。

A. $V_1,V_3,V_4,V_6,V_2,V_5,V_7$　　B. $V_1,V_3,V_2,V_6,V_4,V_5,V_7$
C. $V_1,V_3,V_4,V_5,V_2,V_6,V_7$　　D. $V_1,V_2,V_5,V_3,V_4,V_6,V_7$

【北京航空航天大学 2000 一、7 (2分)】

26. 若一个有向图的邻接距阵中,主对角线以下的元素均为零,则该图的拓扑有序序列(　　)。

A. 存在　　B. 不存在【中科院计算所1998 二、6 (2分)】【中国科技大学 1998 二、6 (2分)】

27. 一个有向无环图的拓扑排序序列（　　）是唯一的。【北京邮电大学 2001 一、3 (2分)】

A. 一定　　B. 不一定

28. 在有向图 G 的拓扑序列中，若顶点 Vi 在顶点 Vj 之前，则下列情形不可能出现的是(　　)。

A．G 中有弧<Vi，Vj>　　B．G 中有一条从 Vi 到 Vj 的路径

C．G 中没有弧<Vi,Vj>　　D．G 中有一条从 Vj 到 Vi 的路径

【南京理工大学 2000 一、9 （1.5 分)】

29. 在用邻接表表示图时，拓扑排序算法时间复杂度为(　　)。

A. O(n)　　B. O(n＋e)　　C. O(n*n)　　D. O(n*n*n)

【合肥工业大学 2000 一、2 （2 分)】【南京理工大学 2001 一、9 （1.5 分)】

【青岛大学 2002 二、3 （2 分)】

30. 关键路径是事件结点网络中（　　)。【西安电子科技大学 2001 应用 一、4 （2 分)】

A．从源点到汇点的最长路径　　B．从源点到汇点的最短路径

C．最长回路　　D．最短回路

31. 下面关于求关键路径的说法不正确的是（　　)。【南京理工大学 1998 一、12 （2 分)】

A．求关键路径是以拓扑排序为基础的

B．一个事件的最早开始时间同以该事件为尾的弧的活动最早开始时间相同

C．一个事件的最迟开始时间为以该事件为尾的弧的活动最迟开始时间与该活动的持续时间的差

D．关键活动一定位于关键路径上

32．下列关于 AOE 网的叙述中，不正确的是（　　)。

A．关键活动不按期完成就会影响整个工程的完成时间

B．任何一个关键活动提前完成，那么整个工程将会提前完成

C．所有的关键活动提前完成，那么整个工程将会提前完成

D．某些关键活动提前完成，那么整个工程将会提前完成

【北方交通大学 1999 一、7 （3 分)】【北京工业大学 1999 一、1 (2 分)】

二、判断题

1.树中的结点和图中的顶点就是指数据结构中的数据元素。(　　)【青岛大学 2001 四、1（1 分)】

2．在 n 个结点的无向图中，若边数大于 n-1,则该图必是连通图。(　　)【中科院软件所 1997 一、4(1 分)】

3．对有 n 个顶点的无向图，其边数 e 与各顶点度数间满足下列等式 $e=\sum_{i=1}^{n} TD(V_i)$ 。（　）

【南京航空航天大学 1996 六、4 （1 分)】

4. 有 e 条边的无向图，在邻接表中有 e 个结点。(　　)【南京理工大学 1998 二、5 （2 分)】

5. 有向图中顶点 V 的度等于其邻接矩阵中第 V 行中的 1 的个数。(　　)【合肥工业大学 2001 二、7(1 分)】

6．强连通图的各顶点间均可达。(　　)【北京邮电大学 2000 一、3 （1 分)】

7．强连通分量是无向图的极大强连通子图。(　　)【北京邮电大学 2002 一、7 （1 分)】

8．连通分量指的是有向图中的极大连通子图。(　　)【燕山大学 1998　二、4 （2 分)】

9．邻接多重表是无向图和有向图的链式存储结构。(　　)【南京航空航天大学 1995 五、5（1 分)】

10. 十字链表是无向图的一种存储结构。(　　)【青岛大学 2001 四、7 （1 分)】

11. 无向图的邻接矩阵可用一维数组存储。(　　)【青岛大学 2000 四、5 （1 分)】

12．用邻接矩阵法存储一个图所需的存储单元数目与图的边数有关。(　　)

【东南大学 2001 一、4 （1 分)】 【中山大学 1994 一、3 （2 分)】

13.有 n 个顶点的无向图, 采用邻接矩阵表示, 图中的边数等于邻接矩阵中非零元素之和的一半。(　　)

【北京邮电大学 1998 一、5 （2 分)】

14. 有向图的邻接矩阵是对称的。(　　)【青岛大学 2001 四、6 （1 分)】

15. 无向图的邻接矩阵一定是对称矩阵，有向图的邻接矩阵一定是非对称矩阵。(　　)

【东南大学 2001 一、3 （1 分)】【哈尔滨工业大学 1999 三、4】

16. 邻接矩阵适用于有向图和无向图的存储，但不能存储带权的有向图和无向图，而只能使用邻接表存储形式来存储它。(　　)【上海海运学院 1995 一、9（1 分） 1997 一、8（1 分） 1998 一、9（1 分)】

17. 用邻接矩阵存储一个图时，在不考虑压缩存储的情况下，所占用的存储空间大小与图中结点个数有关，而与图的边数无关。(　　)【上海海运学院 1996 一、8 （1 分） 1999 一、9 （1 分)】

18. 一个有向图的邻接表和逆邻接表中结点的个数可能不等。(　　)【上海交通大学 1998 一、12】

19. 需要借助于一个队列来实现 DFS 算法。(　　)【南京航空航天大学 1996 六、8 （1 分)】

20. 广度遍历生成树描述了从起点到各顶点的最短路径。(　　)【合肥工业大学 2001 二、8 （1 分)】

21. 任何无向图都存在生成树。(　　)【北京邮电大学 2000 一、1 （1 分)】

22. 不同的求最小生成树的方法最后得到的生成树是相同的.(　　)【南京理工大学 1998 二、3 （2 分)】

23. 带权无向图的最小生成树必是唯一的。(　　)【南京航空航天大学 1996 六、7 （1 分)】

24. 最小代价生成树是唯一的。(　　)【山东大学 2001 一、5 （1 分)】

25. 一个网（带权图）都有唯一的最小生成树。(　　)【大连海事大学 2001 一、14 （1 分)】

26. 连通图上各边权值均不相同，则该图的最小生成树是唯一的。(　　)【哈尔滨工业大学 1999 三、3】

27. 带权的连通无向图的最小（代价）生成树（支撑树）是唯一的。(　　)【中山大学 1994 一、10（2 分)】

28. 最小生成树的 KRUSKAL 算法是一种贪心法（GREEDY)。(　　)【华南理工大学 2002 一、6（1 分)】

29. 求最小生成树的普里姆(Prim)算法中边上的权可正可负。(　　)【南京理工大学 1998 二、2 （2 分)】

30. 带权的连通无向图的最小代价生成树是唯一的。(　　)【东南大学 2001 一、5（1 分)】

31. 最小生成树问题是构造连通网的最小代价生成树。(　　)【青岛大学 2001 四、10（1 分)】

32. 在图 G 的最小生成树 G1 中，可能会有某条边的权值超过未选边的权值。(　　)

【合肥工业大学 2000 二、7（1 分)】

33. 在用 Floyd 算法求解各顶点的最短路径时，每个表示两点间路径的 $path^{k-1}[I,J]$一定是 $path^{k}$ [I,J]的子集(k=1,2,3,…,n)。(　　)【合肥工业大学 2000 二、6 （1 分)】

34. 拓扑排序算法把一个无向图中的顶点排成一个有序序列。(　)【南京航空航天大学 1995 五、8（1 分)】

35. 拓扑排序算法仅能适用于有向无环图。(　　)【南京航空航天大学 1997 一、7 （1 分)】

36. 无环有向图才能进行拓扑排序。(　　)【青岛大学 2002 一、7 （1 分）2001 一、8 （1

分)】
37. 有环图也能进行拓扑排序。(　　)【青岛大学 2000 四、6 (1 分)】
38. 拓扑排序的有向图中，最多存在一条环路。(　　)【大连海事大学 2001 一、6(1 分)】
39. 任何有向图的结点都可以排成拓扑排序，而且拓扑序列不唯一。(　　)【上海交通大学 1998 一、13】
40. 既使有向无环图的拓扑序列唯一，也不能唯一确定该图。(　　)【合肥工业大学 2001 二、6(1 分)】
41. 若一个有向图的邻接矩阵对角线以下元素均为零，则该图的拓扑有序序列必定存在。(　　)
【中科院软件所 1997 一、5 (1 分)】
42. AOV 网的含义是以边表示活动的网。(　　)【南京航空航天大学 1995 五、7 (1 分)】
43.对一个 AOV 网,从源点到终点的路径最长的路径称作关键路径。【南京航空航天大学 1995 五、9(1 分)】
44. 关键路径是 AOE 网中从源点到终点的最长路径。(　　)【青岛大学 2000 四、10(1 分)】
45. AOE 网一定是有向无环图。(　　)【青岛大学 2001 一、9 (1 分)】
46. 在表示某工程的 AOE 网中,加速其关键路径上的任意关键活动均可缩短整个工程的完成时间。(　　)
【长沙铁道学院 1997 一、2 (1 分)】
47. 在 AOE 图中，关键路径上某个活动的时间缩短，整个工程的时间也就必定缩短。(　　)
【大连海事大学 2001 一、15 (1 分)】
48. 在 AOE 图中，关键路径上活动的时间延长多少，整个工程的时间也就随之延长多少。(　　)
【大连海事大学 2001 一、16 (1 分)】
49. 当改变网上某一关键路径上任一关键活动后，必将产生不同的关键路径。【上海交通大学 1998 一、14】

三、填空题

1.判断一个无向图是一棵树的条件是_____。
2. 有向图 G 的强连通分量是指_____。【北京科技大学 1997 一、7】
3. 一个连通图的_____是一个极小连通子图。【重庆大学 2000 一、1】
4. 具有 10 个顶点的无向图，边的总数最多为_____。【华中理工大学 2000 一、7 (1 分)】
5. 若用 n 表示图中顶点数目，则有_____条边的无向图成为完全图。【燕山大学 1998 一、6(1 分)】
6. 设无向图 G 有 n 个顶点和 e 条边，每个顶点 Vi 的度为 di (1<=i<=n)，则 e=_____
【福州大学 1998 二、2 (2 分)】
7. G 是一个非连通无向图，共有 28 条边，则该图至少有_____个顶点。
【西安电子科技大 2001 软件一、8 (2 分)】
8. 在有 n 个顶点的有向图中，若要使任意两点间可以互相到达，则至少需要_____条弧。
【合肥工业大学 2000 三、8 (2 分)】
9. 在有 n 个顶点的有向图中，每个顶点的度最大可达_____。【武汉大学 2000 一、3】
10. 设 G 为具有 N 个顶点的无向连通图，则 G 中至少有_____条边。
【长沙铁道学院 1997 二、2 (2 分)】
11. n 个顶点的连通无向图，其边的条数至少为_____。【哈尔滨工业大学 2000 二、2(1 分)】

12．如果含 n 个顶点的图形形成一个环，则它有_____棵生成树。

【西安电子科技大学　2001 软件　一、2 （2 分）】

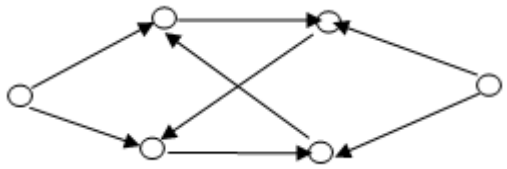
13．N 个顶点的连通图的生成树含有_____条边。【中山大学 1998 一、9 （1 分）】

14．构造 n 个结点的强连通图，至少有_____条弧。【北京轻工业学院 2000　一、4（2 分）】

15．有 N 个顶点的有向图，至少需要量_____条弧才能保证是连通的。【西南交通大学 2000 一、3】

16．右图中的强连通分量的个数为（　　）个。

【北京邮电大学 2001 二、5 （2 分）】



17．N 个顶点的连通图用邻接矩阵表示时,该矩阵
至少有_____个非零元素。【中科院计算所 1998 一、6（1 分）】【中国科技大学 1998 一、6（15/6 分）】

18．在图 G 的邻接表表示中，每个顶点邻接表中所含的结点数，对于无向图来说等于该顶点的_____；对于有向图来说等于该顶点的_____。

【燕山大学 2001 二、5 （3 分）】

19. 在有向图的邻接矩阵表示中，计算第 I 个顶点入度的方法是_____。【青岛大学 2002 三、7 （2 分）】

20. 对于一个具有 n 个顶点 e 条边的无向图的邻接表的表示，则表头向量大小为_____，邻接表的边结点个数为_____。【青岛大学 2002 三、8 （2 分）】

21. 遍历图的过程实质上是_____，breath-first search 遍历图的时间复杂度_____；depth-first search 遍历图的时间复杂度_____，两者不同之处在于_____，反映在数据结构上的差别是_____。

【厦门大学 1999 一、3】

22. 已知一无向图 G=（V，E），其中 V={a,b,c,d,e } E={(a,b),(a,d),(a,c),(d,c),(b,e)} 现用某一种图遍历方法从顶点 a 开始遍历图，得到的序列为 abecd，则采用的是_____遍历方法。

【南京理工大学 1996 二、2 （2 分）】

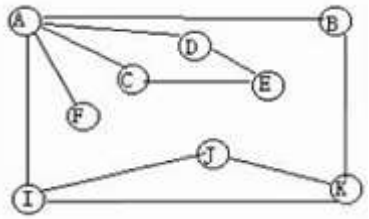
23. 一无向图 G（V，E），其中 V（G）={1,2,3,4,5,6,7}，E（G）={（1,2）,(1,3)，(2,4)，(2,5)，(3,6)，(3,7)，(6,7)（5,1）},对该图从顶点 3 开始进行遍历，去掉遍历中未走过的边，得一生成树 G’（V，E’）,V（G’）=V（G），E（G’）={（1,3)，(3,6)，(7,3)，(1,2)，(1,5)，(2,4）}，则采用的遍历方法是_____。

【南京理工大学 1997 三、6 （1 分）】

24. 为了实现图的广度优先搜索，除了一个标志数组标志已访问的图的结点外，还需_____存放被访问的结点以实现遍历。【南京理工大学 1999 二、9 （2 分）】

25. 按下图所示，画出它的广度优先生成树_____和深度优先生成树_____。

【西安电子科技大学 1998 三、6 （5 分）】



26. 构造连通网最小生成树的两个典型算法是_____。【北京科技大学 1998 一、5】

27. 求图的最小生成树有两种算法，_____算法适合于求稀疏图的最小生成树。【南京理工大学 2001 二、6（2 分）】

28. Prim（普里姆）算法适用于求_____的网的最小生成树；kruskal（克鲁斯卡尔）算法适用于求_____的网的最小生成树。【厦门大学 1999 一、4】

29. 克鲁斯卡尔算法的时间复杂度为_____,它对_____图较为适合。【中科院计算所 1999 二、3 （2 分）】

30. 对于含 N 个顶点 E 条边的无向连通图，利用 Prim 算法生成最小代价生成树其时间复杂度为_____，利用 Kruskal 算法生成最小代价生成树其时间复杂度为_____。【长沙铁道学院 1998 二、2 (4 分)】

31. 下面描述的是一种构造最小生成树算法的基本思想。设要处理的无向图包括 n 个节点 V1，V2,...，Vn，用相邻矩阵 A 表示，边的权全是正数。请在下列划线处填上正确叙述。

（1）. 若（Vi，Vj）是边，则 A（i，j）的值等于_____，若（Vi，Vj）不是边，则 A（i，j）的值是一个比任何边的权_____，　矩阵的对角线元素全为 0。

（2）. 构造最小生成树过程中，若节点 Vi 已包括进生成树，就把相邻矩阵的对角线元素 A（i，i）置成_____，若（Vi，Vj）已包括进生成树，就把矩阵元素 A（i，j）置成_____。

（3）.算法结束时，相邻矩阵中_____的元素指出最小生成树的_____。【山东工业大学 1998 二、4(6 分)】

32. 有一个用于 n 个顶点连通带权无向图的算法描述如下：

（1）. 设集合 T1 与 T2，初始均为空；

（2）. 在连通图上任选一点加入 T1；

（3）. 以下步骤重复 n-1 次：

a.在 i 属于 T1，j 不属于 T1 的边中选最小权的边；

b.该边加入 T2。

上述算法完成后，T2 中共有_____条边，该算法称_____算法，T2 中的边构成图的_____。【南京理工大学 1999 二、7 （4 分）】

33. 有向图 G 可拓扑排序的判别条件是_____。【长沙铁道学院 1998 二、9(2 分)】

34. Dijkstra 最短路径算法从源点到其余各顶点的最短路径的路径长度按_____次序依次产生,该算法弧上的权出现_____情况时,不能正确产生最短路径。【南京理工大学 1999 二、8（4 分）】

35. 求从某源点到其余各顶点的 Dijkstra 算法在图的顶点数为 10，用邻接矩阵表示图时计算时间约为 10ms,则在图的顶点数为 40，计算时间约为_____ms。【南京理工大学 2000 二、3 （1.5 分）】

36. 求最短路径的 Dijkstra 算法的时间复杂度为_____。【哈尔滨工业大学 2001 一、5 （2 分）】

37.有向图 G=(V,E),其中 V(G)={0,1,2,3,4,5}，用<a,b,d>三元组表示弧<a,b>及弧上的权 d.E(G)为{<0,5,100>,<0,2,10><1,2,5><0,4,30><4,5,60><3,5,10><2,3,50><4,3,20>}，则从源点 0 到顶点 3 的最短路径长度是_____,经过的中间顶点是_____。【南京理工大学 1998

三、6 （4 分)】

38. 上面的图去掉有向弧看成无向图则对应的最小生成树的边权之和为_____。

【南京理工大学 1998 三、7（4 分)】

39. 设有向图有 n 个顶点和 e 条边，进行拓扑排序时，总的计算时间为_____。

【西安电子科技大学 1999 软件 一、7 （2 分)】【武汉大学 2000 一、7】

40. AOV 网中,结点表示_____,边表示_____。AOE 网中,结点表示_____,边表示_____。

【北京理工大学 2001 七、3 （2 分)】

41. 在 AOE 网中，从源点到汇点路径上各活动时间总和最长的路径称为_____。【重庆大学 2000 一、2】

42. 在 AOV 网 中，存在环意味着_____，这是_____的；对程序的数据流图来说，它表明存在_____。

【厦门大学 1999 一、2】

43. 当一个 AOV 网用邻接表表示时，可按下列方法进行拓扑排序。

（1). 查邻接表中入度为_____的顶点，并进栈；

（2). 若栈不空，则①输出栈顶元素 Vj，并退栈；②查 Vj 的直接后继 Vk，对 Vk 入度处理，处理方法是_____；

（3). 若栈空时，输出顶点数小于图的顶点数，说明有_____，否则拓扑排序完成。

【南京理工大学 1996 二、3 （6 分)】

44. 已知图的邻接表结构为：

```
CONST  vtxnum={图的顶点数}
TYPE vtxptr=1..vtxnum;
     arcptr=^arcnode;
     arcnode=RECORD adjvex:vtxptr; nextarc:arcptr END;
     vexnode=RECORD vexdata:{和顶点相关的信息}; firstarc:arcptr  END;
     adjlist=ARRAY[vtxptr]OF  vexnode;
```

本算法是实现图的深度优先遍历的非递归算法。其中，使用一个顺序栈 stack。栈顶指针为 top。visited 为标志数组。

```
PROC dfs(g:adjlist;v0:vtxptr);
   top=0; write(v0); visited[v0]:=ture; p:=g[v0].firstarc;
   WHILE (top<>0)OR(p<>NIL)DO
     [WHILE(1)_____DO
       [v:=p^.adjvex;
         IF(2)_____ THEN  p:=p^.nextarc
           ELSE [write(v);  visited[v]:=true; top:=top+1;  stack[top]:=p;
(3)_____] ]
      IF top<>0  THEN[p:=stack[top]; top:=top-1; (4)_____]
     ]
ENDP.同济大学 2000 二、2 (10 分)】
```

