```
{
DataType data;
struct Node * lchild;
struct Node * rchild;
}BiTNode, *BiTree;
```

17 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：
找出最小值结点，且打印该数值；
若该数值是偶数，则将其直接后继结点删除；
单链表类型描述：

```
typedef struct Node
{  ElemType data;
       struct Node   * next;
}Node, *LinkList;
```

18 已知一个二叉树采用二叉链表存放，写一算法，统计出二叉树中叶子结点的个数。

19 在带头结点的单链表 head 的结点 a 之后插入新元素 x。

20 二叉树采用链接存储结构，试设计一个按层次顺序（同一层次自左至右）遍历二叉树的算法。

答案

判断题答案

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. √

50.×

51×　52 ×　53 √　54×　55×　56 √　57 ×　58 √　59 ×　60 √

填空题参考答案

1.线性结构、树形结构　2.　4,2　3.比较关键字，移动记录
4. p->next==L　5.129　6. n-1
7.n+1　8. $\llcorner \log_2 n \rfloor +1$　9.块间（或分块）
10.N_1-1　11. 1296　12.递增
13.逻辑、物理　14.栈空、栈满。　15.1，3，5、1
16.p->next=p->next->next
17. p->next=head　18.5
19.出度，弧数。
20.8,10,11,13,12,15,16,17,19,18
21. n/2　22. 数字分析法、除留取余法　23. 16、4
24.n(n-1)/2，n(n-1)　25.a
26. 0 $\lfloor \log_2 n \rfloor +1$
27. 事件　活动
28. 前驱　后继
29. 双亲表示法　孩子表示法　孩子兄弟表示法（不分顺序）
30. 1196
31. 开放定址法　再哈希法　链地址法　建立公共溢出区（不分顺序）
32. 假溢出
33. n_2+1
34.ABDECF　DEBACF　EDBFCA
35. 集合
36．9174
37．集合 线性 树　图或网（不分顺序）
38．n^2
39. 活动　活动间的优先关系

40. 2^{H-1}　$2^{H}-1$
41. p->ltag==0
42. 比较　移动（不分顺序）
43. 满　空　n　栈底　两栈顶指针相邻（即值之差的绝对值为 1）
44. n-1　n(n-1)/2
45. O(1),随机　　46.其前驱结点的指针域（或 next 指针）
47. 索引表、子块　48. (a,b)　49. 越大
50.值　　51. ⌊n/2⌋, $\lfloor \log_2 n \rfloor +1$
52.28,6,12,20　　53. 出度
54. 连通图　　55. 递增　　56. 21
57.(F+1) % m
58.(n+1)/2
59.s->next=p->next; s->next=s
60.n-1
61.CBA
62. 4，16
63.树的形态
64.左子树，右子树
65.n(n-1)/2
66.正序或反序
67.迪杰斯特拉
68. n(n-1)/2
69.（n-1）/2
70. 数据之间关系
71. O（1）
72. n-i+1
73. 10
74. 0
75. 邻接表
76. 中序
77. 防止出界
78. 1->4->2->3
79. 顺序存储
80.指针
81.s->next=L->next->next ; L->next->next= s.
82.Q.front=Q.rear=Null;
83.空串
84.n/2
85.具有相同关键字的待排记录的相对位置没有发生改变。
86.29
87.25/7
88.P->Ltag= =0;
89. 出度　入度
90. 线性结构　树状结构

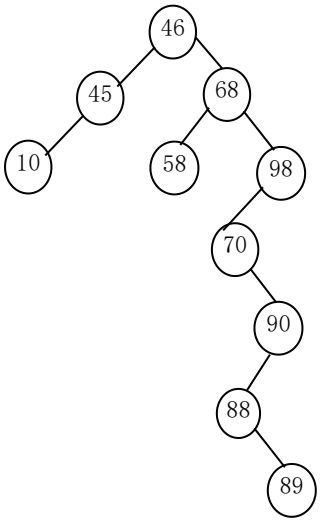
91．0　Vk 入度减 1，若为 0 进栈　　环（回路）
92. 由空格组成的串　空格的个数
93．数字分析法　除留余数法
94．4　16
95．位置相邻　指针
96．栈
97．块间
98．5
99．2n　n+1
100．递增
101. 均匀　冲突
102. 满　空
103．head==NULL
104．出度　入度
105．邻接表
106. 60
107．平方取中法　除留余数法
108．集合
109．受限
110．0
111．p->ltag==0
112．空串
113．归并排序
114．3　4　A　叶子（终端）　A　0
115．ABDGCEF　DGBAECF　GDBEFCA
116.n+1
117. 块内最大（小）关键字　块起始地址

选择题参考答案

1.B　2.D　3.D　4.D　5.B　6.C　7.D　8.C　9.B　10.A
11.A　12.B　13.B　14.A　15.A　16.D　17B　18C　19A　20C
21.A　22.D　23.B　24.D　25D　26.A　27.A　28.D　29.C　30.A
31.C　32.C　33.B　34.B　35.D　36.B　37.C　38.B　39.B　40.C
41.C　42.C　43.C　44.B　45.B　46.A　47.B　48.B　49.A　50.A
51.A　52.B　53.C　54.C　55.B　56. A　57.A　58.C　59.B　60.A
61.A　62.A　63.A　64.C　65.D　66.D　67.C　68.B　69.C　70.C
71.A　72.B　73.B　74.A　75.D　76.A　77.A　78.B　79.B　80.B
81.A　82.D　83.C　84.B　85.B　86.C　87.A　88.D　89.B　90.C
91.C　92.C　93.C　94.C　95.D　96.B　97.D　98.A　99.A　100.A
101.C　102.A　103.B　104.D　105.C　106.C　107.D　108.A　109.D　110.C
111.A　112.D　113.A　114.B　115.B　116.B　117.A　118.B　119.A　120.A