45. 下面的算法完成图的深度优先遍历，请填空。

```
PROGRAM graph_traver;
CONST nl=max_node_number;
TYPE vtxptr=1..nl; vtxptr0=0..nl;
   arcptr=^arcnode;
   arcnode=RECORD vexi ,vexj: vtxptr; nexti, nextj: arcptr; END;;
```

```
    vexnode=RECORD vexdata: char; firstin, firstout: arcptr; END;
    graph=ARRAY[vtxptr0] OF vexnode ;
VAR ga:graph; n: integer;
    visited: ARRAY[vtxptr0] OF boolean ;
FUNC order (g: graph; v: char): vtxptr;
    (1)_____; i:=n;
   WHILE g[i].vexdata<>v DO i:=i-1;
   order:=i;
ENDF;
PROC  creat(var g: graph);
 readln(n,e);
 FOR  i:=  1  TO  n  DO  [readln(g[i].vexdata);  g[i].firstin  :=NIL  ;
g[i].firstout:=NIL;]
 FOR k:= 1 TO e DO [readln (vt,vh);
       i:=order (g,vt); j:=order (g,vh);  new (p); p^.vexi:=i ;  p^.vexj:=j
       p^.nextj:=____(2)____;   ___(3)___ :=p;
       p^.nexti:=:____(4)____;  ___(5)___ :=p;]
ENDP;
FUNC firstadj(g:graph; v:char): vtxptr0;
  i:=order(g,v); p:=g[i].firstout;
 IF p<>NIL THEN firstadj:=(6)_____ELSE firstadj:=0;
ENDF;
FUNC nextadj(g:graph; v:char; w:char): vtxptr0;
  i:=order(g,v); j:=order(g,w); p:=(7)_____;
  WHILE(p<>NIL ) AND (p^.vexj<>j) DO(8)_____;
  IF (9)_____AND(10)_____THEN  nextadj:=p^.nexti^.vexj  ELSE  nextadj:=0;
ENDF;
PROC dfs(g:graph; v0:char);
  write(v0:2); visited[order(g,v0)]:=true;  w:=(11)_____;
  WHILE w<>0 DO
   [IF (12)_____ THEN dfs(g,g[w].vexdata);
    w:=(13)_____;]
ENDP;
PROC traver(g:graph);
  FOR i:=1 TO n DO visited[i]:=false;
  FOR i:=1 TO n DO IF NOT visited[i] THEN  dfs(g,g[i].vexdata);
ENDP;
BEGIN
creat(ga); traver(ga);
END.
```

【北方交通大学 1999 三（20 分）】

46. n 个顶点的有向图用邻接矩阵 array 表示，下面是其拓扑排序算法，试补充完整。

注：(1). 图的顶点号从 0 开始计； (2). indegree 是有 n 个分量的一维数组，放顶点的入度；

(3).函数 crein 用于算顶点入度；(4).有三个函数 push(data),pop(),check()

其含义为数据 data 进栈，退栈和测试栈是否空（不空返回 1，否则 0）。

```
crein( array ,indegree,n)
 { for (i=0;i<n;i++)  indegree[i]= ((1)_____)
   for(i=0,i<n;i++)
      for (j=0;j<n; j++)  indegree[i]+=array[(2)_____][(3)_____];
 }
 topsort (array,indegree,n)
   { count= ((4)_____)
   for (i=0;i<n;i++)  if ((5)_____)  push(i)
   while (check( ))
    { vex=pop( );  printf(vex);  count++;
      for (i=0;i<n;i++)
        { k=array(6)_____
         if ((7)_____ )  { indegree[i]--;  if ((8)_____ )  push(i);}
        }
    }
    if( count<n)   printf(""图有回路");
   }  【南京理工大学 2000 三、4 （6 分)】
```

47. 假设给定的有向图是用邻接表表示，作为输入的是图中顶点个数 n 和边的个数 m, 以及图的 m 条边。在下面的程序中，我们用 readdata 程序过程输入图的信息，并建立该图的邻接表；利用 topol 程序过程获得图中顶点的一个拓扑序列。

```
PROGRAM  topol_order(input , output) ;
  CONST  maxn=20 ;
  TYPE  nodeptr=^nltype ;
        nltype=RECORD num : integer ; link : nodeptr  END ;
        chtype=RECORD count : integer ;  head : nodeptr  END ;
  VAR  ch : ARRAY [1 .. maxn]  OF  chtype ; m , n , top : integer  ;
  PROCEDURE  readdata  ;
   VAR  i , j , u , v : integer ;   p : nodeptr  ;
   BEGIN
     write (' input  vertex  number  n= ' ); readln (n) ;
     write (' input  edge  number  m= ' ); readln(m) ;
     FOR  i:=1 TO n DO BEGIN ch[i].count:= 0; ch[i].head:=NIL  END;
     writeln(' input  edges  :' );
     FOR  j:= 1  TO  m  DO
       BEGIN  write( j :3 , ' : ' ) ;  readln( u , v ) ;  new( p ) ;
              ch[v].count:=ch[v].count+1; p^.num:=v; (1)____ ; (2)___;  END
   END ;
  PROCEDURE  topol ;
  VAR  i, j, k: integer;  t: nodeptr ;
  BEGIN
    top:= 0 ;
    FOR  i := 1  TO n  DO
      IF  ch[i].count=0  THEN  BEGIN ch[i].count := top ;top := i END;
```

```
        i:= 0 ;
        WHILE (3)____ DO
           BEGIN (4) __;    (5) __ ; write(j : 5) ;i:= i + 1 ;t:=ch[j].head ;
                WHILE  t<>NIL  DO
                BEGIN k := t^.num ; ch[k].count:=ch[k].count－1 ;
                      IF  ch[k].count=0  THEN   BEGIN  ch[k].count:=top;  top:=k
   END;
                      (6) _____ ;  END
             END ;  writeln;
         IF i<n THEN  writeln (' the network has a cycle!' )
      END;
  BEGIN
    readdata ;  writeln (' output  topol  order : ' );  topol
  END.
```

【复旦大学 1995 三 （18 分)】

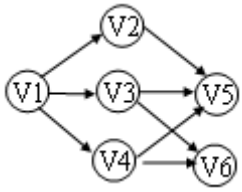
48. 如下为拓扑排序的 C 程序，

（1). 列出对右图执行该程序后的输出结果。

（2). 在程序空白处填上适当语句。

```
 void topsort(hdnodes graph [],int n)
   {int i,j,k,top; node_pointer ptr ;
    top=-1;
    for (i=0; i<n; i++)
      if (!graph[i].count){graph[i].count=top; top=i; }
    for (i=0; i<n; i++)
      if(1)____ {fprintf(stderr, "\ngraph  has  a  cycle  \n"); exit(1); }
      else {j=top;(2)_____;  printf(  "v%d,  " ,j) ;
            for (ptr=graph[j].link; ptr; ptr=ptr->link)
              {k=ptr->vertex; graph[k].count--;
               if((3)_____) {graph[k].count=top; top=k; }  }  }
  }
```

【浙江大学 2000 六(15 分)】



四、 应用题

1.（1). 如果 G1 是一个具有 n 个顶点的连通无向图，那么 G1 最多有多少条边？G1 最少有多少条边？

（2). 如果 G2 是一个具有 n 个顶点的强连通有向图，那么 G2 最多有多少条边？G2 最少有多少条边？

（3). 如果 G3 是一个具有 n 个顶点的弱连通有向图，那么 G3 最多有多少条边？G3 最少有多少条边？

【复旦大学 1997 一（9 分)】

2. n 个顶点的无向连通图最少有多少条边？n 个顶点的有向连通图最少有多少条边？

【山东大学 2000 一、3 (4 分)】

3．一个二部图的邻接矩阵 A 是一个什么类型的矩阵？【北京科技大学 1999 一、8（2 分）】

4．证明：具有 n 个顶点和多于 n-1 条边的无向连通图 G 一定不是树。【东南大学 1993 四（10 分）】

5．证明对有向图的顶点适当的编号，可使其邻接矩阵为下三角形且主对角线为全 0 的充要条件是该图为无环图。【北京邮电大学 2002 三 （10 分）】

6．用邻接矩阵表示图时，矩阵元素的个数与顶点个数是否相关？与边的条数是否有关？

【西安电子科技大学 2000 计应用 一、6（5 分）】

7．请回答下列关于图(Graph)的一些问题：（每题 4 分）

（1）．有 n 个顶点的有向强连通图最多有多少条边?最少有多少条边?

（2）．表示有 1000 个顶点、1000 条边的有向图的邻接矩阵有多少个矩阵元素？是否稀疏矩阵？

（3）．对于一个有向图，不用拓扑排序，如何判断图中是否存在环？【清华大学 2000 一(12 分)】

8．解答问题。设有数据逻辑结构为：

B = (K, R)， K = {k1, k2, …, k9}

R={<k1, k3>, <k1, k8>, <k2, k3>,<k2, k4>, <k2, k5>, <k3, k9>,<k5, k6>, <k8, k9>, <k9, k7>, <k4, k7>, <k4, k6>}

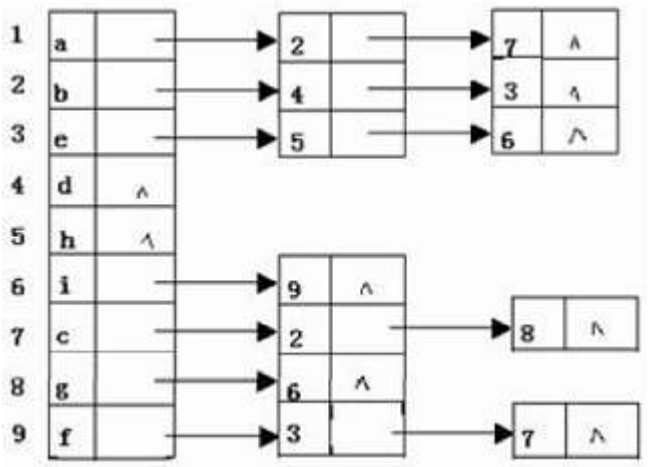
（1）．画出这个逻辑结构的图示。（3 分）

（2）．相对于关系 r，指出所有的开始接点和终端结点。（2 分）

（3）．分别对关系 r 中的开始结点，举出一个拓扑序列的例子。（4 分）

（4）．分别画出该逻辑结构的正向邻接表和逆向邻接表。（6 分）【山东工业大学 1999 三（15 分）】

9．有向图的邻接表存储如下：（1）．画出其邻接矩阵存储；（2）．写出图的所有强连通分量；（3）．写出顶点 a 到顶点 i 的全部简单路径。【东北大学 1997 一、5 (5 分)】



10．试用下列三种表示法画出网 G 的存储结构，并评述这三种表示法的优、缺点：（1）．邻接矩阵表示法；（2）．邻接表表示法；（3）．其它表示法。【华中理工大学 2000 三（12 分）】

11．已知无向图 G，V（G）={1，2，3，4}，E（G）={（1，2），（1，3），（2，3），（2，4），（3，4）}试画出 G 的邻接多表，并说明，若已知点 I，如何根据邻接多表找到与 I 相邻的点 j?

【东南大学 1994 一、2 （8 分） 1998 一、6（8 分）】

12. 如何对有向图中的顶点号重新安排可使得该图的邻接矩阵中所有的 1 都集中到对角线以上？

【清华大学 1999 一、5 （2 分）】

13. 假定 G＝（V，E）是有向图，V＝｛1,2,. . . ,N｝，N＞＝1，G 以邻接矩阵方式存储，G 的邻接矩阵为 A，即 A 是一个二维数组，如果 i 到 j 有边，则 A[i,j]=1，否则 A[i,j]=0，请给出一个算法，该算法能判断 G 是否是非循环图（即 G 中是否存在回路），要求算法的时间复杂性为 O(n*n)。

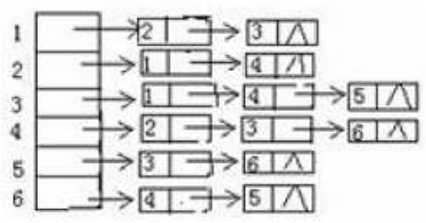
【吉林大学 1998 三(16 分)】

14. 首先将如下图所示的无向图给出其存储结构的邻接链表表示，然后写出对其分别进行深度，广度优先遍历的结果。 【天津大学 1999 一】

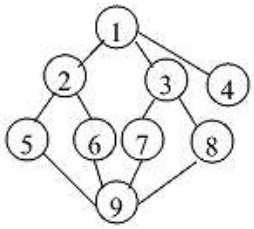
15. 下面的邻接表表示一个给定的无向图

（1）给出从顶点 v1 开始,对图 G 用深度优先搜索法进行遍历时的顶点序列；

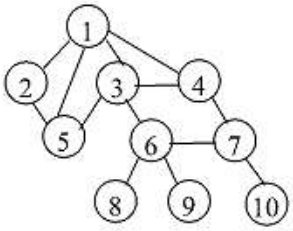
（2）给出从顶点 v1 开始,对图 G 用广度优先搜索法进行遍历时的顶点序列。【复旦大学 1998 六(10 分))



15 题图



14 题图



16 题图

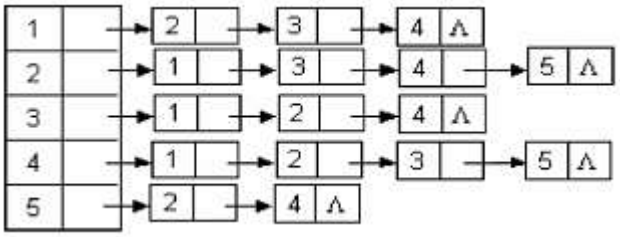
16. 给出图 G：

（1）. 画出 G 的邻接表表示图；

（2）. 根据你画出的邻接表，以顶点①为根，画出 G 的深度优先生成树和广度优先生成树。

【南开大学 1997 五 (14 分)】

17. 设 G=(V,E)以邻接表存储，如图所示，试画出图的深度优先和广度优先生成树。



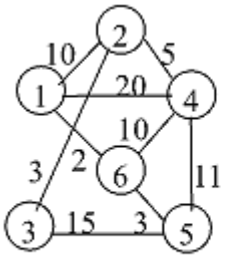
【北京轻工业学院 1998 八 （6 分)】

18. 对一个图进行遍历可以得到不同的遍历序列，那么导致得到的遍历序列不唯一的因素有哪些？

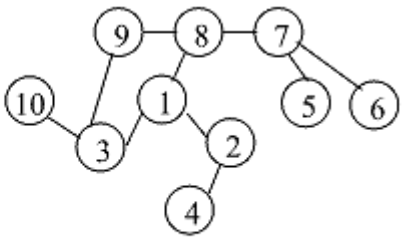
【北京航空航天大学 1998 一、7 （4 分)】

19. 解答下面的问题

（1）. 如果每个指针需要 4 个字节，每个顶点的标号占 2 个字节，每条边的权值占 2 个字节。下图采用哪种表示法所需的空间较多？为什么?



19 题图



20 题图

2). 写出下图从顶点 1 开始的 DFS 树。【西安电子科技大学 2000 计应用 六 (10 分)】

20. 如下所示的连通图，请画出：

(1). 以顶点①为根的深度优先生成树；(5 分)

(2). 如果有关节点，请找出所有的关节点。(5 分)【清华大学 1998 七 (10 分)】

21. 某田径赛中各选手的参赛项目表如下:

姓名	参 赛 项		
ZHAO	A	B	E
QIAN	C	D	
SHUN	C	E	F
LI	D	F	A
ZHOU	B	F	

设项目 A ,B ,…, F 各表示一数据元素,若两项目不能同时举行,则将其连线(约束条件).

(1). 根据此表及约束条件画出相应的图状结构模型,并画出此图的邻接表结构;

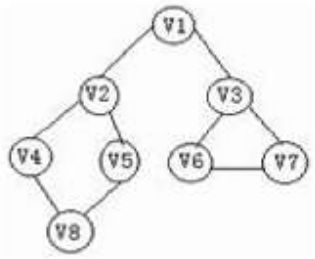
(2). 写出从元素 A 出发按“广度优先搜索”算法遍历此图的元素序列.

【北京科技大学 1999 五 2000 五 (12 分)】

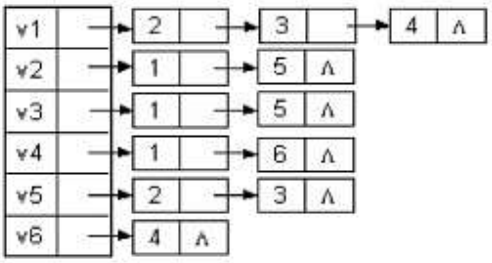
22. 已知无向图如下所示：

(1). 给出从 V1 开始的广度优先搜索序列；(2). 画出它的邻接表；

(3). 画出从 V1 开始深度优先搜索生成树。【燕山大学 2000 五 (5 分)】



第 22 题图



第 23 题图

23. 已知某图的邻接表为

(1). 写出此邻接表对应的邻接矩阵；(2 分)

(2). 写出由 v1 开始的深度优先遍历的序列；(2 分)

(3). 写出由 v1 开始的深度优先的生成树；(2 分)

(4). 写出由 v1 开始的广度优先遍历的序列；(2 分)

（5）. 写出由 v1 开始的广度优先的生成树；(2 分)

（6）. 写出将无向图的邻接表转换成邻接矩阵的算法。(8 分) 【山东大学 1998 六、18 分】

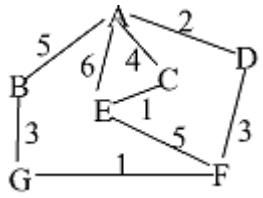
24. 考虑右图：

(1)从顶点 A 出发，求它的深度优先生成树

(2)从顶点 E 出发，求它的广度优先生成树

（3）根据普利姆(Prim) 算法，

求它的最小生成树【上海交通大学 1999 六 (12 分)】



25.在什么情况下，Prim 算法与 Kruskual 算法生成不同的 MST?

【西安电子科技大学 2000 计应用 一、11 (5 分)】

26. 下面是求无向连通图最小生成树的一种方法。

将图中所有边按权重从大到小排序为（e1,e2,…,em）

i:=1

WHILE (所剩边数 >=顶点数)

BEGIN

从图中删去 ei

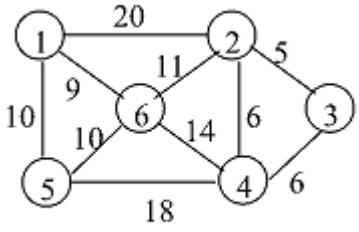
若图不再连通，则恢复 ei

i:=i+1

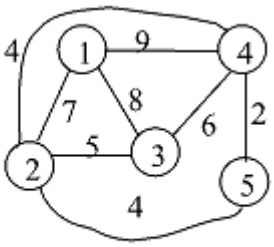
END.

试证明这个算法所得的图是原图的最小代价生成树。【北京邮电大学 1999 五 (10 分)】

27. 已知一个无向图如下图所示，要求分别用 Prim 和 Kruskal 算法生成最小树（假设以①为起点，试画出构造过程)。【哈尔滨工业大学 1999 九 (8 分)】



27 题图



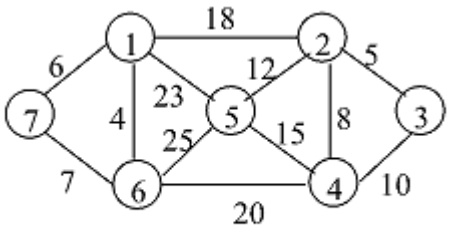
28 题图

28. G=(V,E)是一个带有权的连通图,则：

（1）. 请回答什么是 G 的最小生成树；

（2）. G 为下图所示,请找出 G 的所有最小生成树。【北方交通大学 1993 二 (12 分)】

29. 试写出用克鲁斯卡尔（Kruskal）算法构造下图的一棵最小支撑（或生成）树的过程。



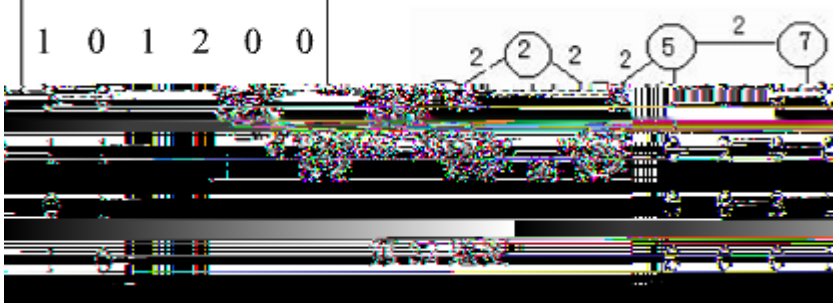
第 29 图

【吉林大学 2000 一、3 (3 分)】

30.求出下图的最小生成树。【合肥工业大学 1999 四、2 （5 分)】 第 30 题图

31．一带权无向图的邻接矩阵如下图 ，试画出它的一棵最小生成树。

$$\begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 2 & 0 & 0 \end{bmatrix}$$



32．请看下边的无向加权图。 （1）．写出它的邻接矩阵（ 5 分）

（2）.按 Prim 算法求其最小生成树，并给出构造最小生成树过程中辅助数组的各分量值（15 分）

辅助数组内各分量值:【华北计算机系统工程研究所 1999 四 （20 分)】

Y Closedge	2	3	4	5	6	7	8	U	V.-U
Vex Lowcost									
Vex Lowcost									
Vex Lowcost									
Vex Lowcost									
Vex Lowcost									
Vex Lowcost									
Vex Lowcost									
Vex Lowcost									

33．已知世界六大城市为:北京(Pe)、纽约(N)、巴黎(Pa)、 伦敦(L) 、 东京(T) 、 墨西

哥(M),下表给定了这六大城市之间的交通里程:

世界六大城市交通里程表(单位:百公里)

	PE	N	PA	L	T	M
PE		109	82	81	21	124
N	109		58	55	108	32
PA	82	58		3	97	92
L	81	55	3		95	89
T	21	108	97	95		113
M	124	32	92	89	113	

1	2	5
1	3	8
1	4	3
2	4	6
2	3	2
3	4	4
3	5	1
3	6	10
4	5	7
4	6	11
5	6	15

（1）. 画出这六大城市的交通网络图;

（2）. 画出该图的邻接表表示法;

（3）. 画出该图按权值递增的顺序来构造的最小(代价)生成树.