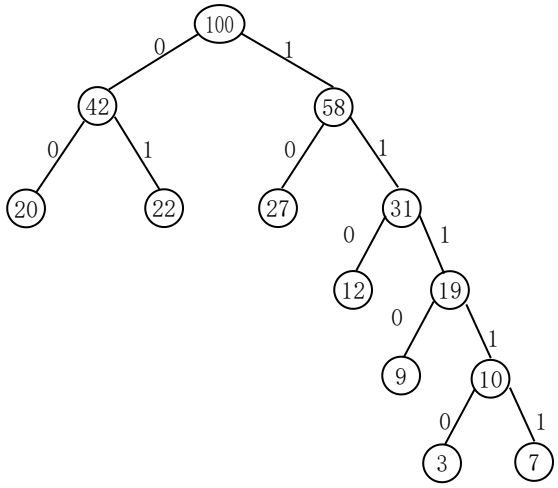
应用题参考答案

1. （4 分）



平均查找长度：(1+2*2+3*3+4*1+5*1+6*1+7*1)/10=3.6 （1 分）

2.把频率扩大 100 倍，Huffman 树为： （3 分）



Huffman 编码：（3 分）

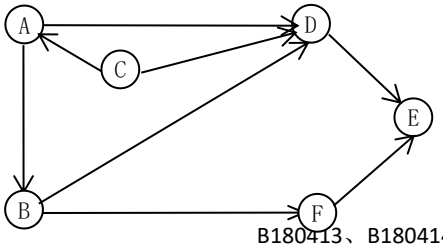
7（a）：11111　　9(b)：1110

12(c)：110　　22(d)：01

20(e)：00　　27(f)：10

3(g)：11110

3. 图：（4 分）



深度优先遍历序列为：CDEABF （1 分）

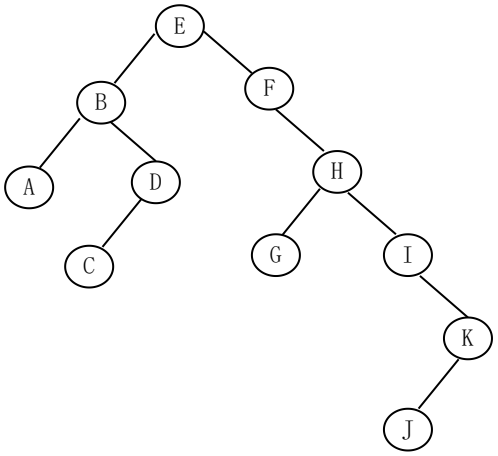
4.哈希表：（3 分）

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	1	14	55	27	68	19	20	84		23	11	10	77				
	1	2	1	4	3	1	1	3		1	1	3	2				

查找成功的平均查找长度为：（1*6+2*2+3*3+4）/12=23/12 （1 分）

查找不成功的平均查找长度为：（1+9+8+7+6+5+4+3+2+1+5+4+3）/13=58/13 （1 分）

5．二叉树为：（5 分）



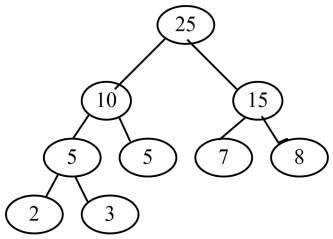
6．第一趟：05 ，55，13，42，94，17，46，70 （2 分）

第二趟：05 ，13，55，42，94，17，46，70 （1 分）

第三趟：05 ，13，17，42，94，55，46，70 （1 分）

该排序方法不稳定 （1 分）

7.（共 8 分）



(6 分)

WPL=55 (2 分)

8.（共 8 分）(18,5,16,19,21,23)，(5，16，21，19，18，23)

9.（共 8 分）线性探测：

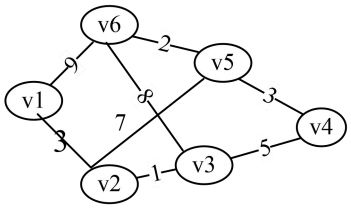
0	
1	
2	54
3	
4	43
5	18
6	31
7	46
8	60
9	58
10	75
11	73
12	90

（4 分）

成功：ASL=（1+1+1+1+1+1+1+2+4+4）/10=17/10　　（2 分）

不成功：ASL=（1+1+2+1+10+9+8+7+6+5+4+3+2）/13=54/13　（2 分）

10.（共 8 分）

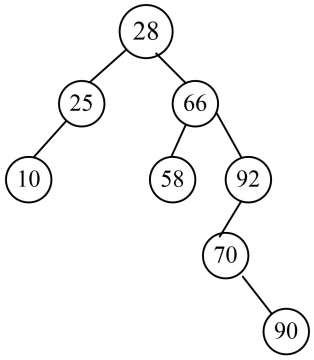


深度：1 2 3 4 5 6　　（2 分）

广度：1 2 6 3 5 4　　（2 分）

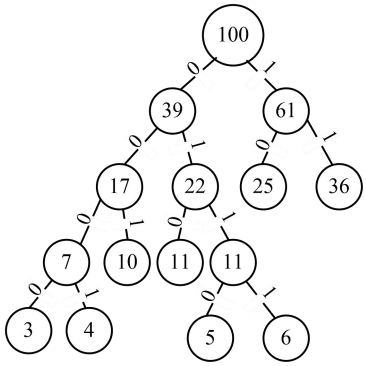
最小生成树边：{（2，3），（2，1），（3，4），（4，5），（5，6）}　（4 分）

11.