【上海海运学院 1995 六（9 分） 1999 五 （14 分)】

34. 已知顶点 1-6 和输入边与权值的序列（如右图所示)：每行三个数表示一条边的两个端点和其权值，共 11 行。请你：

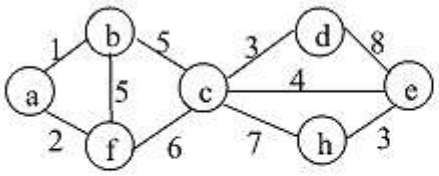
（1）. 采用邻接多重表表示该无向网，用类 PASCAL 语言描述该数据结构，画出存储结构示意图，要求符合在边结点链表头部插入的算法和输入序列的次序。

（2）. 分别写出从顶点 1 出发的深度优先和广度优先遍历顶点序列，以及相应的生成树。

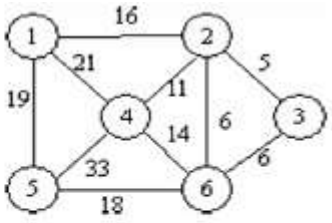
（3）. 按 prim 算法列表计算，从顶点 1 始求最小生成树，并图示该树。

【北京工业大学 1999 四(20 分)】

35. 下图表示一个地区的通讯网，边表示城市间的通讯线路，边上的权表示架设线路花费的代价，如何选择能沟通每个城市且总代价最省的 n-1 条线路，画出所有可能的选择。【东北大学 2000 一、4(4 分)】



第 36 题图



第 35 题图

36. 设无向网 G 如上：

（1）. 设顶点 a、b、c、d、e、f、h 的序号分别为 1、2、3、4、5、6、7，请列出网 G 的邻接矩阵、画出网 G 的邻接表结构:

（2）. 写出从顶点 a 出发,按“深度优先搜索”和“广度优先搜索”方法遍历网 G 所的到的顶点序列：

按 Prim 算法求出网 G 的一棵最小生成树。【北京科技大学 2001 五 (12 分)】

37. 有一图的邻接矩阵如下，试给出用弗洛伊德算法求各点间最短距离的矩阵序列 A^1，A^2，A^3，A^4。

$$A=\begin{bmatrix} 0 & 2 & \infty & \infty \\ \infty & 0 & 1 & 6 \\ 5 & \infty & 0 & 4 \\ 3 & \infty & \infty & 0 \end{bmatrix}$$

【北京邮电大学 2001 四、5(5 分)】

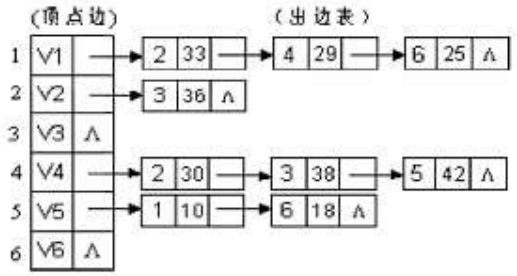
38. 下图所示是一带权有向图的邻接表法存储表示。其中出边表中的每个结点均含有三个字段，依次为边的另一个顶点在顶点表中的序号、边上的权值和指向下一个边结点的指针。试求：

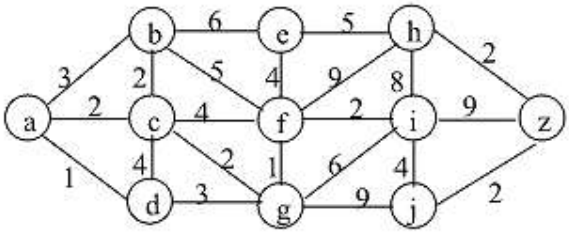
（1）. 该带权有向图的图形；

（2）. 从顶点 V1 为起点的广度优先周游的顶点序列及对应的生成树（即支撑树）；

（3）. 以顶点 V1 为起点的深度优先周游生成树；

（4）. 由顶点 V1 到顶点 V3 的最短路径。【中山大学 1994 四 （12 分）】



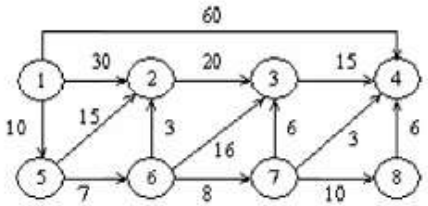


第 39 题图

39. 用最短路径算法，求如下图中 a 到 z 的最短通路。【西南财经大学 1999 四】

40. 已知一有向网的邻接矩阵如下，如需在其中一个结点建立娱乐中心，要求该结点距其它各结点的最长往返路程最短，相同条件下总的往返路程越短越好，问娱乐中心应选址何处？给出解题过程。

$$\begin{array}{c} \\ V1 \\ V2 \\ V3 \\ V4 \\ V5 \\ V6 \end{array}\begin{bmatrix} 0 & 2 & \infty & \infty & \infty & 3 \\ \infty & 0 & 3 & 2 & \infty & \infty \\ 4 & \infty & 0 & \infty & 4 & \infty \\ 1 & \infty & \infty & 0 & 1 & \infty \\ \infty & 1 & \infty & \infty & 0 & 3 \\ \infty & \infty & 2 & 5 & \infty & 0 \end{bmatrix}$$



第 41 题图

【北京邮电大学 2002 四、1 （10 分）】

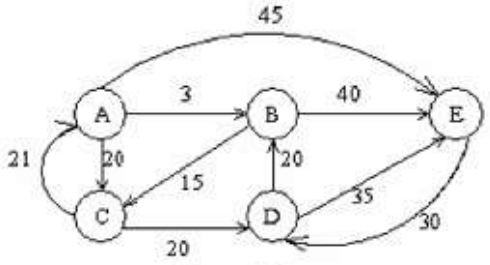
41. 求出下图中顶点 1 到其余各顶点的最短路径。【厦门大学 2002 八、2 （5 分）】

42. 试利用 Dijkstra 算法求下图中从顶点 a 到其他个顶点间的最短路径，写出执行算法过程中各步的状态。

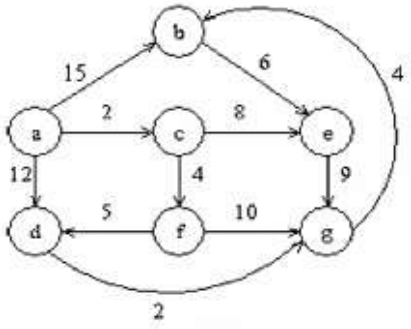
【东南大学 2000 四（10 分）】

43. 对于如下的加权有向图，给出算法 Dijkstra 产生的最短路径的支撑树，设顶点 A 为源

点，并写出生成过程。【吉林大学 1999 一、2 (4 分)】



第 43 题图



第 42 题图

44．已知图的邻接矩阵为：

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
V1	0	1	1	1	0	0	0	0	0	0
V2	0	0	0	1	1	0	0	0	0	0
V3	0	0	0	1	0	1	0	0	0	0
V4	0	0	0	0	0	1	1	0	1	0
V5	0	0	0	0	0	0	1	0	0	0
V6	0	0	0	0	0	0	0	1	1	0
V7	0	0	0	0	0	0	0	0	1	0
V8	0	0	0	0	0	0	0	0	0	1
V9	0	0	0	0	0	0	0	0	0	1
V10	0	0	0	0	0	0	0	0	0	0

当用邻接表作为图的存储结构，且邻接表都按序号从大到小排序时，试写出：

（1）．以顶点 V1 为出发点的唯一的深度优先遍历；

（2）．以顶点 V1 为出发点的唯一的广度优先遍历；

（3）．该图唯一的拓扑有序序列。【同济大学 1998 一 (12 分)】

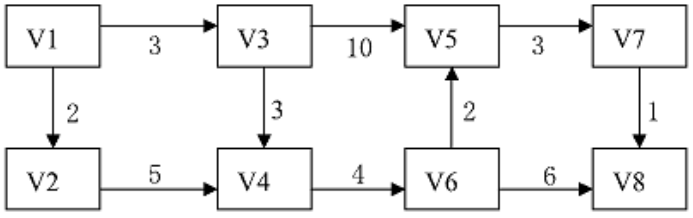
45．已知一图如下图所示：

（1）．写出该图的邻接矩阵；

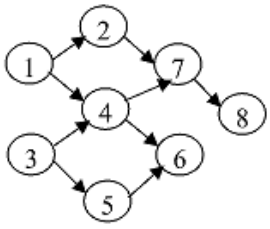
（2）．写出全部拓扑排序；

（3）．以 v1 为源点,以 v8 为终点，给出所有事件允许发生的最早时间和最晚时间，并给出关键路径；

（4）．求 V1 结点到各点的最短距离。【北京邮电大学 2000 五 （15 分）】



第 45 题图



第 46 题图

46．（1）．对于有向无环图，叙述求拓扑有序序列的步骤；

（2）．对于以下的图，写出它的四个不同的拓扑有序序列。【南开大学 1998 二 (12 分)】

47．有向图的拓扑排序能否用图的深度搜索模式来查找?若能，请简述方法，若不能，请简述原因

【西北大学 2000 二、8 (5 分)】

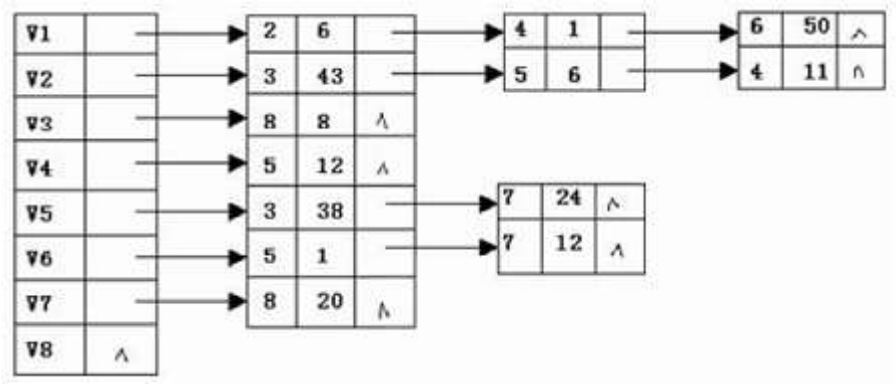
48．下图是带权的有向图 G 的邻接表表示法，求：

（1）．以结点 V1 出发深度遍历图 G 所得的结点序列；

（2）．以结点 V1 出发广度遍历图 G 所得的结点序列；

（3）．从结点 V1 到结点 V8 的最短路径；

（4）．从结点 V1 到结点 V8 的关键路径。



【青岛海洋大学 1999 四（10 分)】

49．对有五个结点{ A,B, C, D, E}的图的邻接矩阵，

$$\begin{bmatrix} 0 & 100 & 30 & \infty & 10 \\ \infty & 0 & \infty & \infty & \infty \\ \infty & 60 & 0 & 20 & \infty \\ \infty & 10 & \infty & 0 & \infty \\ \infty & \infty & \infty & 50 & 0 \end{bmatrix}$$

1）．画出逻辑图 ；

（2）．画出图的十字链表存储；

（3）．基于邻接矩阵写出图的深度、广度优先遍历序列；

（4）．计算图的关键路径。

【华南师范大学 1999 三 (20 分)】

50．何为 AOE 网的始点和终点，一个正常的 AOE 网是否只有一个始点和一个终点？

【首都经贸大学 1997 一、4 （4 分)】

51．下表给出了某工程各工序之间的优先关系和各工序所需时间

（1）．画出相应的 AOE 网

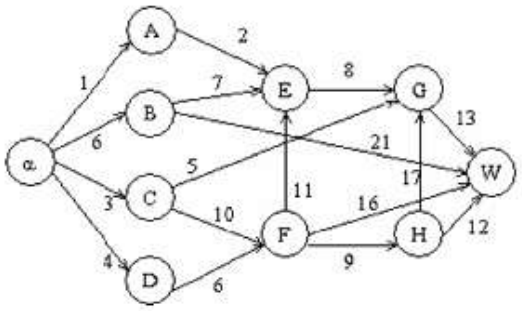
（2）．列出各事件的最早发生时间,最迟发生时间

（3）．找出关键路径并指明完成该工程所需最短时间. 【武汉交通科技大学 1996 二、6 (7 分)】

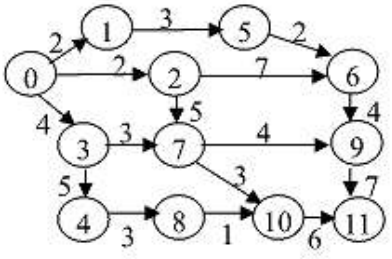
工序代号	A	B	C	D	E	F	G	H	I	J	K	L	M	N
所需时间	15	10	50	8	15	40	300	15	120	60	15	30	20	40
先驱工作	--	--	A,B	B	C,D	B	E	G,I	E	I	F,I	H,J,K	L	G

52．对图示的 AOE 网络，计算各活动弧的 e(ai)和 l(ai)的函数值，各事件（顶点）的 ve（Vj）

和 vl (Vj)的函数值，列出各条关键路径。【北京轻工业学院 1997 四 （15 分）】



第 52 题图



第 53 题 工程作业的网络图

53．请写出应填入下列叙述中（　　）内的正确答案。

某一工程作业的网络图如图所示，其中箭头表示作业，箭头边的数字表示完成作业所需的天数。箭头前后的圆圈表示事件，圆圈中的数字表示事件的编号。用事件编号的序列（例如 0-2-7-9-11）表示进行作业的路径。

完成此工程的关键路径是（A）完成此工程所需的最少天数为（B）天，此工程中具有最大充裕天数的事件是（C），充裕天数是（D）。关键路径上的事件的充裕天数是（E）。【上海大学 2002 三 （10 分）】

五、算法设计题

1.（单独命题考生做）设无向图 G 有 n 个顶点，m 条边。试编写用邻接表存储该图的算法。（设顶点值用 1～n 或 0～n-1 编号） 【南京航空航天大学 1996 十二 （10 分）】

2．请用流程图或类高级语言(pascal 或 c)表示算法。已知有向图有 n 个顶点,请写算法,根据用户输入的偶对建立该有向图的邻接表。即接受用户输入的<vi,vj>(以其中之一为 0 标志结束),对于每条这样的边,申请一个结点,并插入到的单链表中,如此反复,直到将图中所有边处理完毕。提示：先产生邻接表的 n 个头结点（其结点数值域从 1 到 n)。【上海大学 2000 四 （16 分)】

3．设无向图 G 有 n 个点 e 条边，写一算法建立 G 的邻接多表，要求该算法时间复杂性为 O（n+e)，且除邻接多表本身所占空间之外只用 O（1）辅助空间。【东南大学 1995 六（16 分） 1997 二 （15 分)】

4．给出以十字链表作存储结构，建立图的算法，输入(i,j,v)其中 i,j 为顶点号，v 为权值。

【河海大学 1998 六 （10 分)】

5．设有向 G 图有 n 个点(用 1,2,…,n 表示),e 条边,写一算法根据其邻接表生成其反向邻接表,要求算法复杂性为 O(n+e)。【东南大学 1996 三 （13 分)】

类似本题的另外叙述有：

（1）下图（编者略）是有向图按出度建立的邻接表，试写一算法，将此出度邻接表改成入度建立的邻接表。【北京邮电大学 1993 五 （15 分)】

（2）编写算法实现以下功能：根据含有 n 个顶点的有向图邻接表，构造相应的逆邻接表。

【东南大学 1992 六（18 分)】

6．写出从图的邻接表表示转换成邻接矩阵表示的算法，用类 PASCAL 语言（或 C 语言）写成过程形式。

【南开大学 1998 四 (16 分)】

类似本题的另外叙述有：

（1）已知某个图的邻接表，试建立该图的相邻矩阵。【天津大学 1999 五】

7. 设已给出图的邻接矩阵，要求将邻接矩阵转换为邻接表，用类 pascal 语言写为过程形式。【南开大学 1998 四 （ 14 分)】

类似本题的另外叙述有：

（1）设已给出图的邻接矩阵，要求将图的邻接矩阵转化为邻接表，试实现其算法。【南开大学 2000 三 3】

（2）编写算法，将图的邻接矩阵存储改为邻接表的存储。【中山大学 1998 五、2 （10 分)】

8. 试写一算法，判断以邻接表方式存储的有向图中是否存在由顶点 V_i 到顶点 V_j 的路径（i<>j)。注意：算法中涉及的图的基本操作必须在存储结构上实现。【哈尔滨工业大学 2001 九 （12 分)】

类似本题的另外叙述有：

（1）设计一个深度优先搜索算法，以判断用邻接表方式存储的有向图中是否存在由顶点 Vi 到顶点 Vj（i≠j）的路径。【中山大学 1999 数 四 （15 分)】

（2）按图的宽度优先搜索法写一算法判别以邻接矩阵存储的有向图中是否存在由顶点 Vi 到顶点 Vj 的路径（i≠j)。【中山大学 1997 五 （10 分)】

（3）请用流程图或类高级语言(pascal 或 c)表示算法。写算法判别以邻接方式存储的无向图中是否存在由顶点 Vi 到顶点 Vj 的路径(i≠j)。【上海大学 1999 三、2 （14 分)】

9. 已知无向图采用邻接表存储方式，试写出删除边（i，j）的算法。 【东南大学 1999 三（10 分)】

类似本题的另外叙述有：

（1）一个无向连通图的存储结构以邻接表的形式给定，设计算法删除该图中的一条边(i,j)。

【北京工业大学 1996 二 （15 分)】

（2）无向图 G 已按下图（编者略）邻接表存储。试编写算法在该邻接表上操作，删除从顶点 I 到顶点 J 之间的一条边。【上海大学 1996 六（18 分)】

（3）设无向图 G 用邻接表表示，（编者略）请写出在该无向图中删除边 (i,j)的算法。

【青岛海洋大学 1999 五（13 分)】

10. 假设有向图以邻接表存储，试编写算法删除弧<V_i,V_j>的算法。【北京轻工业学院 1997 五（10 分)】

11. 假设有向图以十字链表存储，试编写算法，插入弧<V_i,V_j>。【北京轻工业学院 1998 四（14 分)】

12. 设有向图用邻接表表示,图有 n 个顶点,表示为 1 至 n,试写一个算法求顶点 k 的入度(1<k<n)。

【南京理工大学 1997 四、2（10 分)】

13. 假设以邻接矩阵作为图的存储结构，编写算法判别在给定的有向图中是否存在一个简单有向回路，若存在，则以顶点序列的方式输出该回路（找到一条即可)。(注：图中不存在顶点到自己的弧）

【清华大学 1994 六 （15 分)】

类似本题的另外叙述有：

（1）假定 G=（V，E）是有向图，V={1，2，…，n}，n>=1，G 以邻接矩阵方式存储，G 的邻接矩阵为 A，即 A 是一个二维数组，如果 i 到 j 有边，则 A[i,j]=1，否则 A[i,j]=0。请给出一个算法，该算法能判断 G 是否是非循环图（即 G 中是否存在回路)，要求算法的时间复杂性为 O（ n*n ）。

【吉林大学 1997 五 (16 分)】

14. 假设一个有向图 G 已经以十字链表形式存储在内存中，试写一个判断该有向图中是否有环路（回路）的算法。【东北大学 2000 四、3 (12 分)】

15. 用邻接多重表存储结构，编写 FIRST-ADJ(G,V)函数，函数返回值为第一个邻接点，若 V 没有邻接点，返回零。【北京工商大学 1999 四 (12 分)】

16. 在有向图 G 中，如果 r 到 G 中的每个结点都有路径可达，则称结点 r 为 G 的根结点。编写一个算法完成下列功能：

(1). 建立有向图 G 的邻接表存储结构；

(2).判断有向图 G 是否有根，若有，则打印出所有根结点的值。【东北大学 2001 五 (15 分)】

17. 试编写求无向图 G 的连通分量的算法。要求输出每一连通分量的顶点值。(设图 G 已用邻接表存储)

【南京航空航天大学 1995 十一（10 分)】

类似本题的另外叙述有：

(1)写出求无向图 G 中各连通分量的顶点集的算法 COMF(G)。可调用的运算是:FIRST_ADJ（G，V）--求顶点 V 的第一邻接点，NEXTADJ（G，V，W）--求顶点 V 关于 W 的下一个邻接点。

【北京科技大学 1998 八、2（10 分)】

(2) 编程求解无向图 G 的所有连通分量。 【南京航空航天大学 2000 七】

18. 设无向图 G 已用邻接表结构存储，顶点表为 GL[n] (n 为图中顶点数)，试用“广度优先搜索”方法，写出求图 G 中各连通分量的 C 语言描述算法：BFSCOM(GL)。(注：算法中可调用队列操作的基本算法。)

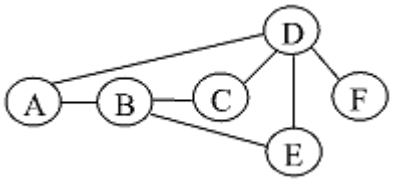
【北京科技大学 2001 七、2 (10 分)】

19. 设一个连通无向图 G=(V,E)采用邻接表方式存储，V={1,2,…,n}，一维数组 HAED[1…n]用来存放每个单链表的头指针，单链表中节点结构为(VER,LINK)，其中 LINK 是链接字段，VER 字段表示顶点内容，一维数组 MARK[1…n]用于对相应顶点加标号，MARK[i]=0 表示顶点 i 未被访问到, MARK[i]=1 表示顶点 i 已经被访问过,试写出对上述图 G 进行广度(或宽度)优先遍历(或访问)的非递归算法 BFS(HEAD,n,s,MARK,MARK)，其中 S 为任一遍历起始顶点。

【吉林大学 2000 二、1 (7 分)】

20. 写出图的深度优先搜索 DFS 算法的非递归算法。 【北京邮电大学 1994 十 (15 分)】

21. 已知连通图如下：



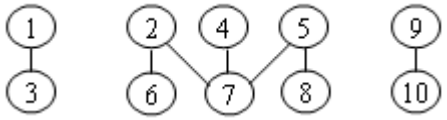
(1). 给出本图的邻接表；

(2). 若从顶点 B 出发对该图进行遍历，在（1）的基础上分别给出本图的按深度优先搜索和按广度优先搜索的顶点序列；

(3). 写出按深度优先搜索的递归程序。【厦门大学 2001 三 (12%分)】

22.试编写从某一顶点出发按深度优先搜索策略在图的邻接表上遍历一个强连通图的非递归算法。(用类 PASCAL 语言)【燕山大学 1999 十、2 (8 分)】

23. 设计算法以实现对无向图 G 的深度遍历，要求：将每一个连通分量中的顶点以一个表的形式输出。例如，下图的输出结果为：(1，3)(2，6，7，4，5，8)(9，10)



注：本算法中可以调用以下几个函数： firstadj（g,v）——返回图 g 中顶点 v 的第一个邻接点的号码，若不存在，则返回 0；

nextadj（g,v,w）——返回图 g 中顶点 v 的邻接点中处于 w 之后的邻接点的号码，若不存在，则返回 0。nodes（g）——返回图 g 中的顶点数） 【合肥工业大学 2000 五、4（8 分）】

24. 请设计一个图的抽象数据类型（只需要用类 PASCAL 或类 C/C++语言给出其主要功能函数或过程的接口说明，不需要指定存储结构，也不需要写出函数或过程的实现方法），利用抽象数据类型所提供的函数或过程编写图的宽度优先周游算法。算法不应该涉及具体的存储结构，也不允许不通过函数或过程而直接引用图结构的数据成员，抽象数据类型和算法都应该加足够的注释。【北京大学 1999 二、1（10 分）】

25. 设计算法以判断给定的无向图 G 中是否存在一条以 V0 为起点的包含所有顶点的简单路径，若存在，返回 TRUE，否则，返回 FALSE（注：本算法中可以调用以下几个函数：FIRSTADJ（G，V）——返回图 G 中顶点 V 的第一个邻接点的号码，若不存在，则返回 0；NEXTADJ（G，V，W）——返回图 G 中顶点 V 的邻接点中处于 W 之后的邻接点的号码，若不存在，则返回 0；NODES（G）——返回图 G 中的顶点数）

【合肥工业大学 1999 五、5 (8 分)】

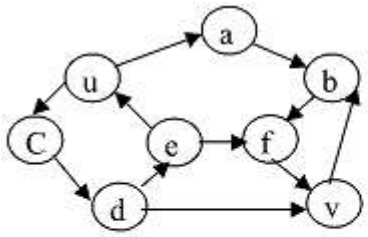
26. 已有邻接表表示的有向图，请编程判断从第 u 顶点至第 v 顶点是否有简单路径，若有则印出该路径上的顶点。要求：先描述图的存储结构，并简述算法思路；查找邻接点等图的运算要自己实现。(尽量采用非递归算法，否则满分 1 5 分)【北京工业大学 2000 六 （20 分）】

类似本题的另外叙述有：

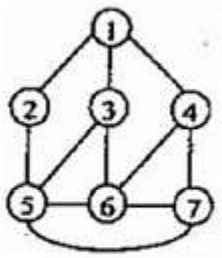
(1) 已知有向图和图中的两个结点 u 和 v，试编写算法求有向图中从 u 到 v 的所有简单路径。

【东南大学 2001 四 （15 分）】

(2) 已知有向图和图中两个顶点 U 和 V，编写算法求有向图中从 U 到 V 的所有简单路径，并以下图为例执行所编写的算法，画出相应的搜索过程图。【山东科技大学 2002 六 （18 分）】



第 26 题图



第 27 题图

27. 图的 D_搜索类似与 BFS,不同之处在于使用栈代替 BFS 中的队列 ,入出队列的操作改为入出栈的操作，即当一个顶点的所有邻接点被搜索之后,下一个搜索出发点应该是最近入栈(栈顶)的顶点。

（1）. 用邻接表做存储结构,写一个 D_搜索算法；（15 分）

（2）. 用 D_搜索方法的访问次序和相应的生成树，当从某顶点出发搜索它的邻接点，请按邻接点序号递增序搜索，以使答案唯一。（5 分）【中科院 1998 六 （20 分）】

28. 令 G=（V，E）为一个有向无环图，编写一个给图 G 中每一个顶点赋以一个整数序号的算法，并满足以下条件：若从顶点 i 至顶点 j 有一条弧则应使 i<j。【清华大学 1996 七】

29. 二部图（bipartite graph） G=（V，E）是一个能将其结点集 V 分为两不相交子集 V1 和 V2=V-V1 的无向图，使得：V1 中的任何两个结点在图 G 中均不相邻，V2 中的任何结点在图 G 中也均不相邻。

（1）. 请各举一个结点个数为 5 的二部图和非二部图的例子。

（2）. 请用 C 或 PASCAL 编写一个函数 BIPARTITE 判断一个连通无向图 G 是否是二部图，并分析程序的时间复杂度。设 G 用二维数组 A 来表示，大小为 n*n（n 为结点个数）。请在程序中加必要的注释。若有必要可直接利用堆栈或队列操作。【浙江大学 1998 八 （15 分）】

30. 我们可用“破圈法”求解带权连通无向图的一棵最小代价生成树。所谓“破圈法”就是“任取一圈，去掉圈上权最大的边”，反复执行这一步骤，直到没有圈为止。请给出用“破圈法”求解给定的带权连通无向图的一棵最小代价生成树的详细算法，并用程序实现你所给出的算法。注：圈就是回路。

【复旦大学 1997 六 （13 分）】

31. 设图用邻接表表示，写出求从指定顶点到其余各顶点的最短路径的 Dijkstra 算法。

要求：（1）. 对所用的辅助数据结构，邻接表结构给以必要的说明；（6 分）

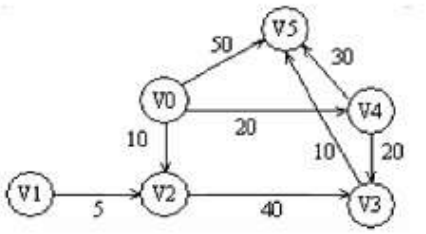
（2）. 写出算法描述。(C,类-Pascal,类-C 均可)（14 分）

【南京理工大学 1996 四、1 （20 分）】

类似本题的另外叙述有：

（1）写出求从某个源点到其余各顶点最短路径的 Dijkstra 算法。要求说明主要的数据结构及其作用，最后针对所给有向图，利用该算法，求 V0 到各顶点的最短距离和路线，即填写下表：

终点	从 V0 到到各终点的 dist 的值和最短距离和路线				
V1					
V2					
V3					
V4					
V5					
Vj	V2	V3	V4	V5	



【山东师范大学 1999 六 （14 分）】

32. 已知个 n 顶点的有向图，用邻接矩阵表示，编写函数计算每对顶点的最短路径。

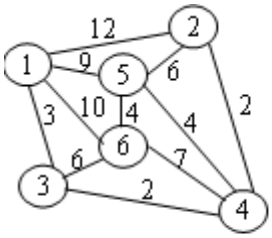
【南京航空航天大学 2001 九 （10 分）】

类似本题的另外叙述有：

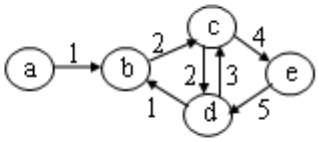
（1）假定有 n 个城市组成的一个公路网，且认为公路是有向的，并用代价邻接矩阵表示

该网络。试设计从指定城市 V_1 到其他城市的最短路径的算法。【西安电子科技大学 1996 三（10 分）】

33. 给定 n 个村庄之间的交通图，若村庄 i 和 j 之间有道路，则将顶点 i 和 j 用边连接，边上的 Wij 表示这条道路的长度，现在要从这 n 个村庄中选择一个村庄建一所医院，问这所医院应建在哪个村庄，才能使离医院最远的村庄到医院的路程最短?试设计一个解答上述问题的算法，并应用该算法解答如图所示的实例。【中国矿业大学 2000 十五 （15 分）】



第33题图



第34题图

34、求解下面有向图的有关问题：（1）判断此有向图是否有强连通分量？若有请画出；

（2）画出此有向图的十字链表存储结构；其顶点表结点为(data, firstin, firstout) ，其中 data 是 顶点的有关信息,firstin 是指向以该顶点为弧头的第一条边的指针,firstout 是指向以该顶点为弧尾的第一条边的指针。其表结点的结构为（tailvex ,headvex ,weight, hlink, tlink），其中 tailvex，headvex 分别为弧尾和弧头在图中的序号，weight 是弧上的权值，hlink，tlink 分别为指向弧头相同和弧尾相同的下一条边的指针。

（3）设其顶点 a, b, c, d, e 表示一个乡的 5 个村庄，弧上的权值表示为两村之间的距离；

① 求每个村庄到其它村庄的最短距离；

② 乡内要建立一所医院，问医院设在哪个村庄才能使各村离医院的距离较近。

【北京邮电大学 1997 五（15 分）】

35. 设计算法，求出无向连通图中距离顶点 V0 的最短路径长度（最短路径长度以边数为单位计算）为 K 的所有的结点，要求尽可能地节省时间。【西北大学 2001 七】

36. 自由树(即无环连通图)T=(V,E)的直径是树中所有点对间最短路径长度的最大值,即 T 的直径定义为 MAX D(u,v),这里 D(u,v) （u,v∈V）表示顶点 u 到顶点 v 的最短路径长度(路径长度为路径中所包含的边数)。写一算法求 T 的直径,并分析算法的时间复杂度。(时间复杂度越小得分越高)

【中科院 1999 五、3 （20 分）】

37. 求图的中心点的算法。设 V 是有向图 G 的一个顶点，我们把 V 的偏心度定义为：max{从 w 到 v 的最短距离|w 是 g 中所有顶点}，如果 v 是有向图 G 中具有最小偏心度的顶点，则称顶点 v 是 G 的中心点。

【长沙铁道学院 1998 五、2 (10 分)】

38. 设 G 是含有 n 顶点（设顶点编号为 1,2,…,n）的有向无环图。将 G 用如下定义的邻接表存储：

```
TYPE    arcptr=↑arcnode;
        arcnode=RECORD{邻接表中的结点}
                    adjvex:1..n;  nextarc:arcptr; END;
        vexnode=RECORD{邻接表的表头结点}
                    vexnum: 1..n;  firstarc:arcptr;  mpl:integer  END;
Hnodes=ARRAY[1..n] OF  vexnode;
```

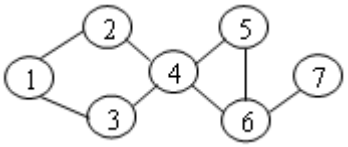
请编写一个非递归算法求 G 的每个顶点出发的最长路径的长度（每条弧的长度均为 1）并存

入 mpl 域中。 要求：首先写出算法思想，然后写算法过程。【山东科技大学 2001 六 （20 分）】

39. 图 G 有 n 个点，利用从某个源点到其余各点最短路径算法思想，设计一产生 G 的最小生成树的算法。

【东南大学 1994 四（18 分）】

40. 设 G 是一个用邻接表表示的连通无向图。对于 G 中某个顶点 v，若从 G 中删去顶点 v 及与顶点 v 相关联的边后，G 变成由两个或两个以上非空连通分量所组成的图，则称 v 是原来图 G 的一个关节顶点。如下图中，只有顶点 4 和顶点 6 是关节顶点，而其它顶点都不是关节顶点。试叙述寻找图 G 的所有关节顶点的算法，并用算法语言（PASCAL 或 C）编写一个实现你所给出的算法的程序。【复旦大学 1996 八 （20 分）】



41. 对于一个使用邻接表存储的有向图 G，可以利用深度优先遍历方法，对该图中结点进行拓扑排序。其基本思想是：在遍历过程中，每访问一个顶点，就将其邻接到的顶点的入度减一，并对其未访问的、入度为 0 的邻接到的顶点进行递归。

（1）. 给出完成上述功能的图的邻接表定义（结构）:（4 分）

（2）. 定义在算法中使用的全局辅助数组。（4 分）

（3）. 写出在遍历图的同时进行拓扑排序的算法:（10 分）

【东北大学 1999 五 (18 分)】 【清华大学 1997 一（18 分）】

42. 欲用四种颜色对地图上的国家涂色，有相邻边界的国家不能用同一种颜色（点相交不算相邻）。

（1）. 试用一种数据结构表示地图上各国相邻的关系，（ 6 分）。

（2）. 描述涂色过程的算法。（不要求证明）（ 1 2 分）。

【浙江大学 2002 八 （18 分）】

第 7 章 图（答案）

一. 选择题

1.A	2.B	3.A	4.B	5.D	6.1B	6.2D	7.1B	7.2C	8.A	9.A	10.1C
10.2BDE	11.B	12.1B	12.2B	12.3D	13.B	14.C	15.C	16.D	17.D	18.1C	18.2C
19.AB	20.B	21.1C	21.2A	21.3B	21.4A	22.A	23.A	24.D	25.A	26.A	27.B
28.D	29.B	30.A	31.C	32.B							

二. 判断题

1.√	2. ×	3.×	4.×	5.×	6.√	7.×	8.×	9.×	10.×	11.√	12.×
13.√	14.×	15.×	16.×	17.√	18.×	19.×	20.×	21.×	22.×	23.×	24.×
25.×	26.√	27.×	28.√	29.×	30.×	31.√	32.√	33.×	34.×	35.×	36.√
37.×	38.×	39.×	40.×	41.√	42.×	43.×	44.√	45.×	46.×	47.×	48. √
49.×											

部分答案解释如下。

2. 不一定是连通图，可能有若干连通分量

11. 对称矩阵可存储上（下）三角矩阵

14. 只有有向完全图的邻接矩阵是对称的

16. 邻接矩阵中元素值可以存储权值

21. 只有无向连通图才有生成树　　22. 最小生成树不唯一，但最小生成树上权值之和相等

26. 是自由树，即根结点不确定

35. 对有向无环图，拓扑排序成功；否则，图中有环，不能说算法不适合。

42. AOV 网是用顶点代表活动，弧表示活动间的优先关系的有向图,叫顶点表示活动的网。

45. 能求出关键路径的 AOE 网一定是有向无环图

46. 只有该关键活动为各关键路径所共有，且减少它尚不能改变关键路径的前提下，才可缩短工期。

48. 按着定义，AOE 网中关键路径是从“源点”到“汇点”路径长度最长的路径。自然，关键路径上活动的时间延长多少，整个工程的时间也就随之延长多少。

三．填空题

1.有 n 个顶点，n-1 条边的无向连通图　　2.有向图的极大强连通子图　　3. 生成树

4. 45　　5. n(n-1)/2　　6 .　　7. 9　　8. n

9. 2(n-1)　　10. N-1　　11. n-1　　12. n　　13. N-1　　14. n

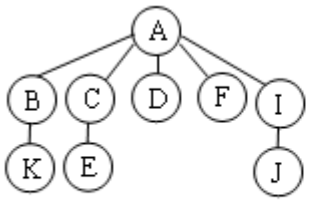
15. N

16. 3　　17. 2（N-1）　　18. 度　出度　　19. 第 I 列非零元素个数　　20.n　2e

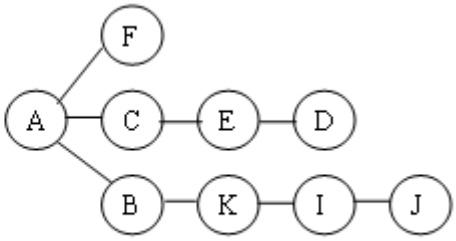
21.(1)查找顶点的邻接点的过程　　(2)O(n+e)　　(3)O(n+e)　　(4)访问顶点的顺序不同　(5)队列和栈

22. 深度优先　　23.宽度优先遍历　　24.队列

25.因未给出存储结构，答案不唯一。本题按邻接表存储结构，邻接点按字典序排列。



25题（1）



25题（2）

26.普里姆（prim）算法和克鲁斯卡尔（Kruskal）算法　　27.克鲁斯卡尔

28.边稠密　边稀疏　　29. O(eloge)　边稀疏　　30.O(n^2)　O(eloge)

31.(1)（V_i，V_j）边上的权值　都大的数　（2）1　负值　（3）为负　边

32.(1)n-1　(2)普里姆　(3)最小生成树　　33.不存在环　　34.递增　负值

35.160

36.O(n^2)　　37. 50，经过中间顶点④　　38. 75　　39.O(n+e)

40.（1）活动　（2）活动间的优先关系　（3）事件　（4）活动　边上的权代表活动持续时间

41.关键路径　　42.（1）某项活动以自己为先决条件　（2）荒谬　（3）死循环

43.（1）零　（2）V_k度减 1，若 V_k 入度已减到零，则 V_k 顶点入栈　（3）环

44. （ 1 ） p<>nil　　(2)visited[v]=true　　(3)p=g[v].firstarc　(4)p=p^.nextarc

45.(1)g[0].vexdata=v　(2)g[j].firstin　(3)g[j].firstin　(4)g[i].firstout

(5)g[i].firstout　(6)p^.vexj　(7)g[i].firstout　(8)p:=p^.nexti
(9)p<>nil　(10)p^.vexj=j
(11)firstadj(g,v_0) (12)not visited[w] (13)nextadj(g,v_0,w)