（4 分）平均查找长度：(1+2*2+3*3+4*1+5*1)/8=23/8　（1 分）

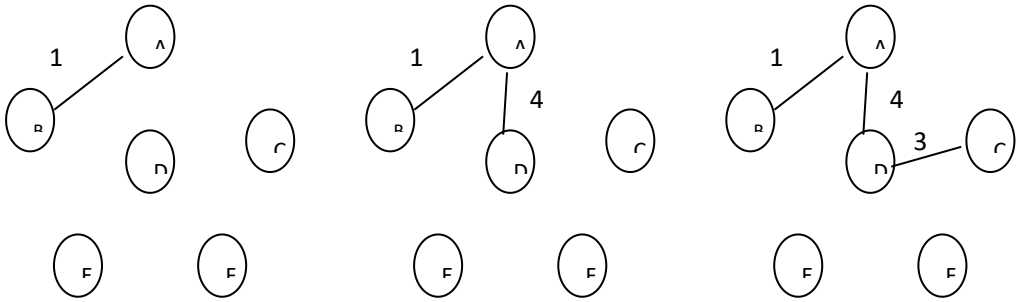
12.把频率扩大 100 倍，Huffman 树为：（3 分）

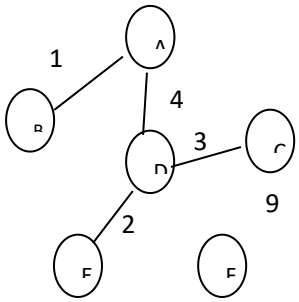


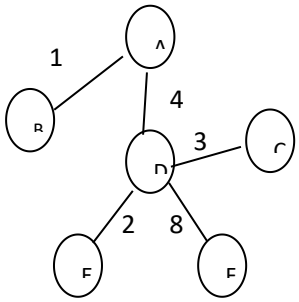
Huffman 编码：（2 分）

5（a）：0110　　25(b)：10　　3(c)：0000　　6(d)：0111
10(e)：001　　11(f)：010　　36(g)：11　　4(h)：0001

13.







每个步骤 1 分，共 5 分

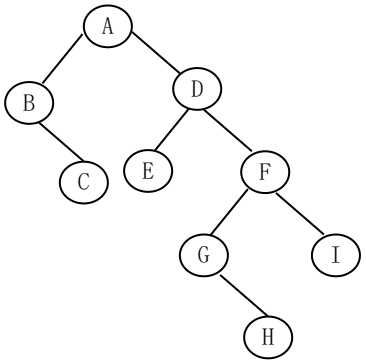
14. 哈希表：（3 分）7，15，20，31，48，53，64，76，82，99

0	1	2	3	4	5	6	7	8	9	10
53	64	76	99	15	48	82	7		20	31
3	4	4	4	1	2	2	1		1	2

查找成功的平均查找长度为：（3+4*3+1*3+2*3）/12=2（1 分）

查找不成功的平均查找长度为：（9+8+7+6+5+4+3+2+1+11+10）/11=6（1 分）

15．二叉树为：(5 分)



16．[初始关键字]：（ 49 ） 38 65 97 76 13 27 49

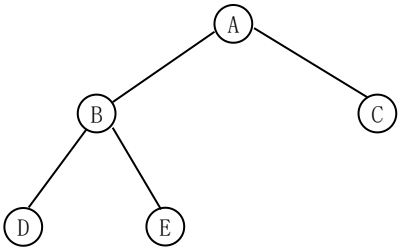
第 1 趟(2 分) i=2 （ 38 49 ） 65 97 76 13 27 49

第 2 趟(1 分) i=3 （ 38 49 65 ） 97 76 13 27 49

第 3 趟(1 分) i=4 （ 38 49 65 97） 76 13 27 49

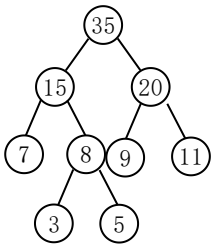
该排序方法不稳定（1 分）

17．链式存储结构(2 分)



前序序列：ABDEC（1 分） 中序序列：DBEAC（1 分） 后序序列：DEBCA（1 分）

18. 哈夫曼树：(4 分)

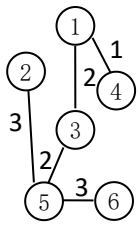


WPL=（7*2+3*3+5*3+9*2+11*2）=78（1 分）

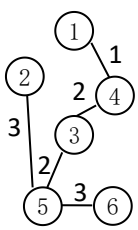
19. 深度优先遍历序列：1,2,3,4,6,5（不唯一）(1 分)

广度优先遍历序列：1,2,3,4,5,6（不唯一）(1 分)

最小生成树： (3 分)



或



20.