46.(1)0　(2)j　(3)i　(4)0　(5)indegree[i]==0　(6)[vex][i]　(7)k==1
(8)indegree[i]==0

47.(1)p^.link:=ch[u].head　(2)ch[u].head:=p　(3)top<>0　(4)j:=top
(5)top:=ch[j].count
(6)t:=t^.link

48.（1）V1 V4 V3 V6 V2 V5(尽管图以邻接表为存储结构，但因没规定邻接点的排列，所以结果是不唯一的。本答案是按邻接点升序排列给出的。)
（2） ① top==-1　② top=graph[j].count　③ graph[k].count==0

四.应用题

1.（1）G1 最多 n(n-1)/2 条边，最少 n-1 条边　(2) G2 最多 n(n-1)条边，最少 n 条边
(3) G3 最多 n(n-1)条边，最少 n-1 条边 (注：弱连通有向图指把有向图看作无向图时，仍是连通的)　2．n-1，n　3.分块对称矩阵

4．证明：具有 n 个顶点 n-1 条边的无向连通图是自由树，即没有确定根结点的树，每个结点均可当根。若边数多于 n-1 条，因一条边要连接两个结点，则必因加上这一条边而使两个结点多了一条通路，即形成回路。形成回路的连通图不再是树（在图论中树定义为无回路的连通图)。

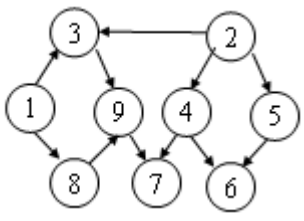
5．证明：该有向图顶点编号的规律是让弧尾顶点的编号大于弧头顶点的编号。由于不允许从某顶点发出并回到自身顶点的弧，所以邻接矩阵主对角元素均为 0。先证明该命题的充分条件。由于弧尾顶点的编号均大于弧头顶点的编号，在邻接矩阵中，非零元素（A[i][j]=1）自然是落到下三角矩阵中；命题的必要条件是要使上三角为 0，则不允许出现弧头顶点编号大于弧尾顶点编号的弧，否则，就必然存在环路。(对该类有向无环图顶点编号，应按顶点出度顺序编号。)

6．设图的顶点个数为 n(n≥0)，则邻接矩阵元素个数为 n^2，即顶点个数的平方，与图的边数无关。

7.（1）n(n-1)，　n
(2) 10^6，不一定是稀疏矩阵（稀疏矩阵的定义是非零个数远小于该矩阵元素个数，且分布无规律）

(3)使用深度优先遍历，按退出 dfs 过程的先后顺序记录下的顶点是逆向拓扑有序序列。若在执行 dfs(v)未退出前，出现顶点 u 到 v 的回边，则说明存在包含顶点 v 和顶点 u 的环。

8. (1)



2）开始结点：（入度为 0）K_1，K_2，终端结点（出度为 0）K_6，K_7。

（3）拓扑序列 K_1，K_2，K_3，K_4，K_5，K_6，K_8，K_9，K_7
K_2，K_1，K_3，K_4，K_5，K_6，K_8，K_9，K_7
规则：开始结点为 K_1 或 K_2，之后，若遇多个入度为 0 的顶点，按顶点编号顺序选择。

（4）

8（4）邻接表和逆邻接表

9.（1）

序:abcdefghi

注：邻接矩阵下标按字母升

（2）强连通分量：(a)，(d)，(h)，(b，e，i，f，c，g)

（3）顶点 a 到顶点 i 的简单路径：(a→b→e→i)，(a→c→g→i)，(a→c→b→e→i)

10. 图 G 的具体存储结构略。

邻接矩阵表示法，有 n 个顶点的图占用 n^2 个元素的存储单元，与边的个数无关，当边数较少时，存储效率较低。这种结构下，对查找结点的度、第一邻接点和下一邻接点、两结点间是否有边的操作有利，对插入和删除顶点的操作不利。

邻接表表示法是顶点的向量结构与顶点的邻接点的链式存储结构相结合的结构，顶点的向量结构含有 n（n≥0）个顶点和指向各顶点第一邻接点的指针，其顶点的邻接点的链式存储结构是根据顶点的邻接点的实际设计的。这种结构适合查找顶点及邻接点的信息，查顶点的度，增加或删除顶点和边（弧）也很方便，但因指针多占用了存储空间，另外，某两顶点间是否有边（弧）也不如邻接矩阵那么清楚。对有向图的邻接表，查顶点出度容易，而查顶点入度却困难，要遍历整个邻接表。要想查入度象查出度那样容易，就要建立逆邻接表。无向图邻接表中边结点是边数的二倍也增加了存储量。

十字链表是有向图的另一种存储结构，将邻接表和逆邻接表结合到一起，弧结点也增加了信息（至少弧尾，弧头顶点在向量中的下标及从弧尾顶点发出及再入到弧头顶点的下一条弧的四个信息）。查询顶点的出度、入度、邻接点等信息非常方便。

邻接多重表是无向图的另一种存储结构，边结点至少包括 5 个域：连接边的两个顶点在顶点向量中的下标，指向与该边相连接的两顶点的下一条边的指针，以及该边的标记信息（如该边是否被访问）。边结点的个数与边的个数相同，这是邻接多重表比邻接表优越之处。

11.

已知顶点 i，找与 i 相邻的顶点 j 的规则如下：在顶点向量中，找到顶点 i，顺其指针找到第一个边结点（若其指针为空，则顶点 i 无邻接点）。在边结点中，取出两顶点信息，若其中有 j，则找到顶点 j；否则，沿从 i 发出的另一条边的指针（ilink）找 i 的下一邻接点。在这种查找过程中，若边结点中有 j，则查找成功；若最后 ilink 为空，，则顶点 i 无邻接点 j。

12. 按各顶点的出度进行排序。n 个顶点的有向图，其顶点最大出度是 n-1，最小出度为 0。这样排序后，出度最大的顶点编号为 1，出度最小的顶点编号为 n。之后，进行调整，即若存在弧<i，j>，而顶点 j 的出度大于顶点 i 的出度，则将把 j 编号在顶点 i 的编号之前。本题算法见下面算法设计第 28 题。

13. 采用深度优先遍历算法，在执行 dfs(v)时，若在退出 dfs(v)前，碰到某顶点 u ，其邻接点是已经访问的顶点 v，则说明 v 的子孙 u 有到 v 的回边，即说明有环，否则，无环。（详见下面算法题 13）

14. 深度优先遍历序列：125967384

宽度优先遍历序列：123456789

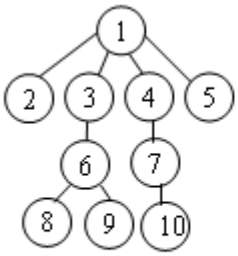
注：（1）邻接表不唯一，这里顶点的邻接点按升序排列

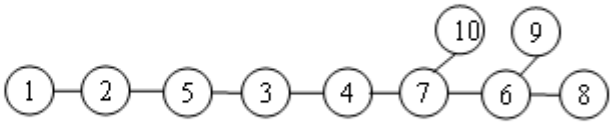
（2）在邻接表确定后，深度优先和宽度优先遍历序列唯一

（3）这里的遍历，均从顶点 1 开始

15.（1）$V_1V_2V_4V_3V_5V_6$　　（2）$V_1V_2V_3V_4V_5V_6$

16.(1)





16题（3）宽度优先生成树

（2）深度优先生成树

为节省篇幅，生成树横画，下同。

17. 设从顶点 1 开始遍历，则深度优先生成树(1)和宽度优先生成树(2)为

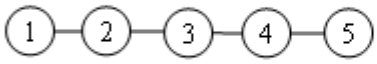


图17(1)

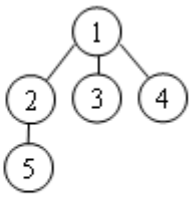


图17(2)

18．遍历不唯一的因素有：开始遍历的顶点不同；存储结构不同；在邻接表情况下邻接点的顺序不同。

19．（1）邻接矩阵：（6*6 个元素）*2 字节/元素=72 字节

邻接表：表头向量 6*（4+2）+边结点 9*(2+2+4)*2=180 字节

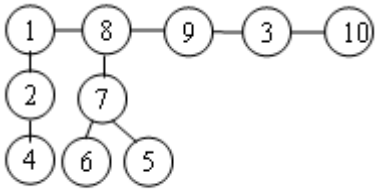
邻接多重表：表头向量 6*（4+2）+边结点 9*（2+2+2+4+4）=162 字节

邻接表占用空间较多，因为边较多，边结点又是边数的 2 倍，一般来说，邻接矩阵所占空间与边个数无关（不考虑压缩存储），适合存储稠密图，而邻接表适合存储稀疏图。邻接多重表边结点个数等于边数，但结点中增加了一个顶点下标域和一个指针域。

（2）因未确定存储结构，从顶点 1 开始的 DFS 树

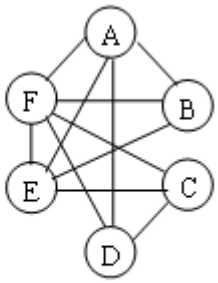
不唯一，现列出两个：

20．未确定存储结构，其 DFS 树不唯一，其中之一（按邻接点逆序排列）是



（2）关节点有3，1，8，7，2

21．（1）



（2）AFEDBC

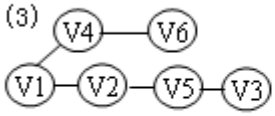
22．设邻接表（略）中顶点的邻接点按顶点编号升序排列（V1 编号为 1）

（1）　　广度优先搜索序列：V1V2V3V4V5V6V7V8

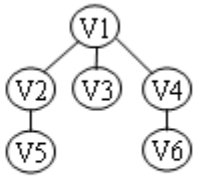
（2）　　深度优先搜索序列：V1V2V4V8V5V3V6V7

23．（1）略　　（2）V1V2V5V3V4V6　　(4) V1V2V3V4V5V6

(3)



(5)

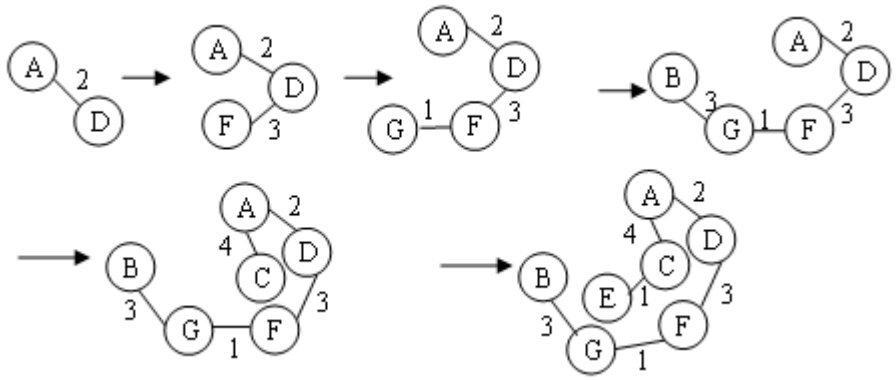


(6) 见本章五算法设计第6题

24. 设该图用邻接表存储结构存储，顶点的邻接点按顶点编号升序排列

(1)ABGFDEC　　(2)EACFBDG

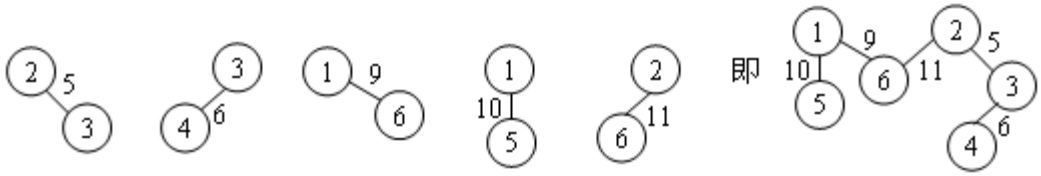
(3)



25. 在有相同权值边时生成不同的 MST，在这种情况下，用 Prim 或 Kruskal 也会生成不同的 MST。

26. 无向连通图的生成树包含图中全部 n 个顶点，以及足以使图连通的 n-1 条边。而最小生成树则是各边权值之和最小的生成树。从算法中 WHILE（所剩边数>=顶点数）来看，循环到边数比顶点数少 1（即 n-1）停止，这符合 n 个顶点的连通图的生成树有 n-1 条边的定义；由于边是按权值从大到小排序，删去的边是权值大的边，结果的生成树必是最小生成树；算法中“若图不再连通，则恢复 ei”，含义是必须保留使图连通的边，这就保证了是生成树，否则或者是有回路，或者成了连通分量，均不再是生成树。

27. Prim 算法构造最小生成树的步骤如 24 题所示，为节省篇幅，这里仅用 Kruskal 算法，构造最小生成树过程如下：（下图也可选(2,4)代替(3,4)，(5,6)代替(1,5)）



28.（1）最小生成树的定义见上面 26 题

（2）最小生成树有两棵。

（限于篇幅，下面的生成树只给出顶点集合和边集合，边以三元组（Vi,Vj,W）形式），其中 W 代表权值。

V（G）={1，2，3，4，5}　　E1(G)={(4，5，2)，(2，5，4)，(2，3，5)，(1，2，7)}；

E2(G)={(4，5，2)，(2，4，4)，(2，3，5)，(1，2，7)}

29. V（G）={1，2，3，4，5，6，7}

E（G）={（1，6，4)，(1，7，6)，(2，3，5)，(2，4，8)，(2，5，12)，(1，2，18)}

30. V（G）={1，2，3，4，5，6，7，8}

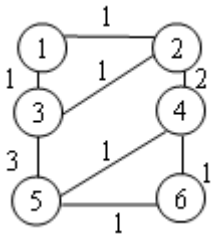
E（G）={（3，8，2），（4，7，2），（3，4，3），（5，8，3），（2，5，4），（6，7，4），（1，2，5）}

注：（或将（3，4，3）换成（7，8，3））

31. 设顶点集合为{1，2，3，4，5，6}，

由右边的逻辑图可以看出，在{1，2，3}和{4，5，6}回路中，

各任选两条边，加上（2，4），则可构成 9 棵不同的最小生成树。

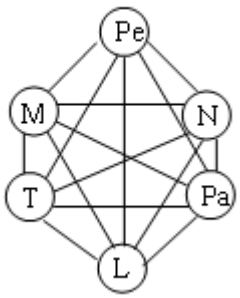


32.（1）邻接矩阵略

（2）

Y Closedge	2	3	4	5	6	7	8	U	V-U
Vex Lowcost	① 2	① 3						{1}	{2,3,4,5,6,7,8}
Vex Lowcost		① 3	② 2					{1,2}	{3,4,5,6,7,8}
Vex Lowcost		④ 1		④ 2	④ 4			{1,2,4}	{3,5,6,7,8}
Vex Lowcost				④ 2	④ 4			{1,2,4,3}	{5,6,7,8}
Vex Lowcost					⑤ 1	⑤ 2		{1,2,4,3,5}	{6,7,8}
Vex Lowcost						⑤ 2	⑥ 4	{1,2,4,3,5,6}	{7,8}
Vex Lowcost							⑦ 3	{1,2,4,3,5,6,7}	{8}
Vex Lowcost								{1,2,4,3,5,6,7,8}	{}

33.(1)　　(2)



(3)最小生成树 6 个顶点 5 条边：V(G)={Pe,N,Pa,L,T,M}

E(G)={(L,Pa,3),(Pe,T,21),(M,N,32),(L,N,55),(L,Pe,81)}

34.(1)

TYPE　edgeptr=^enode;

enode=RECORD ivex,jvex:vtxptr; w:integer; jlink,ilink:edgeptr; END ;

vexnode=RECORD data:vertertype; firstedge:edgeptr; END;

adjmulist=ARRAY[vexptr] OF vexnode;

(2)深度优先遍历序列：1,4,6,5,3,2;

深度优先生成树的边集合：{(1,4),(4,6),(6,5),(5,3),(3,2)}

(3)宽度优先遍历序列：143265;

宽度优先生成树的边集合：{(1,4),(1,3),(1,2),(4,6),(4,5)}：

（4 按 Prim 构造过程只画出最小生成树的边和权植：

E(G)={(1,4,3),(4,3,4),(3,5,1),(3,2,2),(3,6,10)}

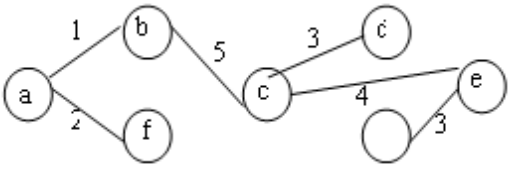
35. E1(G)={(1，2，16)，(2，3，5)，(2，6，6)，(2，4，11)，(6，5，18)}，V(G)={1,2,3,4,5,6}

E2(G)={（1，2，16)，(2，3，5)，(3，6，6)，(2，4，11)，(6，5，18）}

36（1）邻接矩阵和邻接表

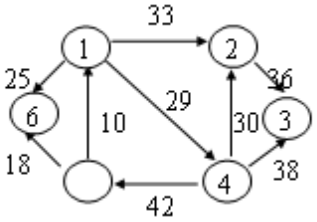
36（2）深度优先搜索序列:abcdehf;　36(3) 最小生成树

宽度优先搜索序列: abfcdef

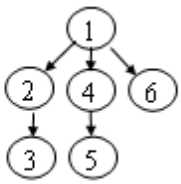


37. A^0= A^1= A^2= A^3= A^4=

38. (1)



(2)V1,V2,V4,V6,V3,V5



(3) 顶点集合 V(G)={V1,V2,V3,V4,V5,V6}

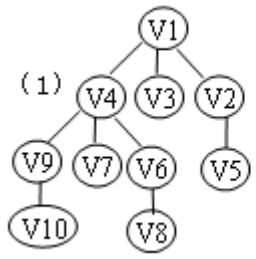
边的集合 E(G)={(V1,V2),(V2,V3),(V1,V4),(V4,V5),(V5,V6)}

(4) V1 到 V3 最短路径为 67: (V1—V4—V3)

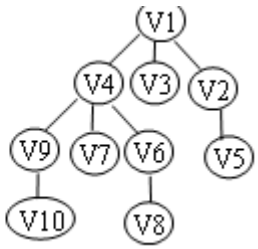
所选顶点	S（已确定最短路径的顶点集合）	T(尚未确定最短路径的顶点集合)	DIST								
			b	c	d	e	f	g	h	i	j	z
初态	{a}	{b,c,d,e,f,g,h,i,j,z}	3	2	1	∞	∞	∞	∞	∞	∞	∞
d	{a,d}	{b,c,d,e,f,g,h,i,j,z}	3	2		∞	∞	4	∞	∞	∞	∞
c	{a,d,c}	{b,e,f,g,h,i,j,z}	3			∞	6	4	∞	∞	∞	∞
b	{a,d,c,b}	{e,f,g,h,i,j,z}				9	6	4	∞	∞	∞	∞
g	{a,d,c,b,g}	{e,f,h,i,j,z}				9	5		∞	10	13	∞
f	{a,d,c,b,g,f}	{e,h,i,j,z}				9			14	7	13	∞
i	{a,d,c,b,g,f,i}	{e,h,j,z}				9			14		11	16
e	{a,d,c,b,g,f,i,e}	{h,j,z}							14		11	16
j	{a,d,c,b,g,f,i,e,j}	{h,z}							14			13
z	{a,d,c,b,g,f,i,e,j,z}	{h}							14			
h	{a,d,c,b,g,f,i,e,j,z,h}	{}										

39．注：(1).本表中 DIST 中各列最下方的数字是顶点 a 到顶点的最短通路，a 到 z 为 13。
(2)．DIST 数组的常量表达式（即下标）应为符号常量，即‘b’，‘c’等，为节省篇幅，用了 b,c 等。
(3)．最短路径问题属于有向图，本题给出无向图，只能按有向完全图理解。
40．下面用 FLOYD 算法求出任意两顶点的最短路径（如图 A[6] 所示）。题目要求娱乐中心“距其它各结点的最长往返路程最短”，结点 V1，V3，V5 和 V6 最长往返路径最短都是 9。按着“相同条件下总的往返路径越短越好”，选顶点 V5，总的往返路径是 34。
41．顶点 1 到其它顶点的最短路径依次是 20，31，28，10，17，25，35。按 Dijkstra 算法所选顶点依次是 5，6，2，7，4，3，8，其步骤原理请参见第 39 题，因篇幅所限，不再画出各步状态。
42．顶点 a 到顶点 b，c，d，e，f，g 间的最短路径分别是 15，2，11，10，6，13。
43．顶点 A 到顶点 B，C，D，E 的最短路径依次是 3，18，38，43，
按 Dijkstra 所选顶点过程是 B,C,D,E。支撑树的边集合为{<A,B>,<B,C>,<C,D>,<B,E>}

44．(1)

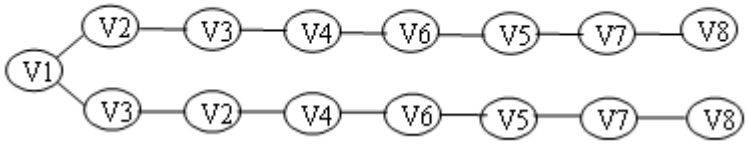


(2)



(3)V1,V2,V5,V3,V4,V6,V8,V7,V9,V10
设一栈，将入度为零的顶点放入栈中

45．（1）略
（2）



3）关键路径有 3 条，长 17。（各事件允许发生的最早时间和最晚时间略）
V1->V2->V4->V6->V8，V1->V3->V5->V7->V8，V1->V2->V4->V6->V5->V7->V8
（4）V1 结点到其它各结点的最短距离为：2，3，6，12，10，15，16。
46．（1）对有向图，求拓扑序列步骤为：
1）在有向图中选一个没有前驱（即入度为零）的顶点并输出。
2）在图中删除该顶点及所有以它为尾的弧。
3）重复 1）和 2），直至全部顶点输出，这时拓扑排序完成；否则，图中存在环，拓扑排序失败。
（2）这里使用形式化描述方法，当有多个顶点可以输出时，将其按序从上往下排列，这样不会丢掉一种拓扑序列。这里只画出从顶点 1 开始的所有可能的拓扑序列，从顶点 3 开始的拓扑序列可类似画出

1	2	3	4	5	6	7	8
					7	6	8
						8	6
				7	5	6	8
						8	6
					8	5	6
			5	4	6	7	8
					7	6	8
						8	6
	3	2	4	5	6	7	8
					7	6	8
						8	6

1 3	2	4 7	5	6	8
				8	6
			8	5	6
		5 4	6	7	8
			7	6	8
				8	6
	4 2	5	6	7	8
			7	6	8
				8	6
		7	5	6	8
				8	6
			8	5	6

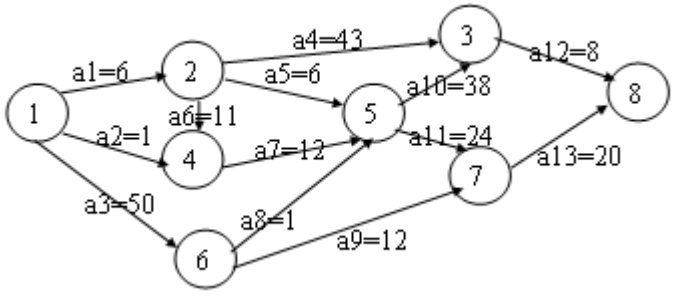
1 3	4	5	2	6	7	8
				7	6	8
					8	6
			6	2	7	8
	5	2 4		6	7	8
				7	6	8
					8	6
		4	2	6	7	8
				7	6	8
					8	6
			6	2	7	8

47. 图的深度优先遍历可用于拓扑排序。带环的有向图不能拓扑排序。深度优先遍历如何发现环呢？若从有向图某顶点 V 出发进行遍历，在 dfs(v)结束之前出现从顶点 W 到顶点 V 的回边，由于 W 在生成树上是 V 的子孙，则有向图中必定存在包含顶点 V 和 W 的环。其算法实现见下面算法题第 41 题。

48.（1）V1,V2,V3,V8,V5,V7,V4,V6　　（2）V1,V2,V4,V6,V3,V5,V7,V8

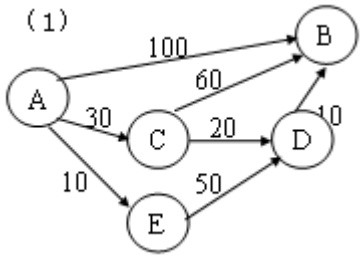
（3）V1 到 V8 最短路径 56，路径为 V1----V2----V5----V7----V8

（4）



V1到V8的关键路径是V1----V6----V5----V3----V8，长97。

49. （1）

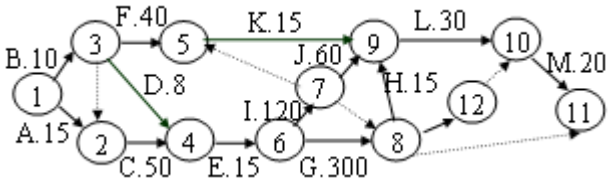


（2）略

（3）深度优先遍历序列：ABCDE　　广度优先遍历序列：ABCED　　（4）关键路径 A--B（长 100）

50．AOE 网的始点是入度为零的顶点，该顶点所代表的事件无任何先决条件，可首先开始。终点是出度为零的顶点，表示一项工程的结束。正常的 AOE 网只有一个始点和一个终点。

51．（1）AOE 网如右图



图中虚线表示在时间上前后工序之间仅是接续顺序关系不存在依赖关系。顶点代表事件，弧代表活动，弧上的权代表活动持续时间。题中顶点 1 代表工程开始事件，顶点 11 代表工程结束事件。

（2）各事件发生的最早和最晚时间如下表

事　　件	1	2	3	4	5	6	7	8	9	10	11	12
最早发生时间	0	15	10	65	50	80	200	380	395	425	445	420
最晚发生时间	0	15	57	65	380	80	335	380	395	425	445	425

（3）关键路径为顶点序列：1->2->4->6->8->9->10->11；

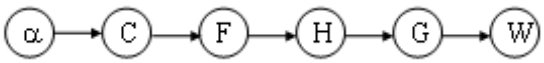
事件序列：A->C->E->G->H->L->M，完成工程所需的最短时间为 445。

52

顶点	α	A	B	C	D	E	F	G	H	W
Ve(i)	0	1	6	3	4	24	13	39	22	52
Vl(i)	0	29	24	3	7	31	13	39	22	52

活动	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10	a11	a12	a13	a14	a15	a16	a17
e(i)	0	0	0	0	1	6	6	3	3	4	24	13	13	13	39	22	22
l(i)	28	18	0	3	29	24	31	34	3	7	31	20	36	13	39	22	40

关键路径是：



活动与顶点的对照表：a1<α,A> a2<α,B> a3<α,C> a4<α,D> a5<A,E> a6<B,E> a7<B,W> a8<C,G>

a9<C,F> a10<D,F> a11<E,G> a12<F,E> a13<F,W> a14<F,H> a15<G,W> a16<H,G> a17<H,W>

53. A.0—2—6—9—11　　B.20　　C.1，5，7　　D.各 2 天　　E.0

五.算法设计题

1.
```
void CreatGraph (AdjList g)
    //建立有 n 个顶点和 m 条边的无向图的邻接表存储结构
{int n,m;
 scanf("%d%d",&n,&m);
 for (i =1,i<=n;i++)//输入顶点信息,建立顶点向量
   {scanf(&g[i].vertex);  g[i].firstarc=null;}
 for (k=1;k<=m;k++)//输入边信息
   {scanf(&v1,&v2);//输入两个顶点
    i=GraphLocateVertex (g,v1); j=GraphLocateVertex (g,v2); //顶点定位
    p=(ArcNode *)malloc(sizeof(ArcNode));//申请边结点
    p->adjvex=j; p->next=g[i].firstarc; g[i].firstarc=p;//将边结点链入
    p=(ArcNode *)malloc(sizeof(ArcNode));
    p->adjvex=i;  p->next=g[j].firstarc;  g[j].frstarc=p;
   }
}//算法 CreatGraph 结束
```

2.
```
void CreatAdjList(AdjList g)
    //建立有向图的邻接表存储结构
    {int n;
     scanf("%d",&n);
     for (i=1;i<=n;j++)
       {scanf(&g[i].vertex); g[i].firstarc=null;}//输入顶点信息
       scanf(&v1,.&v2);
       while(v1 && v2)//题目要求两顶点之一为 0 表示结束
        {i=GraphLocateVertex(g2,v1);
         p=(ArcNode*)malloc(sizeof(ArcNode));
         p->adjvex=j;  p->next=g[i].firstarc;  g[i].firstarc=p;
        scanf(&v1,&v2);
        } }
```

3.
```
void CreatMGraph(AdjMulist g)
    //建立有 n 个顶点 e 条边的无向图的邻接多重表的存储结构
    {int n,e;
     scanf("%d%d",&n,&e);
     for(i=1,i<=n;i++)     //建立顶点向量
      { scanf(&g[i].vertex);  g[i].firstedge=null;}
     for(k=1;k<=e;k++)     //建立边结点
      {scanf(&v1,&v2);
       i=GraphLocateVertex(g,v1); j=GraphLocateVertex(g,v2);
       p=(ENode *)malloc(sizeof(ENode));
       p->ivex=i; p->jvex=j; p->ilink=g[i].firstedge; p->jlink=g[j].firstedge;
       g[i].firstedge=p; g[j].firstedge=p;
      }
    }//算法结束
```

4. void CreatOrthList(OrthList g)
//建立有向图的十字链表存储结构
{int i,j,v; //假定权值为整型
scanf("%d",&n);
for(i=1,i<=n;i++) //建立顶点向量
{ scanf(&g[i].vertex); g[i].firstin=null; g[i].firstout=null;}
scanf("%d%d%d",&i,&j,&v);
while (i && j && v) //当输入 i,j,v 之一为 0 时,结束算法运行
{p=(OrArcNode *)malloc(sizeof(OrArcNode)); //申请结点
p->headvex=j; p->tailvex=i; p->weight=v; //弧结点中权值域
p->headlink=g[j].firstin; g[j].firstin=p;
p->tailink=g[i].firstout; g[i].firstout=p;
scanf("%d%d%d",&i,&j,&v);
} }算法结束

[算法讨论] 本题已假定输入的 i 和 j 是顶点号,否则,顶点的信息要输入,且用顶点定位函数求出顶点在顶点向量中的下标。图建立时，若已知边数（如上面 1 和 2 题），可以用 for 循环；若不知边数，可用 while 循环（如本题），规定输入特殊数（如本题的零值）时结束运行。本题中数值设为整型，否则应以和数值类型相容的方式输入。

5. void InvertAdjList(AdjList gin,gout)
//将有向图的出度邻接表改为按入度建立的逆邻接表
{for (i=1;i<=n;i++)//设有向图有 n 个顶点，建逆邻接表的顶点向量。
{gin[i].vertex=gout[i].vertex; gin.firstarc=null; }
for (i=1;i<=n;i++) //邻接表转为逆邻接表。
{p=gout[i].firstarc;//取指向邻接表的指针。
while (p!=null)
{ j=p->adjvex;
s=(ArcNode *)malloc(sizeof(ArcNode));//申请结点空间。
s->adjvex=i; s->next=gin[j].firstarc; gin[j].firstarc=s;
p=p->next;//下一个邻接点。
}//while
}//for }

6. void AdjListToAdjMatrix(AdjList gl, AdjMatrix gm)
//将图的邻接表表示转换为邻接矩阵表示。
{for (i=1;i<=n;i++) //设图有 n 个顶点，邻接矩阵初始化。
for (j=1;j<=n;j++) gm[i][j]=0;
for (i=1;i<=n;i++)
{p=gl[i].firstarc; //取第一个邻接点。
while (p!=null) {gm[i][p->adjvex]=1;p=p->next; }//下一个邻接点
}//for }//算法结束

7. void AdjMatrixToAdjList(AdjMatrix gm, AdjList gl)
//将图的邻接矩阵表示法转换为邻接表表示法。
{for (i=1;i<=n;i++) //邻接表表头向量初始化。
{scanf(&gl[i].vertex); gl[i].firstarc=null;}
for (i=1;i<=n;i++)

```
        for (j=1;j<=n;j++)
           if (gm[i][j]==1)
              {p=(ArcNode *)malloc(sizeof(ArcNode)) ;//申请结点空间。
                p->adjvex=j;//顶点 I 的邻接点是 j
                p->next=gl[i].firstarc; gl[i].firstarc=p; //链入顶点 i 的邻接点链表中
                }
        }//end
```

[算法讨论] 算法中邻接表中顶点在向量表中的下标与其在邻接矩阵中的行号相同。

8.[题目分析] 在有向图中，判断顶点 Vi 和顶点 Vj 间是否有路径，可采用遍历的方法，从顶点 Vi 出发，不论是深度优先遍历（dfs）还是宽度优先遍历（bfs），在未退出 dfs 或 bfs 前,若访问到 Vj，则说明有通路，否则无通路。设一全程变量 flag。初始化为 0，若有通路，则 flag=1。

算法 1：

```
int visited[]=0;  //全局变量，访问数组初始化
    int  dfs(AdjList g , vi)
    //以邻接表为存储结构的有向图 g，判断顶点 Vi 到 Vj 是否有通路,返回 1 或 0 表示有或无
   { visited[vi]=1;   //visited 是访问数组，设顶点的信息就是顶点编号。
    p=g[vi].firstarc;  //第一个邻接点。
    while ( p!=null)
       { j=p->adjvex;
        if (vj==j) { flag=1; return（1）;} //vi 和 vj 有通路。
        if (visited[j]==0) dfs(g,j);
        p=p->next;  }//while
     if (!flag) return(0);
  }//结束
```

[算法讨论] 若顶点 vi 和 vj 不是编号，必须先用顶点定位函数,查出其在邻接表顶点向量中的下标 i 和 j。下面算法 2 输出 vi 到 vj 的路径，其思想是用一个栈存放遍历的顶点，遇到顶点 vj 时输出路径。

算法 2：

```
void dfs(AdjList g , int i)
        //有向图 g 的顶点 vi（编号 i）和顶点 vj（编号 j）间是否有路径，如有，则输出。
   {int top=0,stack[]; //stack 是存放顶点编号的栈
    visited[i]=1;       //visited 数组在进入 dfs 前已初始化。
    stack[++top]=i;
    p=g[i].firstarc; /求第一个邻接点.
    while (p)
      {if (p->adjvex==j)
         {stack[++top]=j; printf( "顶点 vi 和 vj 的路径为：\n");
          for (i=1; i<=top; i++)  printf( "%4d",stack[i]); exit(0);
          }//if
       else     if     (visited[p->adjvex]==0)     {dfs(g,g->adjvex);     top--;
         p=p->next;}//else if
      }//while
    }//结束算法 2
```

算法 3：本题用非递归算法求解。

```
  int  Connectij (AdjList g , vertype vi , vj )
 //判断 n 个顶点以邻接表表示的有向图 g 中，顶点 Vi 各 Vj 是否有路径，有则返回 1，否则返回 0。
{ for (i=1;i<=n;i++)  visited[i]=0; //访问标记数组初始化。
  i=GraphLocateVertex(g,vi); //顶点定位，不考虑 vi 或 vj 不在图中的情况。
  j=GraphLocateVertex(g,vj);
  int stack[],top=0;stack[++top]=i;
  while(top>0)
   {k=stack[top--]; p=g[k].firstarc;
    while(p!=null && visited[p->adjvex]==1) p=p->next; //查第 k 个链表中第一个未访问的弧结点。
    if(p==null) top--;
    else {i=p->adjvex;
          if(i==j) return(1); //顶点 vi 和 vj 间有路径。
          else {visited[i]=1; stack[++top]=i;}//else
         }//else
    }while
    return(0); } //顶点 vi 和 vj 间无通路。
```

9.
```
 void DeletEdge(AdjList g,int i,j)
  //在用邻接表方式存储的无向图 g 中，删除边(i,j)
  {p=g[i].firstarc;  pre=null; //删顶点 i 的边结点(i,j),pre 是前驱指针
   while (p)
    if (p->adjvex==j)
      {if(pre==null)g[i].firstarc=p->next;else pre->next=p->next;free(p);}//释放结点空间。
     else {pre=p; p=p->next;}   //沿链表继续查找
   p=g[j].firstarc;  pre=null; //删顶点 j 的边结点(j,i)
   while (p)
    if (p->adjvex==i)
      {if(pre==null)g[j].firstarc=p->next;else pre->next=p->next;free(p);}//释放结点空间。
     else {pre=p; p=p->next;}   //沿链表继续查找
  }// DeletEdge
```

[算法讨论] 算法中假定给的 i，j 均存在，否则应检查其合法性。若未给顶点编号，而给出顶点信息，则先用顶点定位函数求出其在邻接表顶点向量中的下标 i 和 j。

10.
```
 void  DeleteArc (AdjList g,vertype vi,vj)
 //删除以邻接表存储的有向图 g 的一条弧<vi,vj>，假定顶点 vi 和 vj 存在
  {i=GraphLocateVertex(g,vi); j=GraphLocateVertex(g,vj); //顶点定位
   p=g[i].firstarc;  pre=null;
   while (p)
   if (p->adjvex==j)
     {if(pre==null) g[i].firstarc=p->next;else pre->next=p->next;free(p);}//释放结点空间。
```

```
      else  { pre=p; p=p->next;}
    }//结束
```

11.
```
void InsertArc ( OrthList g ,vertype vi,vj)
  //在以十字链表示的有向图 g 中插入弧<vi,vj>
    { i=GraphLocateVertex(g,vi); //顶点定位.
      j=GraphLocateVertex(g,vj);
      p=(OrArcNode *)malloc(sizeof(OrArcNode));
      p=headvex=j; p=tailvex=i;  //填写弧结点信息并插入十字链表。
      p->headlink=g[j].firstin; g[j].firstin=p;
      p->taillink=g[i].firstout; g[i].firstout=p;
    }//算法结束
```

12. [题目分析]在有向图的邻接表中，求顶点的出度容易，只要简单在该顶点的邻接点链表中查结点个数即可。而求顶点的入度，则要遍历整个邻接表。

```
int count (AdjList g , int k )
    //在 n 个顶点以邻接表表示的有向图 g 中，求指定顶点 k(1<=k<=n)的入度。
    { int count =0;
     for (i=1;i<=n;i++)    //求顶点 k 的入度要遍历整个邻接表。
       if(i!=k)            //顶点 k 的邻接链表不必计算
         {p=g[i].firstarc;//取顶点 i 的邻接表。
          while (p)
             {if (p->adjvex==k) count++;
              p=p->next;
             }//while
         }//if
      return(count); //顶点 k 的入度.
    }
```

13. [题目分析]有向图判断回路要比无向图复杂。利用深度优先遍历，将顶点分成三类：未访问；已访问但其邻接点未访问完;已访问且其邻接点已访问完。下面用 0，1，2 表示这三种状态。前面已提到，若 dfs（v）结束前出现顶点 u 到 v 的回边，则图中必有包含顶点 v 和 u 的回路。对应程序中 v 的状态为 1，而 u 是正访问的顶点，若我们找出 u 的下一邻接点的状态为 1，就可以输出回路了。

```
  void  Print(int v,int start )  //输出从顶点 start 开始的回路。
    {for(i=1;i<=n;i++)
      if(g[v][i]!=0 && visited[i]==1 )  //若存在边（v,i），且顶点 i 的状态为 1。
        {printf( " %d " ,v);  if(i==start)  printf( " \n " );  else
Print(i,start);break;}//if
    }//Print
  void dfs(int v)
   {visited[v]=1;
    for(j=1;j<=n;j++ )
       if (g[v][j]!=0) //存在边(v,j)
         if (visited[j]!=1) {if (!visited[j]) dfs(j); }//if
         else {cycle=1; Print(j,j);}
    visited[v]=2;
```

```
  }//dfs
 void find_cycle() //判断是否有回路，有则输出邻接矩阵。visited 数组为全局变量。
   {for (i=1;i<=n;i++) visited[i]=0;
    for (i=1;i<=n;i++ ) if (!visited[i])  dfs(i);
   }//find_cycle
```