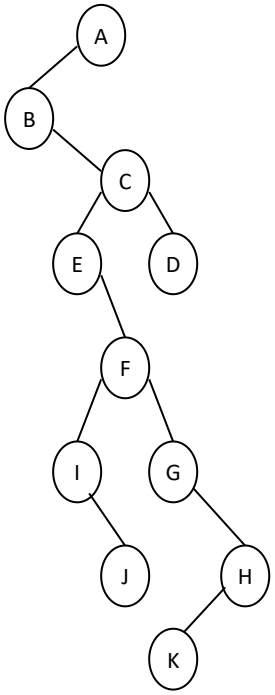
（3 分）

0	1	2	3	4	5	6	7
	8		10	25	32	27	68
	1		1	1	2	1	3

查找成功的平均查找长度为：（1*4+2+3）/6=9/6（1 分）

查找不成功的平均查找长度为：（1+2+1+6+5+4+3）/7=22/7（1 分）

21．转化后的二叉树：　（5 分）



22．第一趟：46 13 42 55 17 05 70 94（2 分）

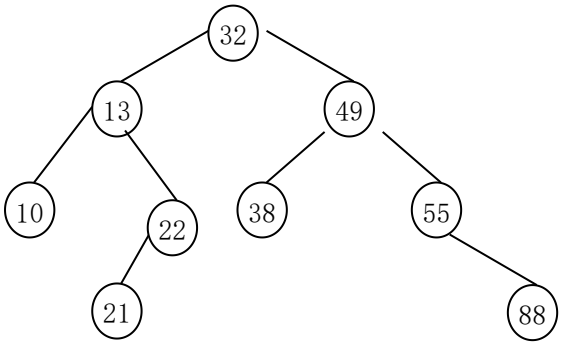
第二趟：13 42 46 17 05 55 70 94（1 分）

第三趟：13 42 17 05 46 55 70 94（1 分）

该排序方法稳定（1 分）

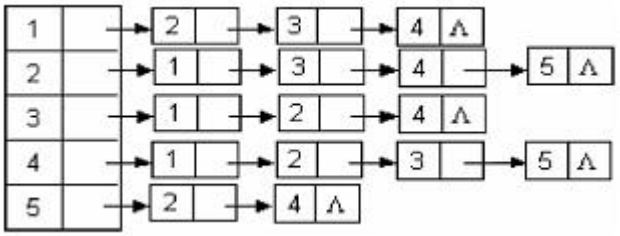
23.（共 5 分）

ASL=（1+2*2+4*3+2*4）/9=25/9　（1 分）

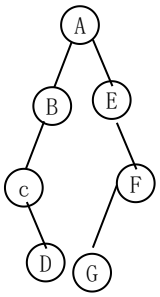


（4 分）

24. （共 5 分）



（3 分）



深度优先遍历该图：12345　(2 分)注：此答案不唯一

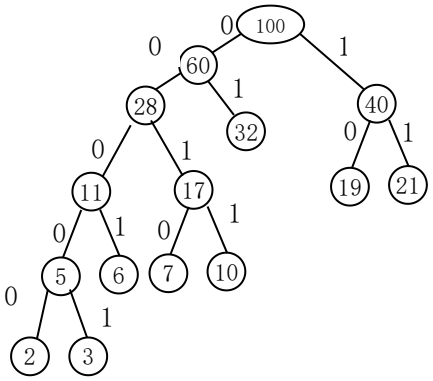
25. （共 5 分。画出二叉树 3 分，写出后序遍历序列 2 分）

后序遍历序列：DCBGFEA

26. （共 5 分）

0.02:00000　　0.03:00001　　0.06:0001　　0.07:0010

0.10:0011　　0.19:10　　0.21:11　　0.32:01　　（2 分）



（3 分）

27. （共 5 分）

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

	32	17	63	49					24	40	10				30	31	46	47
成功的比较次数	1	1	6	3					1	2	1				1	1	3	3
不成功的比较次数	5	4	3	2	1	1	1	1	4	3	2	1	1	1	9	8		

画出阴影中的表格得 3 分

ASLsucc=(6*1+2+3*3+6)/11=23/11　　（1 分）

ASLunsucc=(5+4+3+2+1+1+1+1+4+3+2+1+1+1+1+9+8)/16=47/16.　　（1 分）

28.（共 5 分）

第一趟：{ 50，52，22，85，17，36，55，96 }　　（2 分）

第二趟：{ 50，22，52，17，36，55，85，96 }　　（1 分）

第三趟：{ 22，50，　17，36，52，55，85，96 }　　（1 分）

是稳定的排序法　　（1 分）

29.（共 5 分）

0	1	2	3	4	5	6	7	8	9
27		66	52	12	75	33	41	88	
4		3	1	3	1	2	2	4	

h(75)=75%7=5

h(33)=33%7=5 冲突 h1=（5+1）%10=6

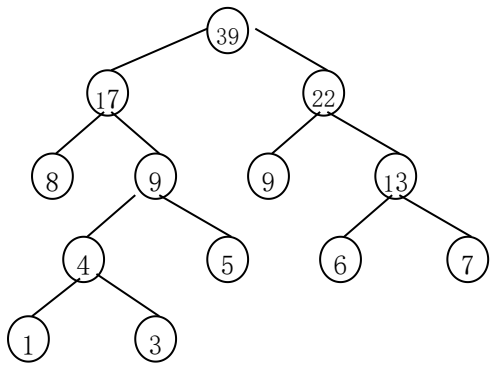
h(52)=52%7=3

其他计算略

得出哈希表　（4 分）

ASLsucc=（4+3+1+3+1+2+2+4）/7=20/7　　（1 分）

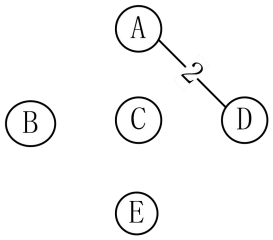
30.　（共 5 分）

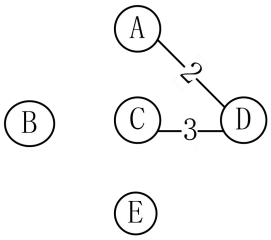


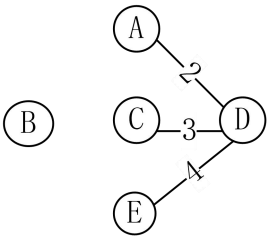
（4 分）

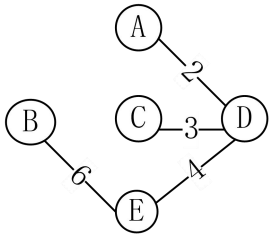
WPL=8*2+9*2+1*4+3*4+5*3+6*3+7*3=104　　（1 分）

31.　（共 5 分）





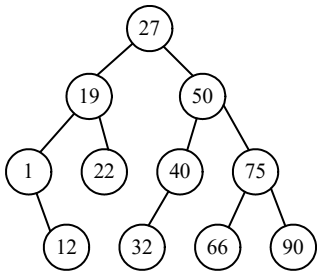




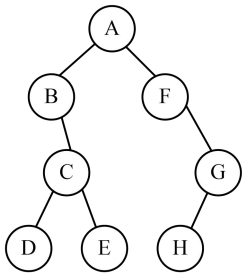
(1)1 分　　(2)　1 分　　(3)　1 分

(4)2 分

32. （共 5 分）



33. （共 5 分）



34. （共 5 分）

第 1 趟（2 分）：07,19,59,38,13,31,29,97,100,45

第 2 趟（1 分）：07,13,31,29,19,59,38,97,100,45

第 3 趟（1 分）：07,13,19,29,31,38,48,59,97,100

该排序方法不稳定（1 分）

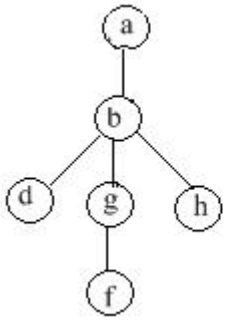
35. （共 5 分）答：顺序存储：用地址连续的地址表示逻辑上的相邻关系。可以实现随机存取，但进行插入删除操作时需要移动大量元素。适合于查询操作比较多时。（2 分）

链式存储：用随机的不连续的存储地址存储线性表，通过指针来表示逻辑上的相邻关系。不能随机存储，要找到链表里面某一元素时必须从头指针开始，依次访问链表。插入删除操作时只需修改指针，不用移动元素。适合插入、删除操作比较多时。（3 分）