14.[题目分析]有几种方法判断有向图是否存在环路，这里使用拓扑排序法。对有向图的顶点进行拓扑排序，若拓扑排序成功，则无环路；否则，存在环路。题目已假定有向图用十字链表存储，为方便运算，在顶点结点中，再增加一个入度域 indegree，存放顶点的入度。入度为零的顶点先输出。为节省空间，入度域还起栈的作用。值得注意的是，在邻接表中，顶点的邻接点非常清楚，顶点的单链表中的邻接点域都是顶点的邻接点。由于十字链表边（弧）结点个数与边（弧）个数相同（不象无向图边结点个数是边的二倍），因此，对某顶点 v，要判断其邻接点是 headvex 还是 tailvex。

```
   int Topsor(OrthList  g)
     //判断以十字链表为存储结构的有向图 g 是否存在环路，如是，返回 1，否则，返回 0。
    {int top=0; //用作栈顶指针
     for (i=1;i<=n;i++)     //求各顶点的入度。设有向图 g 有 n 个顶点，初始时入度域均为 0
       {p=g[i].firstin;      //设顶点信息就是顶点编号，否则，要进行顶点定位
        while(p)
          {g[i].indegree++; //入度域增 1
           if (p->headvex==i) p=p->headlink; else p=p->taillink; //找顶点 i 的邻接点
          }//while(p)  }//for
      for (i=1;i<=n;i++)       //建立入度为 0 的顶点的栈
         if (g[i].indegree==0) {g[i].indegree=top; top=i; }
      m=0;  //m 为计数器，记输出顶点个数
      while (top<>0)
        {i=top; top=g[top].indegree; m++;       //top 指向下一入度为 0 的顶点
         p=g[i].firstout;
         while (p) //处理顶点 i 的各邻接点的入度
           {if (p->tailvex==i) k=p->headvex; else k=p->tailvex;}//找顶点 i 的邻接点
            g[k].indegree--; //邻接点入度减 1
            if (g[k].indegree==0) {g[k].indegree=top; top=k; }  //入度为 0 的顶点再入栈
            if (p->headvex==i) p=p->headlink; else p=p->taillink;//找顶点 i 的下一邻接点
          }//while (p)
        }// while (top<>0)
      if (m<n) retun(1);  //有向图存在环路
        else return(0);   //有向图无环路
   }//算法结束
```

15. int FirstAdj(AdjMuList g , vertype v)

```
  //在邻接多重表 g 中，求 v 的第一邻接点，若存在，返回第一邻接点，否则，返回 0。
    {i=GraphLocateVertex(g,v); //确定顶点 v 在邻接多重表向量中的下标，不考虑不存在 v 的情况。
     p=g[i].firstedge;   //取第一个边结点。
     if (p==null ) return (0);
     else {if (ivex==i) return (jvex); else return (ivex);}
           //返回第一邻接点，ivex 和 jvex 中必有一个等于 i
    }// FirstAdj
```

16. [题目分析]本题应使用深度优先遍历，从主调函数进入 dfs(v)时 ，开始记数，若退出 dfs()前，已访问完有向图的全部顶点（设为 n 个），则有向图有根，v 为根结点。将 n 个顶点从 1 到 n 编号，各调用一次 dfs()过程，就可以求出全部的根结点。题中有向图的邻接表存储结构、记顶点个数的变量、以及访问标记数组等均设计为全局变量。建立有向图 g 的邻接表存储结构参见上面第 2 题，这里只给出判断有向图是否有根的算法。

```
  int  num=0,  visited[]=0  //num 记访问顶点个数,访问数组 visited 初始化。
  const  n=用户定义的顶点数;
  AdjList g ;              //用邻接表作存储结构的有向图 g。
  void  dfs(v)
    {visited [v]=1;  num++; //访问的顶点数＋1
     if (num==n) {printf(“%d 是有向图的根。\n”,v); num=0;}//if
     p=g[v].firstarc;
     while (p)
       {if (visied[p->adjvex]==0) dfs (p->adjvex);
        p=p->next;} //while
     visited[v]=0; num--; //恢复顶点 v
    }//dfs
  void  JudgeRoot()
     //判断有向图是否有根，有根则输出之。
   {static int i ;
    for (i=1;i<=n;i++ ) //从每个顶点出发，调用 dfs()各一次。
     {num=0; visited[1..n]=0; dfs(i); }
   }// JudgeRoot
```

算法中打印根时，输出顶点在邻接表中的序号（下标），若要输出顶点信息，可使用 g[i].vertex。

17. [题目分析] 使用图的遍历可以求出图的连通分量。进入 dfs 或 bfs 一次，就可以访问到图的一个连通分量的所有顶点。

```
  void  dfs ()
    {visited[v]=1; printf ( “%3d”,v); //输出连通分量的顶点。
     p=g[v].firstarc;
     while (p!=null)
       {if (visited[p->adjvex==0])  dfs(p->adjvex);
        p=p->next;
       }//while
     }// dfs
  void  Count()
```

```
//求图中连通分量的个数
{int k=0 ; static AdjList g ; //设无向图 g 有 n 个结点
 for (i=1;i<=n;i++ )
    if (visited[i]==0) { printf ("\n 第%d 个连通分量:\n",++k); dfs(i);}//if
 }//Count
```

算法中 visited[]数组是全程变量，每个连通分量的顶点集按遍历顺序输出。这里设顶点信息就是顶点编号，否则应取其 g[i].vertex 分量输出。

18.
```
void bfs(AdjList GL, vertype v )
//从 v 发广度优先遍历以邻接表为存储结构的无向图 GL。
 {visited[v]=1;
  printf( "%3d",v);              //输出第一个遍历的顶点。
  QueueInit(Q); QueueIn(Q ,v); //先置空队列，然后第一个顶点 v 入队列，设队列容量足够大
  while (!QueueEmpty(Q))
    {v=QueueOut(Q); p=GL[v].firstarc; //GL 是全局变量， v 入队列。
     while (p!=null)
       {if(visited[p->adjvex]==0)
          {printf("%3d",p->adjvex);                    visited[p->adjvex]=1;
QueueIn(Q,p->adjvex);}//if
         p=p->next;
         }//while
    }// while (!QueueEmpty(Q))
 }//bfs
void  BFSCOM()
  //广度优先搜索，求无向图 G 的连通分量。
  { int count=0; //记连通分量个数。
    for (i=1;i<=n;i++)  visited[i]=0;
    for (i=1;i<=n;i++)
      if (visited[i]==0) {printf("\n 第 %d 个连通分量:\n",++count);
bfs(i);}//if
   }//BFSCOM
```

19.请参见上题 18。HEAD，MARK，VER，LINK 相当于上题 GL，visited，adjvex，next。

20.
```
void  Traver(AdjList g,vertype v)
  //图 g 以邻接表为存储结构，算法从顶点 v 开始实现非递归深度优先遍历。
  {struct arc *stack[];
   visited[v]=1; printf(v);  //输出顶点 v
   top=0; p=g[v].firstarc; stack[++top]=p;
   while(top>0 || p!=null)
     {while (p)
        if (p && visited[p->adjvex]) p=p->next;
        else {printf(p->adjvex); visited[p->adjvex]=1;
              stack[++top]=p; p=g[p->adjvex].firstarc;
             }//else
      if (top>0) {p=stack[top--]; p=p->next; }
```

```
    }//while  }//算法结束。
```

[算法讨论] 以上算法适合连通图，若是非连通图，则再增加一个主调算法,其核心语句是

```
for (vi=1;vi<=n;vi++) if (!visited[vi]) Traver(g,vi);
```

21. （1）限于篇幅，邻接表略。

（2）在邻接点按升序排列的前提下，其 dfs 和 bfs 序列分别为 BADCEF 和 BACEDF。

（3）

```
void dfs(v)
  {i=GraphLocateVertex(g ,v); //定位顶点
   visited[i]=1; printf(v);   //输出顶点信息
   p=g[i].firstarc;
   while (p)
     { if (visited[p->adjvex]==0) dfs(g[p->adjvex].vertex);
       p=p->next;
      }//while
  }//dfs
void traver( )
  //深度优先搜索的递归程序；以邻接表表示的图 g 是全局变量。
 { for (i=1;i<=n;i++)  visited[i]=0; //访问数组是全局变量初始化。
   for (vi=v1;vi<=vn;vi++)
       if (visited[GraphLocateVertex(g, vi)]==0) dfs(vi);
 }//算法结束。
```

22. [题目分析]强连通图指从任一顶点出发,均可访问到其它所有顶点,故这里只给出 dfs()过程。

```
PROC dfs(g:AdjList , v0:vtxptr)
 //从 v0 出发，深度优先遍历以邻接表为存储结构的强连通图 g。
 TYPE stack=ARRAY[1..n] OF arcptr; //栈元素为弧结点指针，n 为顶点个数。
 s:stack; top:integer; top:=0
 visited[v0]:=1;
 write(v0:4); //设顶点信息就是顶点编号；否则要顶点定位。
 p:=g[v0]^.firstarc;
WHILE (top>0 || p!=NIL) DO
  BEGIN WHILE (p!= NIL) DO
          IF (visited[p^.adjvex]=1) THEN p:=p^.next; //查未访问的邻接点。
          ELSE  BEGIN w:=p^.adjvex; visited[w]:=1; top:=top+1; s[top]:=p;
                      p:=g[w].firstarc ;
                END;  //深度优先遍历。
        IF (top>0) THEN BEGIN p:=s[top]; top:=top-1; p:=p^.next END;
  END;
ENDP;
```

23. [题目分析] 本题是对无向图G的深度优先遍历，dfs 算法上面已有几处（见 20-22）。这里仅给出将连通分量的顶点用括号括起来的算法。为了得出题中的输出结果，各顶点的邻接点要按升序排列。

```
void Traver( )
  {for (i=1;i<=nodes(g);i++)  visited[i]=0; //visited 是全局变量，初始化。
```

```
    for (i=1;i<=nodes(g);i++)
      if (visied[i]==0) {printf ("("); dfs(i); printf (")");}//if
    }//Traver
```

24.
```
void  visit(vertype v)          //访问图的顶点 v。
void  initqueue (vertype Q[])   //图的顶点队列 Q 初始化。
void  enqueue (vertype Q[] ,v)   //顶点 v 入队列 Q 。
vertype delqueue (vertype Q[])   //出队操作。
int   empty (Q)                  //判断队列是否为空的函数，若空返回 1 ，否则返回 0 。
vertype firstadj(graph g ,vertype v)//图 g 中 v 的第一个邻接点。
vertype nextadj(graph g ,vertype v ,vertype w)//图 g 中顶点 v 的邻接点中在 w 后的邻接点
void  bfs (graph g ,vertype v0)
      //利用上面定义的图的抽象数据类型，从顶点 v0 出发广度优先遍历图 g。
      {visit(v0);
      visited[v0]=1; //设顶点信息就是编号，visited 是全局变量。
      initqueue(Q);  enqueue(Q,v0); //v0 入队列。
      while (!empty(Q))
        {v=delqueue(Q); //队头元素出队列。
         w=firstadj(g ,v); //求顶点 v 的第一邻接点
         while (w!=0) //w!=0 表示 w 存在。
           {if (visited[w]==0) //若邻接点未访问。
              {visit(w); visited[w]=1; enqueue(Q,w);}//if
            w=nextadj(g,v,w); //求下一个邻接点。
            }//while }//while
       }//bfs
  void  Traver()
       //对图 g 进行宽度优先遍历，图 g 是全局变量。
      {for (i=1;i<=n;i++)  visited[i]=0;
       for (i=1;i<=n;i++)
         if (visited[i]==0) bfs(i);
       }   //Traver
```

25.[题目分析] 本题应使用深度优先遍历。设图的顶点信息就是顶点编号，用 num 记录访问顶点个数，当 num 等于图的顶点个数（题中的 NODES(G)），输出所访问的顶点序列，顶点序列存在 path 中，path 和 visited 数组，顶点计数器 num，均是全局变量，都已初始化。

```
void SPathdfs(v0)
   //判断无向图 G 中是否存在以 v0 为起点，包含图中全部顶点的简单路径。
  {visited[v0]=1;  path[++num]=v0;
   if (num==nodes(G) //有一条简单路径，输出之。
       {for (i=1;i<=num;i++) printf( "%3d",path[i]); printf( "\n"); exit(0);}
//if
   p=g[v0].firstarc;
   while (p)
     {if (visited[p->adjvex]==0) SPathdfs(p->adjvex); //深度优先遍历。
```

```
        p=p->next; //下一个邻接点。
       }//while
      visited[v0]=0; num--; //取消访问标记，使该顶点可重新使用。
     }//SPathdfs
```

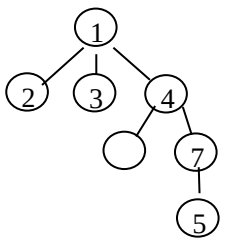
26. 与上题类似，这里给出非递归算法，顶点信息仍是编号。

```
    void AllSPdfs(AdjList g,vertype u,vertype v)
      //求有向图 g 中顶点 u 到顶点 v 的所有简单路径，初始调用形式：AllSPdfs(g,u,v)
      { int top=0,s[];
        s[++top]=u; visited[u]=1;
        while (top>0 || p)
             {p=g[s[top]].firstarc;  //第一个邻接点。
              while (p!=null && visited[p->adjvex]==1) p=p->next; //下一个访问邻接点表。
               if (p==null) top--; //退栈。
               else { i=p->adjvex; //取邻接点（编号）。
                      if (i==v) //找到从 u 到 v 的一条简单路径，输出。
                        {for (k=1;k<=top;k++) printf("%3d",s[k]); printf("%3d\n",v);}//if
                      else { visited[i]=1; s[++top]=i; } //else 深度优先遍历。
                    }//else  }//while
         }// AllSPdfs
```

类似本题的另外叙述的第（2）题 ：u 到 v 有三条简单路径：uabfv，ucdv，ucdefv。

27. （1）[题目分析]D_搜索类似 BFS，只是用栈代替队列，入出队列改为入出栈。查某顶点的邻接点时，若其邻接点尚未遍历，则遍历之，并将其压入栈中。当一个顶点的所有邻接点被搜索后，栈顶顶点是下一个搜索出发点。

```
  void D_BFS(AdjList g ,vertype v0)
      // 从 v0 顶点开始，对以邻接表为存储结构的图 g 进行 D_搜索。
      { int s[], top=0;     //栈，栈中元素为顶点，仍假定顶点用编号表示。
        for (i=1,i<=n;i++) visited[i]=0;  //图有 n 个顶点，visited 数组为全局变量。
        for (i=1,i<=n;i++)  //对 n 个顶点的图 g 进行 D_搜索。
          if (visited[i]==0)
            {s[++top]=i; visited[i]=1; printf("%3d",i);
             while (top>0)
                {i=s[top--]; //退栈
                 p=g[i].firstarc; //取第一个邻接点
                 while (p!=null)  //处理顶点的所有邻接点
                   {j=p->adjvex;
                    if (visited[j]==0) //未访问的邻接点访问并入栈。
                      {visited[j]=1; printf("%3d",i);s[++top]=j;}
                    p=p->next;
                   } //下一个邻接点
                }//while(top>0)
            } //if
       }//D_BFS
```

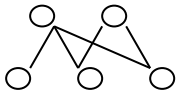


（2）D_搜索序列：1234675，生成树如图：

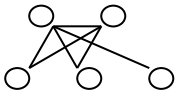
28，[题目分析] 本题的含义是对有向无环图(DAG)的顶点，以整数适当编号后，可使其邻接矩阵中对角线以下元素全部为零。根据这一要求，可以按各顶点出度大小排序，使出度最大的顶点编号为 1，出度次大者编号为 2，出度为零的顶点编号为 n。这样编号后，可能出现顶点编号 i<j，但却有一条从顶点到 j 到 i 的弧。这时应进行调整，即检查每一条弧，若有<i,j>，且 i>j，则使顶点 j 的编号在顶点 i 的编号之前。

```
void Adjust(AdjMatrix g1 ,AdjMatrix g2)
  //对以邻接矩阵存储的 DAG 图 g1 重新编号，使若有<i,j>，则编号 i<j，重新编号后的图以邻接矩阵 g2 存储。
{typedef struct { int vertex ,out ,count }node ; //结点结构：顶点，出度，计数。
 node v[];   //顶点元素数组。
 int c[];    //中间变量
 for (i=1;i<=n;i++)    //顶点信息数组初始化，设图有 n 个顶点。
   {v[i].vertex=i; v[i].out=0; v[i].count=1; c[i]=1;}   //count=1 为最小
 for (i=1;i<=n;i++)    //计算每个顶点的出度。
   for (j=1;j<=n;j++) v[i].out+=g1[i][j];
 for (i=n;i>=2;i--)    //对 v 的出度域进行计数排序，出度大者，count 域中值小。
   for (j=i-1;j>=1;j--)
     if (v[i].count<=v[j].count) v[i].count++; else v[j].count++;
 for (i=1;i<=n;i++) //第二次调整编号。若<i,j>且 i>j，则顶点 j 的编号在顶点 i 的编号之前
   for (j=i;j<=n;j++)
     if(g1[i][j]==1 && v[i].count>v[j].count) {v[i].count=v[j].count;v[j].count++;}
 for (i=n;i>=2;i--)) //对 v 的计数域 v[i].count 排序，按 count 域从小到大顺序存到数组 c 中。
   for (j=i-1;j>=1;j--)
     if (v[i].count<v[j].count) c[j]++; else c[i]++;
 for (i=1;i<=n;i++)  v[i].count=c[i]; //将最终编号存入 count 域中。
 for (i=1;i<=n;i++)
   for (j=1;j<=n;j++)
     g2[v[i].count][v[j].count]=g1[v[i].vertex][v[j].vertex];
}//算法结束
```

29.



二部图



非二部图

[题目分析] 将顶点放在两个集合 V1 和 V2。对每个顶点，检查其和邻接点是否在同一个集合中，如是，则为非二部图。为此，用整数 1 和 2 表示两个集合。再用一队列结构存放图中访问的顶点。

```
int BPGraph (AdjMatrix g)
   //判断以邻接矩阵表示的图 g 是否是二部图。
  {int s[]; //顶点向量，元素值表示其属于那个集合（值 1 和 2 表示两个集合）
```

```
    int Q[];//Q 为队列，元素为图的顶点，这里设顶点信息就是顶点编号。
    int f=0,r,visited[]; //f 和 r 分别是队列的头尾指针,visited[]是访问数组
    for (i=1;i<=n;i++) {visited[i]=0;s[i]=0;} //初始化，各顶点未确定属于那个集合
    Q[1]=1; r=1; s[1]=1;//顶点 1 放入集合 S1
  while(f<r)
   {v=Q[++f]; if (s[v]==1) jh=2; else jh=1;//准备 v 的邻接点的集合号
    if (!visited[v])
     {visited[v]=1; //确保对每一个顶点，都要检查与其邻接点不应在一个集合中
      for (j=1,j<=n;j++)
       if (g[v][j]==1){if (!s[j]) {s[j]=jh; Q[++r]=j;}  //邻接点入队列
                       else if (s[j]==s[v]) return(0);} //非二部图
     }//if (!visited[v])
   }//while
   return(1); }//是二部图
```

[算法讨论] 题目给的是连通无向图，若非连通，则算法要修改。

30. [题目分析] 连通图的生成树包括图中的全部 n 个顶点和足以使图连通的 n-1 条边，最小生成树是边上权值之和最小的生成树。故可按权值从大到小对边进行排序，然后从大到小将边删除。每删除一条当前权值最大的边后，就去测试图是否仍连通，若不再连通，则将该边恢复。若仍连通，继续向下删；直到剩 n-1 条边为止。

```
void  SpnTree (AdjList g)
  //用“破圈法”求解带权连通无向图的一棵最小代价生成树。
  {typedef struct {int i,j,w}node; //设顶点信息就是顶点编号，权是整型数
   node edge[];
   scanf( "%d%d",&e,&n) ; //输入边数和顶点数。
   for (i=1;i<=e;i++)      //输入 e 条边：顶点，权值。
      scanf("%d%d%d" ,&edge[i].i ,&edge[i].j ,&edge[i].w);
   for (i=2;i<=e;i++)     //按边上的权值大小，对边进行逆序排序。
    {edge[0]=edge[i]; j=i-1;
     while (edge[j].w<edge[0].w)  edge[j+1]=edge[j--];
     edge[j+1]=edge[0]; }//for
   k=1;  eg=e;
   while (eg>=n)        //破圈，直到边数 e=n-1.
     {if (connect(k))  //删除第 k 条边若仍连通。
        {edge[k].w=0; eg--; }//测试下一条边 edge[k]，权值置 0 表示该边被删除
      k++;  //下条边
     }//while
  }//算法结束。
```

connect()是测试图是否连通的函数，可用图的遍历实现，若是连通图，一次进入 dfs 或 bfs 就可遍历完全部结点，否则，因为删除该边而使原连通图成为两个连通分量时，该边不应删除。前面第 17,18 题就是测连通分量个数的算法，请参考。“破圈”结束后，可输出 edge 中 w 不为 0 的 n-1 条边。限于篇幅，这些算法不再写出。

31. [题目分析]求单源点最短路径问题，存储结构用邻接表表示，这里先给出所用的邻接表中的边结点的定义： `struct node {int adjvex,weight; struct node *next;}p;`

```
void  Shortest_Dijkstra(AdjList cost ,vertype v0)
 //在带权邻接表 cost 中，用 Dijkstra 方法求从顶点 v0 到其它顶点的最短路径。
  {int dist[],s[]; //dist 数组存放最短路径，s 数组存顶点是否找到最短路径的信息。
   for (i=1;i<=n;i++) {dist[i]=INFINITY; s[i]=0; } //初始化，INFINITY 是机器中最大的数
   s[v0]=1;
   p=g[v0].firstarc;
   while(p)  //顶点的最短路径赋初值
     {dist[p->adjvex]=p->weight; p=p->next;}
   for (i=1;i<n;i++) //在尚未确定最短路径的顶点集中选有最短路径的顶点 u。
     {mindis=INFINITY; //INFINITY 是机器中最大的数，代表无穷大
      for (j=1;j<=n;j++)
        if (s[j]==0 && dist[j]<mindis) {u=j; mindis=dist[j];}//if
      s[u]=1;   //顶点 u 已找到最短路径。
      p=g[u].firstarc;
      while(p)  //修改从 v0 到其它顶点的最短路径
      {j=p->adjvex;
       if (s[j]==0 && dist[j]>dist[u]+p->weight) dist[j]=dist[u]+p->weight;
       p=p->next;
      }
     }// for (i=1;i<n;i++)
  }//Shortest_Dijkstra
```

32. 本题用 FLOYD 算法直接求解如下：

```
 void ShortPath_FLOYD(AdjMatrix g)
   //求具有 n 个顶点的有向图每对顶点间的最短路径
 {AdjMatrix length; //length[i][j]存放顶点 vi 到 vj 的最短路径长度。
  for (i=1;i<=n;i++)
    for (j=1;j<=n;j++) length[i][j]=g[i][j]; //初始化。
  for (k=1;k<=n;k++)
    for (i=1;i<=n;i++)
      for (j=1;j<=n;j++)
        if (length[i][k]+length[k][j]<length[i][j])
            length[i][j]=length[i][k]+length[k][j];
   }//算法结束
```

33.[题目分析] 该题可用求每对顶点间最短路径的 FLOYD 算法求解。求出每一顶点（村庄）到其它顶点（村庄）的最短路径。在每个顶点到其它顶点的最短路径中，选出最长的一条。因为有 n 个顶点，所以有 n 条，在这 n 条最长路径中找出最短一条，它的出发点（村庄）就是医院应建立的村庄。

```
void  Hospital(AdjMatrix w,int n)
 //在以邻接带权矩阵表示的 n 个村庄中，求医院建在何处，使离医院最远的村庄到医院的路径最短。
 {for (k=1;k<=n;k++)   //求任意两顶点间的最短路径
   for (i=1;i<=n;i++)
```

```
      for (j=1;j<=n;j++)
        if (w[i][k]+w[k][j]<w[i][j])  w[i][j]=w[i][k]+w[k][j];
    m=MAXINT;              //设定 m 为机器内最大整数。
    for (i=1;i<=n;i++)     //求最长路径中最短的一条。
      {s=0;
       for (j=1;j<=n;j++) //求从某村庄 i（1<=i<=n）到其它村庄的最长路径。
         if (w[i][j]>s) s=w[i][j];
       if (s<=m) {m=s; k=i;}//在最长路径中，取最短的一条。m 记最长路径，k 记出发顶点的下标。
       Printf(“医院应建在%d 村庄，到医院距离为%d\n”,i,m);
       }//for
    }//算法结束
```

对以上实例模拟的过程略。在 $A^{(6)}$矩阵中（见下图），各行中最大数依次是 9，9，6，7，9，9。这几个最大数中最小者为 6，故医院应建在第三个村庄中,离医院最远的村庄到医院的距离是 6。

34. （1）该有向图的强连通分量有两个：一个是顶点 a,另一个由 b,c,e,d 四个顶点组成。

（2）因篇幅有限从略，请参见本章四？

（3）用 FLOYD 算法求每对顶点间最短距离，其 5 阶方阵的初态和终态如下：

$A^{(0)}=$ $A^{(5)}=$ 第 33 题结果矩阵

$A^{(6)}=$

由于要求各村离医院较近（不是上题中“离医院最远的村庄到医院的路径最短”），因此算法中求出矩阵终态后，将各列值相加，取最小的一个，该列所在村庄即为所求（本题答案是医院建在 b 村，各村到医院距离和是 11），下面是求出各顶点间最短的路径后，求医院所在位置的语句段：

```
  min=MAXINT ; //设定机器最大数作村庄间距离之和的初值。
  k=1; //k 设医院位置。
  for (j=1;j<=n;j++)
    {m=0 ;
     for (i=1;i<=n;i++) m=m+w[i][j];
     if (min>m) { min=m ;k=j;}      //取顶点间的距离之和的最小值。
    }//for
```

35. [题目分析] 本题应用宽度优先遍历求解。若以 v0 作生成树的根为第 1 层，则距顶点

v0 最短路径长度为 K 的顶点均在第 K+1 层。可用队列存放顶点，将遍历访问顶点的操作改为入队操作。队列中设头尾指针 f 和 r，用 level 表示层数。

```
void bfs_K ( graph g ,int v0 ,K)
  //输出无向连通图 g（未指定存储结构）中距顶点 v0 最短路径长度为 K 的顶点。
 {int Q[]; //Q 为顶点队列，容量足够大。
  int f=0,r=0,t=0; //f 和 r 分别为队头和队尾指针，t 指向当前层最后顶点。
  int level=0,flag=0;  //层数和访问成功标记。
  visited[v0]=1; //设 v0 为根。
  Q[++r]=v0; t=r; level=1; //v0 入队。
  while (f<r && level<=K+1)
    {v=Q[++f];
     w=GraphFirstAdj(g,v);
     while (w!=0)   //w!=0 表示邻接点存在。
       {if (visited[w]==0)
         {Q[++r]=w; visited[w]=1;//邻接点入队列。
          if (level==K+1){ printf("距顶点 v0 最短路径为 k 的顶点%d ",w);
          flag=1;}//if
          }//if
        w=GraphNextAdj(g ,v ,w);
       }//while(w!=0)
     if (f==t) {level++; t=r; }//当前层处理完，修改层数，t 指向下一层最后一
          个顶点
    }//while(f<r && level<=K+1)
   if (flag==0) printf( "图中无距 v0 顶点最短路径为%d 的顶点。\n",K);
  }//算法结束。
```

[设法讨论]本题亦可采取另一个算法。由于在生成树中结点的层数等于其双亲层次数加 1，故可设顶点和层次数 2 个队列，其入队和出队操作同步，其核心语句段如下：

```
QueueInit(Q1) ; QueueInit(Q2); //Q1 和 Q2 是顶点和顶点所在层次数的队列。
visited[v0]=1;  //访问数组初始化，置 v0 被访问标记。
level=1; flag=0; //是否有层次为 K 的顶点的标志
QueueIn(Q1,v0); QueueIn(Q2,level); //顶点和层数入队列。
while (!empty(Q1) && level<=K+1)
  {v=QueueOut(Q1); level=QueueOut(Q2);//顶点和层数出队。
   w=GraphFirstAdj(g,v0);
   while (w!=0)  //邻接点存在。
     {if (visited[w]==0)
        if (level==K+1)
          {printf("距离顶点 v0 最短路径长度为 K 的顶点是%d\n",w);
           visited[w]=1; flag=1; QueueIn(Q1 ,w); QueueIn(Q2,level+1); }//if
      w=GraphNextAdj(g ,v ,w);
     }//while(w!=0)
  }//while(!empty(Q1) && level<K+1)
 if (flag==0) printf( "图中无距 v0 顶点最短路径为%d 的顶点。\n",K);
```

36. 对于无环连通图，顶点间均有路径，树的直径是生成树上距根结点最远的两个叶子间的

距离，利用深度优先遍历可求出图的直径。

```
int dfs(Graph g ,vertype parent ,vertype child ,int len)
   //深度优先遍历，返回从根到结点 child 所在的子树的叶结点的最大距离。
 {current_len=len; maxlen=len;
  v=GraphFirstAdj(g ,child);
  while (v!=0) //邻接点存在。
    if (v!=parent)
      {len=len+length(g ,child ,c); dfs(g ,child ,v ,len);
       if (len>maxlen) maxlen=len;
       v=GraphNextAdj(g ,child ,v); len=current_len; } //if
   len=maxlen;
   return(len);
 }//结束 dfs。
int  Find_Diamenter (Graph g)
    //求无向连通图的直径，图的顶点信息为图的编号。
  {maxlen1=0; maxlen2=0; //存放目前找到的根到叶结点路径的最大值和次大值。
   len=0;  ///深度优先生成树根到某叶结点间的距离
   w=GraphFirstAdj(g,1);   //顶点 1 为生成树的根。
    while (w!=0)  //邻接点存在。
      {len=length(g ,1 ,w);
       if (len>maxlen1) {maxlen2=maxlen1; maxlen1=len;}
       else if (len>maxlen2) maxlen2=len;
       w=GraphNextAdj(g ,1 ,w);//找顶点 1 的下一邻接点。
      }//while
    printf( "无向连通图 g 的最大直径是%d\n" ,maxlen1+maxlen2);
    return(maxlen1+maxlen2);
  }//结束 find_diamenter
```

算法主要过程是对图进行深度优先遍历。若以邻接表为存储结构，则时间复杂度为 O(n+e)。

37.[题目分析] 利用 FLOYD 算法求出每对顶点间的最短路径矩阵 w，然后对矩阵 w，求出每列 j 的最大值，得到顶点 j 的偏心度。然后在所有偏心度中，取最小偏心度的顶点 v 就是有向图的中心点。

```
void  FLOYD_PXD(AdjMatrix g)
  //对以带权邻接矩阵表示的有向图 g，求其中心点。
  {AdjMatrix w=g ;       //将带权邻接矩阵复制到 w。
   for (k=1;k<=n;k++)     //求每对顶点间的最短路径。
     for (i=1;i<=n;i++)
       for (j=1;j<=n;j++)
         if ( w[i][j]>w[i][k]+w[k][j]) w[i][j]=w[i][k]+w[k][j];
    v=1;   dist=MAXINT;   //中心点初值顶点 v 为 1，偏心度初值为机器最大数。
    for (j=1;j<=n;j++)
      {s=0;
       for (i=1;i<=n;i++) if (w[i][j]>s) s=w[i][j]; //求每列中的最大值为该顶点的偏心度。
```

```
        if (s<dist) {dist=s; v=j;}//各偏心度中最小值为中心点。
        }//for
      printf( "有向图 g 的中心点是顶点%d,偏心度%d\n",v,dist);
    }//算法结束
```

38. 上面第 35 题是求无向连通图中距顶点 v_0的最短路径长度为 k 的所有结点，本题是求有向无环图中距每个顶点最长路径的长度，由于每条弧的长度是 1，也需用宽度优先遍历实现，当队列为空前，最后一个结点的层次（减 1）可作为从某顶点开始宽度优先遍历的最长路径长度。

```
PROC  bfs_Longest ( g: Hnodes)
  //求以邻接表表示的有向无环图 g 的各个顶点的最长路径长度。有关变量已在高层定义。
   visited: ARRAY[1..n] OF integer; //访问数组。
   Q: ARRAY[1..n] OF integer; //顶点队列。
   f,r,t,level: integer;       //f,r 是队头队尾指针，t 记为当前队尾，level 层数。
   FOR i:=1 TO n DO            //循环，求从每个顶点出发的最长路径的长度
    [visited[1..n]:=0;  f:=0;  level:=0; //初始化。
     visited[i]:=1; r:=r+1; Q[r]:=i; t:=r; level:=1; //从顶点 i 开始宽度优先遍历.
     WHILE (f<r) DO
       [f:=f+1; v:=Q[f];   //队头元素出队。
        p:=g[v].firstarc;
        WHILE (p<>NIL) DO
         [j:=p^adjvex; //邻接点。
          IF visited[j]=0 THEN [visited[j]:=1; r:=r+1; Q[r]:=j;]
          p=p^nextarc;
         ]//WHILE(p<>NIL)
        IF f=t THEN [level:=level+1; t:=r; ]//t 指向下层最后一个顶点。
       ]//WHILE (f<r)
      g[i].mpl:=level-1; //将顶点 i 的最长路径长度存入 mpl 域
    ]//FOR
  EDNP;
```

39. [题目分析] 按 Dijkstra 路径增序求出源点和各顶点间的最短路径，上面 31 题已有解答。本题是求出最小生成树，即以源点为根，其路径权值之和最小的生成树。在确定顶点的最短路径时，总是知道其（弧出自的）前驱（双亲）顶点，我们可用向量 p[1..n]记录各顶点的双亲信息，源点为根，无双亲，向量中元素值用-1 表示。

```
   void Shortest_PTree ( AdjMatrix G, vertype v0)
      //利用从源点 v0 到其余各点的最短路径的思想,产生以邻接矩阵表示的图 G 的最小生成树
      {int d[] ,s[] ; //d 数组存放各顶点最短路径，s 数组存放顶点是否找到最短路径。
       int p[] ;       //p 数组存放顶点在生成树中双亲结点的信息。
       for (i=1;i<=n;i++)
         {d[i]=G[v0][i]; s[i]=0;
          if (d[i]<MAXINT) p[i]=v0;  //MAXINT 是机器最大数，v0 是 i 的前驱（双亲）。
          else p[i]=-1; }//for        //i 目前无前驱，p 数组各量初始化为-1。
       s[v0]=1; d[v0]=0; p[v0]=-1;    //从 v0 开始，求其最小生成树。
```

```
    for (i=1;i<n;i++)
      {mindis=MAXINT;
       for (j=1;j<=n;j++)
         if (s[j]==0 && d[j]<mindis) { u=j; mindis=d[j];}
       s[u]=1;   //顶点 u 已找到最短路径。
       for (j=1;j<=n;j++)   //修改 j 的最短路径及双亲。
         if (s[j]==0 && d[j]>d[u]+G[u][j]) {d[j]=d[u]+G[u][j]; p[j]=u;}
      }//for (i=1;i<=n;i++)
      for (i=1;i<=n;i++) //输出最短路径及其长度，路径是逆序输出。
        if (i!=v0)
          {pre=p[i]; printf( "\n 最短路径长度＝%d，路径是：%d，",d[i],i);
           while (pre!=-1) { printf( ",%d",pre); pre=p[pre];} //一直回溯到根
结点。
          }//if
     }//算法结束
```

40. [题目分析] 由关节点定义及特性可知，若生成树的根有多于或等于两棵子树，则根结点是关节点；若生成树某非叶子顶点 v，其子树中的结点没有指向 v 的祖先的回边，则 v 是关节点。因此，对图 G＝(v,{Edge})，将访问函数 visited[v]定义为深度优先遍历连通图时访问顶点 v 的次序号，并定义一个 low( )函数：

low(v)＝Min(visited[v],low[w],visited[k])

其中(v,w)∈Edge，(v,k)∈Edge，w 是 v 在深度优先生成树上的孩子结点，k 是 v 在深度优先生成树上的祖先结点。在如上定义下，若 low[w]>=visited[v]，则 v 是根结点，因 w 及其子孙无指向 v 的祖先的回边。由此，一次深度优先遍历连通图，就可求得所有关节点。

```
int  visited[] ,low[]; //访问函数 visited 和 low 函数为全局变量。
int  count;            //记访问顶点个数。
AdjList g;            //图 g 以邻接表方式存储
void dfs_articul(vertype v0)
    //深度优先遍历求关节点。
 {count++; visited[v0]=count; //访问顶点顺序号放入 visited
  min=visited[v0];  //初始化访问初值。
  p=g[v0].firstarc; //求顶点 v 的邻接点。
  while (p!=null)
    {w=p->adjvex; //顶点 v 的邻接点。
     if (visited[w]==0) //w 未曾访问，w 是 v0 的孩子。
       {dfs_articul(g ,w);
        if (low[w]<min) min =low[w];
        if (low[w]>=visited[v0]) printf(g[v0].vertex); //v0 是关节点。
        }//if
     else //w 已被访问，是 v 的祖先。
       if (visited[w]<min) min=visited[w];
     p=p->next;
    }//while
   low[v0]=min;
 }//结束 dfs_articul
```

```
void  get_articul( )
    //求以邻接表为存储结构的无向连通图 g 的关节点。
   {for (vi=1;vi<=n;vi++) visited[vi]=0; //图有 n 个顶点，访问数组初始化。
    count=1; visited[1]=1;                  //设邻接表上第一个顶点是生成树的根。
    p=g[1].firstarc;  v=p->adjvex;
    dfs_articul(v);
    if (count<n) //生成树的根有两棵以上子树。
      {printf(g[1].vertex);//根是关节点
       while(p->next!=null)
         {p=p->next; v=p->adjvex; if(visited[v]==0) dfs-articul(v);
         }//while
       }//if  }//结束 get-articul
```

41. [题目分析] 对有向图进行深度优先遍历可以判定图中是否有回路。若从有向图某个顶点 v 出发遍历，在 dfs(v)结束之前，出现从顶点 u 到顶点 v 的回边，图中必存在环。这里设定 visited 访问数组和 finished 数组为全局变量，若 finished[i]=1,表示顶点 i 的邻接点已搜索完毕。由于 dfs 产生的是逆拓扑排序，故设一类型是指向邻接表的边结点的全局指针变量 final，在 dfs 函数退出时，把顶点 v 插入到 final 所指的链表中，链表中的结点就是一个正常的拓扑序列。邻接表的定义与本书相同，这里只写出拓扑排序算法。

```
int visited[]=0; finished[]=0; flag=1;   //flag 测试拓扑排序是否成功
ArcNode  *final=null;    //final 是指向顶点链表的指针，初始化为 0
void  dfs(AdjList g,vertype v)
   //以顶点 v 开始深度优先遍历有向图 g，顶点信息就是顶点编号.
  {ArcNode *t;  //指向边结点的临时变量
   printf("%d",v); visited[v]=1; p=g[v].firstarc;
   while(p!=null)
     {j=p->adjvex;
      if (visited[j]==1 && finished[j]==0) flag=0 //dfs 结束前出现回边
        else if(visited[j]==0) {dfs(g,j); finished[j]=1;} //if
      p=p->next;
     }//while
   t=(ArcNode *)malloc(sizeof(ArcNode));//申请边结点
   t->adjvex=v; t->next=final; final=t;    //将该顶点插入链表
  } //dfs 结束
int dfs-Topsort(Adjlist g)
  //对以邻接表为存储结构的有向图进行拓扑排序，拓扑排序成功返回 1，否则返回 0
  {i=1;
   while (flag && i <=n)
     if (visited[i]==0) {dfs(g,i);  finished[i]=1; }//if
   return(flag);
    }// dfs-Topsort
```

42. [题目分析] 地图涂色问题可以用“四染色“定理。将地图上的国家编号（1 到 n），从编号 1 开始逐一涂色，对每个区域用 1 色，2 色，3 色，4 色（下称“色数”）依次试探，若当前所取颜色与周围已涂色区域不重色，则将该区域颜色进栈；否则，用下一颜色。若 1 至 4 色均与相邻某区域重色，则需退栈回溯,修改栈顶区域的颜色。用邻接矩阵数据结构

C[n][n]描叙地图上国家间的关系。n 个国家用 n 阶方阵表示，若第 i 个国家与第 j 个国家相邻，则 C_{ij}=1，否则 C_{ij}=0。用栈 s 记录染色结果，栈的下标值为区域号，元素值是色数。

```
    void  MapColor(AdjMatrix  C)
        //以邻接矩阵 C 表示的 n 个国家的地区涂色
      {int s[];   //栈的下标是国家编号，内容是色数
       s[1]=1;    //编号 01 的国家涂 1 色
       i=2;j=1;   //i 为国家号，j 为涂色号
       while (i<=n)
         {while (j<=4 && i<=n)
            {k=1;  //k 指已涂色区域号
             while (k<i && s[k]*C[i][k]!=j) k++;  //判相邻区是否已涂色
             if (k<i) j=j+1;       //用 j+1 色继续试探
             else {s[i]=j;i++;j=1;}//与相邻区不重色，涂色结果进栈，继续对下一区涂色进行试探
            }
         }//while (j<=4 && i<=n)
       if (j>4) {i--; j=s[i]+1;}  //变更栈顶区域的颜色。
      }//while   }//结束 MapColor
```