36. （共 10 分）

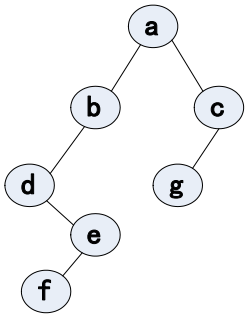
(1) 先：a b d g f h c e　　中：d f g h b a c e　　后：f h g d b e c a （3 分）

(2)



（3 分）

(3)



（4 分）

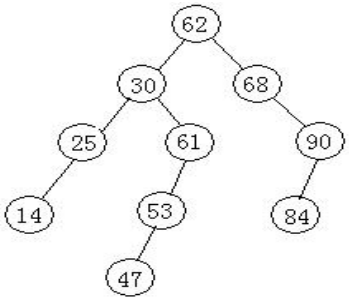
37. （共 10 分）

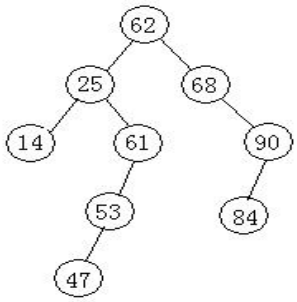
第一趟：18　15　13　20　26　62　29　37　85　（4 分）

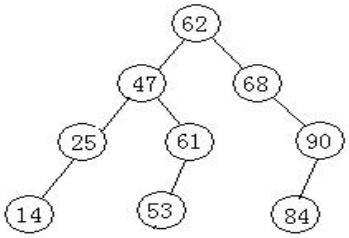
第二趟：13　15　18　20　26　37　29　62　85　（4 分）

第三趟：13　15　18　20　26　29　37　62　85　（2 分）

38. （共 5 分）







（1）　　（2）　　或者（2）

（2 分）　　（3 分）

39.（共 10 分）

27			52	88	7 5	33	41	12	66
0	1	2	3	4		5	6	7	

8　　9

H(75)=75 MOD 7=5；

H(33)=33 MOD 7=5; H1=(H(33)+1)mod 10=6;

H(52)=52 MOD 7=3;

H(41)=41 MOD 7=6; H1=(H(41)+1)mod 10=7;

H(12)=12 MOD 7=5; H1=(H(12)+1)mod 10=6; H2=(H(12)+2)mod 10=6;

H3=(H(12)+3)mod 10=7; H4=(H(12)+4)mod 10=8;

H(88)=88 MOD 7=4;

H(66)=66 MOD 7=3 ; H1=(H(66)+1)mod 10=4; H2=(H(66)+2)mod 10=5;

H3=(H(66)+3)mod 10=6; H4=(H(66)+4)mod 10=7; H5=(H(66)+5)mod 10=8;

H6=(H(66)+6)mod 10=9;

H(27)=27 MOD 7=6; H1=(H(27)+1)mod 10=7; H2=(H(27)+2)mod 10=8;

H3=(H(27)+3) mod 10=9; H4=(H(27)+4) mod 10=0;

(5 分)

ASL(succ)=(1+2+1+2+4+1+7+5)/8=23/8; （3 分）

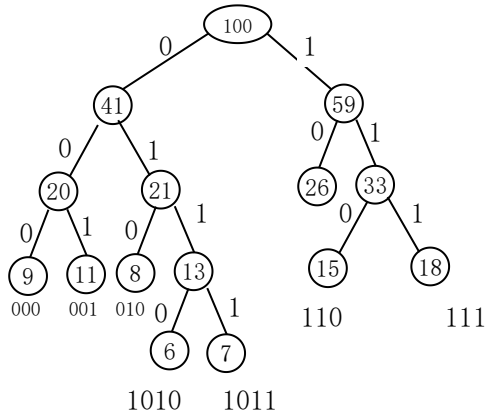
ASL(usucc)=(2+1+1+9+8+7+6)/7=34/7 （2 分）

40.（共 5 分）

（1）head (tail(L1)) （2 分）

（2）head(head(tail(head(L2)))) （3 分）

41.



画出 huffman 树 3 分，给出各编码 2 分。

42.

（1）

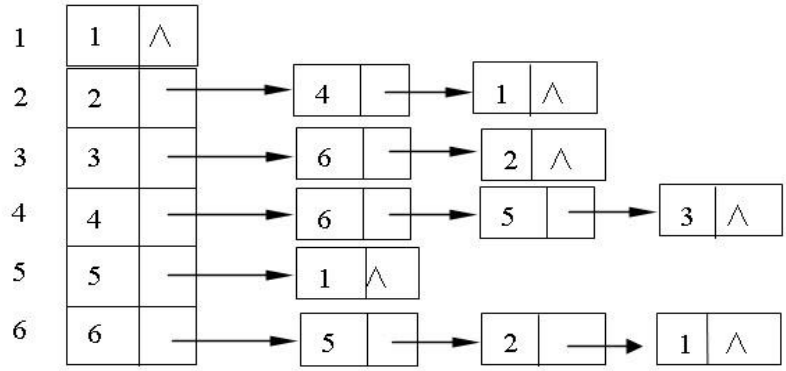
顶点	入度	出度
1	3	0
2	2	2
3	1	2
4	1	3
5	2	1
6	2	3

（1 分）

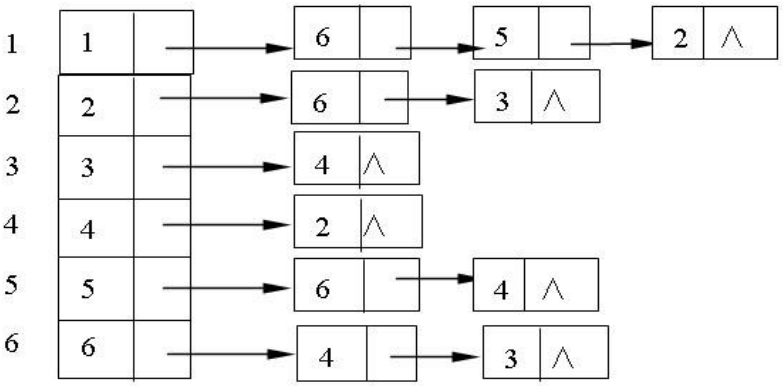
（2）邻接矩阵（1 分）

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

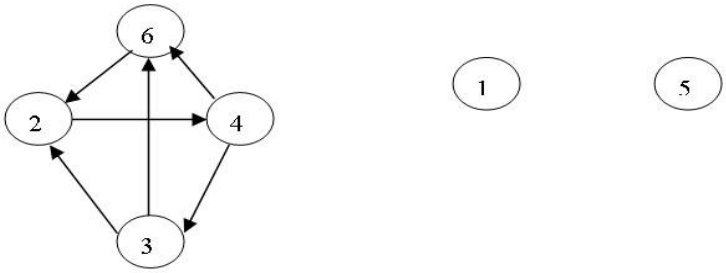
（3）邻接表（1 分）



（4）逆邻接表（1 分）



（5）强连通分量（1 分）



43.

地址	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
关键字		14	01	68	27	55	19	20	84	79	23	11	10			
比较次数		1	2	1	4	3	1	1	3	9	1	1	3			

（3 分）

ASL_{succ}=1/12(1×6+2+3×3+4+9)=2.5 （1 分）

ASL_{unsucc}=1/13(1+13+12+11+10+9+8+7+6+5+4+3+2)=7 （1 分）

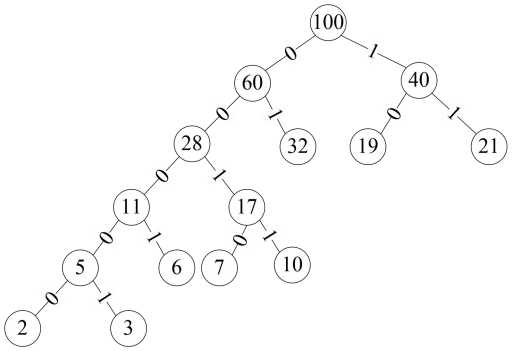
44. 第一趟：5，98，10，23，89，17，69，45，28，19 （2 分）

第二趟：5，10，98，23，89，17，69，45，28，19 （1 分）

第三趟：5，10，17，23，89，98，69，45，28，19 （1 分）

该排序方法不稳定。（1 分）

45.（共 10 分）



(5 分)

0.07， 0.19， 0.02， 0.06， 0.32， 0.03， 0.21， 0.10

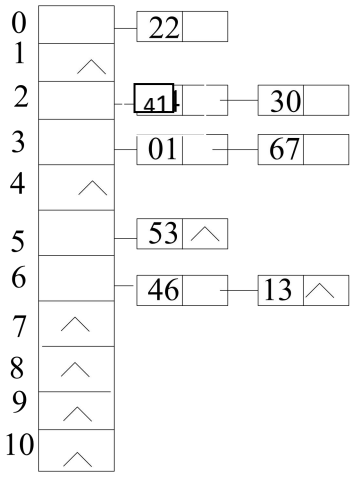
编码 0010 10 00000 0001 01 00001 11 0011 （5 分）

46.（共 10 分）答：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。（5 分）

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。

（5 分）

47.



（4 分）

成功 ASL=(5+3*2)/8=11/8　　（3 分）

不成功 ASL=(6*1+2*2+3*3)/11=19/11　　（3 分）

48. 答：单链表每个结点含有两个域，一个存放数据，一个用来存放后继元素的地址，整个
链表由头指针唯一确定，最后一个结点的后继为空。　　(4 分)

循环链表和单链表类似，唯一区别在于最后一个的后继指向头指针所指向的结点。双向链表每个结点含有三个域，一个存放数据，两个指针域分别存放其前驱和后继结点的地址。三者之间的插入删除操作只需修改只指针，思路基本一致，没有太大区别。　　(4 分)

49. （共 10 分）

```
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;
```

（3 分）

(1) 初始状态

```
cq.front=cq.rear=0;
```

（2 分）

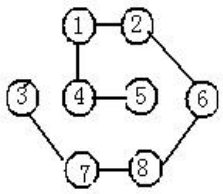
(2) 队列空

cq.front==cq.rear;　　（2 分）

(3) 队列满

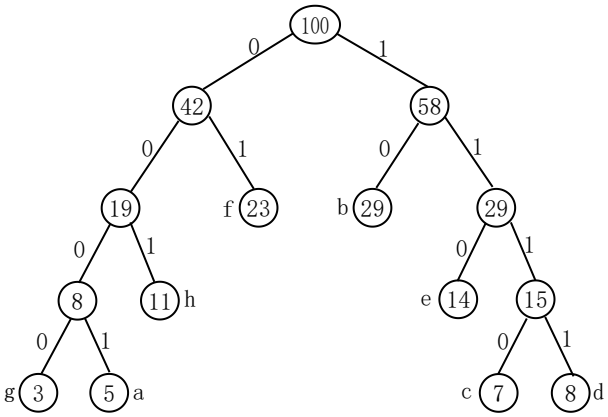
(cq.rear+1)%m==cq.front;　　（3 分）

50. （共 5 分）



（5 分）

51. 为方便计算，可以将所有字符的频率乘以 100，使其转换成整型数值集合，得到{5,29,7,8,14,23,3,11}，以此集合中的数值作为叶子结点的权值构造一棵哈夫曼树，如下图所示。(3 分)

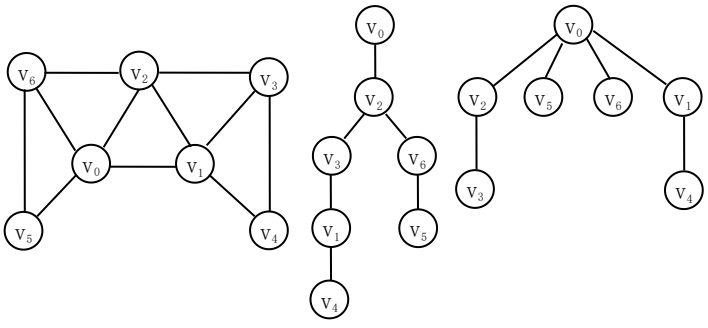


字符集 D 中字符的哈夫曼编码为：a:0001； b:10； c:1110； d:1111； e:110； f:01； g:0000； h:001（2 分）

52. （1）该无向图如下图（a）所示。(3 分)

（2）根据该无向图的邻接表表示，从顶点 v_0 开始的深度优先遍历的序列为 $v_0v_2v_3v_1v_4v_6v_5$，其相应的生成树如下图（b）所示；(1 分)

从顶点 v_0 开始的广度优先遍历的序列为 $v_0v_2v_5v_6v_1v_3v_4$，其相应的生成树如图（c）所示。(1 分)



(a) 无向图　　(b) DFS 生成树　　(c) BFS 生成树

53.

地址	0	1	2	3	4	5	6	7	8	9	10
关键字	22		4	30	01	53	46	13	67		
比较次数	1		1	2	2	1	1	2	6		

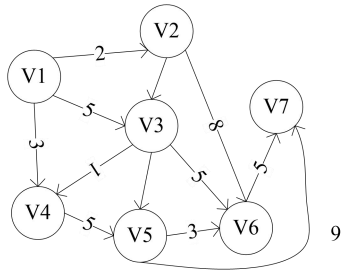
（3 分）

ASLsucc=(1*4+2*3+6)/8=2（1 分）

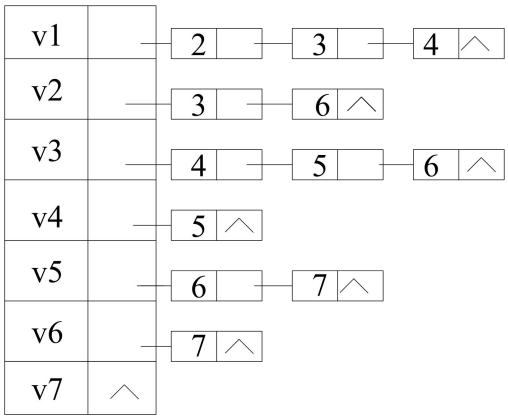
ASLunsucc=(2+1+8+7+6+5+4+3+2+1+1)/11=40/11（1 分）

54. 第一趟：89，98，10，23，5，17，69，45，28，19 （2 分）
第二趟：10，89，98，23，5，17，69，45，28，19 （1 分）
第三趟：10，23，89，98，5，17，69，45，28，19 （1 分）
该排序方法稳定。（1 分）

55.



（2 分）



（4 分）

深度：v1,v2,v3,v4,v5,v6,v7 （2 分）

广度：v1,v2,v3,v4,v6,v5,v7 （2 分）

算法题参考答案

本题无统一答案。每道题编写算法时完成题目功能即可。

1．设顺序表 L 中的数据元素递减有序，编写一个算法，将数据元素 x 插入到顺序表 L 的适当位置上，以保持该顺序表的有序性。

顺序表的类型描述：

```
#define    MAXSIZE   线性表可能达到的最大长度；
typedef   struct
{ ElemType   elem[MAXSIZE]；   /*  线性表占用的数组空间。*/
   int       last；   /*记录线性表中最后一个元素在数组 elem[ ]中的位置（下标值），空表置为-1*/
} SeqList；
```

参考答案：

```
void  SLOrderInsert  (SeqList  *L，ElemType  x)

{  int  i；

   if(L->last>=MaxSize-1)

   {   printf(“\nOverflow!\n”)；

       return；

    }

    /*顺序表 L 中从下标  L.last 至 i 的数据元素依次后移一个位置*/

   for(i=L->last；i>=0  &&  x<L->elem[i]；i--)

       L->elem[i+1]=L->elem[i]；

   L->elem[i+1]=x；        /*把数据元素 x 插入*/

   L->last++；

}
```

评分标准：
参数正确：2 分
移动正确：6 分
插入正确及长度修改正确：2 分
方法二：

```
int Linsert(SeqList *L,int X)
{ int i=0,k;
  if(L->last>=MAXSIZE-1)                /*如果表满，无法插入 X，返回0*/
     { printf("表已满无法插入");
       return(0);
      }
  while(i<=L->last&&L->elem[i]<X)   /*从第一个元素开始比较，找到待插入位置 i*/
       i++;
   for(k=L->last;k>=i;k--)      /*从最后一个元素开始将第 n～i 个元素后移一位*/
       L->elem[k+1]=L->elem[k];
   L->elem[i]=X;          /*将 X 插入到第 i 个位置上*/
   L->last++;             /*表长加1*/
   return(1);
  }
```

评分标准：
参数定义正确：2分
合法性判断：2分
正确移动元素：4分
正确插入元素：1分
正确修改表长：1分

2.二叉树采用二叉链表形式存放，编写递归算法，采用扩展先序遍历序列创建一棵二叉树的二叉链表。
参考答案：

```
void  CreateBiTree(BiTree  *bt)

{  char  ch;

  ch=getchar();

  if(ch=='.')  *bt=NULL;

  else

   {
```

```
      *bt=(BiTree)malloc(sizeof(BiTNode));

   (*bt)->data=ch;

   CreateBiTree(&((*bt)->LChild));

   CreateBiTree(&((*bt)->RChild));

  }

}
```

评分标准：
参数正确：2 分
条件正确：2 分
正确建立二叉链表：6 分

3. 设单链表 head 中的数据元素递增有序，编写一个算法，将数据元素 x 插入到单链表中的适当位置上，以保持该链表的有序性。

```
 int Linsert(LinkList head,int    x)
   { Node *p,*s;
     p=head;
   while(p->next!=NULL&&p->next->data<x)
              p=p->next;
     s=(Node *)malloc(sizeof(Node));
     s->data=x;
     s->next=p->next;
     p->next=s;
   }
```

评分标准：
参数定义正确：2分
找到插入位置：4分
正确插入新结点：4分

4．已知一个二叉树采用二叉链表存放，写一算法，计算二叉树的深度。

```
int dep=0;
 void Depth(BiTree t，int level)
{if(t!=NULL)
   {if(level>dep)    depth=level;
      Depth(t->LChild, level+1);
      Depth(t->RChild, level+1);
     }
}
```

评分标准：

参数定义正确：2分
变量定义正确：1分
条件正确：3分
正确递归调用：4分

5. 假设以带头结点的单链表表示非递减有序表，设计一算法删除表中所有值大于 min 且小于 max（假设 min<max）同时释放结点空间。（共 10 分）

```
 int Delete_Between(LinkList L, int max, int min)
{p=L;
while(p->next->data<=min)       p=p->next;
if(p->next)
{q=p->next;
 while(q->data<max)     {k=q; q=q->next; free(k); }
 p->next=q;
}
}
```

评分标准：循环指针 p 赋初值 2 分，找到要删除第一个结点 3 分，删除结点修改链 5 分。

6. 编写在二叉排序树中查找关键字 K 的算法。（共 10 分）

```
BSTree SearchBST（BSTree bst ， KeyType key）
{if(!bst) return NULL;
else if（bst->key==key）return bst;
else if (bst->key>key)
return SearchBST(bst->lchild,key);
 else
return SearchBST(bst->rchild,key);
}
```

评分标准：查找条件正确 2 分，递归查找左子树 4 分，递归查找右子树 4 分。

7．假设有一个带头结点的循环链表 L 的长度大于 1，试编写算法在此链表中删除尾结点。

```
void Dele(LinkList L)
{ Node *p,*r;
   p=L;
   while(p->next->next!=L)
              p=p->next;
 r=p->next;
 p->next=r->next;
 free(r);
}
```

评分标准：
参数定义正确：2分
正确找到尾结点的前驱：4分
正确删除尾结点：4分

8．已知一个二叉树采用二叉链表存放，写一算法，要求统计出二叉树中度为 2 的结点个数。

```
int count=0;
  Void Degree1(BiTree t)
{if(t!=NULL)
   {if(t->lchild&&t->rchild)
      count++;
      Degree1(t->lchid);
      Degree1(t->rchid);
    }
}
```

评分标准：
参数定义正确：2分
变量定义正确：1分
条件正确：3分
正确递归调用：4分

9. 假设在长度大于1的循环链表中，既无头结点也无头指针。s 为指向链表中某个结点的指针，编写算法删除指针 s 所指结点的前趋结点。

```
void Dele(Node * s)
{ Node *p,*r;
   p=s;
   while(p->next->next!=s)
              p=p->next;
  r=p->next;
  p->next=r->next;
  free(r);
}
```

评分标准：
参数定义正确：2分
正确找到 s 的前驱：4分
正确删除结点：4分

10. 设计一个在二叉链表存储结构上统计二叉树中结点个数的算法。

```
    int count=0；
     void countnode(bitree    bt)
     {
         if(bt!=NULL)
         {count++;
         countnode(bt->LChild);
         countnode(bt->RChild);}
     }
```

评分标准：
参数定义正确：2分
变量定义正确：1分
条件正确：3分
正确递归调用：4分

11. 设头指针为 head，编写算法实现带头结点的单链表 head 的就地逆置，即利用原带头结点的单链表 head 的结点空间把数据元素序列(a_1，a_2，……a_n)逆置为(a_n，

a_{n-1}，……，a_1)

参考答案：

```
 void ConverseLink (LinkList head)
{ Node *p，  *q；
   p=head->next；      /*p 为不带头结点的单链表的头指针 ，指向剩下链表中第一个结点*/
   head->next=NULL     /*  使 head 指向带头结点的空单链表  */
   while(p!=NULL)       /*  循环从链表头部插入  */
   { q=p；                  /*q 指向正在处理的结点  */
     p=p->next；          /*p 指向原结点的下一个结点  */
     q->next=head->next；     /*把 q 所指结点链入链表  */
     head->next=q；           /*q 所指结点为链表的首结点  */
   }
}
```

评分标准：

参数正确：2 分

链表头指针及头结点设置正确：2 分

各结点逆置正确：6 分

12．已知一个二叉树采用二叉链表存放，编写算法，在二叉树中求位于先序序列中第 k 个位置的结点的值。

参考答案：

```
void CountNum(BiTree root, int k,int i)
{
 if(root && i!=k)
 {
     i++;
     if(i==k) printf("%c",root->data);
     else
      {
        CountNum(t->LChild,k,i);

            CountNum(t->RChild,k,i);
      }
   }
}
```

评分标准：

参数正确：2 分

条件正确：2 分

正确求出第 k 个值：6 分

13．设计在单链表中删除值相同的多余结点的算法。

```
     typedef  int  datatype;
     typedef  struct  node
     {  datatype  data;
      struct  node  *next;
```

```
}lklist;                                          （2 分）
void  delredundant(lklist  *head)
{
  lklist  *p,*q,*s;
  for(p=head->next;p!=NULL;p=p->next)
  {
   for(q=p->next,s=q;q!=NULL;     )              （3 分）
    if  (q->data==p->data)
     {p->next=q->next;
      q=q->next;
      free(s);                                    （2 分）
      s=q;}
    else
      {s=q;
            q=q->next;                            （3 分）
      }
   }
}
```

评分标准：数据类型定义 2 分；写出查找相同结点循环语句 3 分；找到要删除结点修改链 2 分，找不到修改链 3 分。

14．已知一个二叉树采用二叉链表存放，写一算法，要求统计出二叉树中度为 1 的结点个数。

```
typedef  struct  node
{  datatype  data;
 struct  node  *lchild,*rchild;
}  bitree;
void  fun(bitree  *root)              （2 分）
{
  if  (bt!=NULL)
  {if(root->lchild!=NULL&&root->rchild==NULL)
Count++;
else  if(root->lchild==NULL&&root->rchild!=NULL)  （4 分）
Count++;
}
fun(root->lchild);                                  （2 分）
fun(root->rchild);                                   （2 分）
}
```

评分标准：写出数据结构类型 2 分；写出度为 1 结点判定条件 4 分；递归操作左右子树各 2 分。

15. 设有一个由正整数组成的无序单链表，编写完成下列功能的算法：

（1）找出最大值结点，且打印该数值；

（2）若该数值是偶数，则将其与直接后继结点的数值交换；

参考答案：

```
void fun(LinkList    head)
{
   int max,t;
   Node    *p, *q;
   p=head->next; q=p;
   if(p!=NULL)
   { max=p->data;
      while(p!=NULL)
    {
        if(max<p->data) {max=p->data;q=p;}
        p=p->next;
     }
    printf("max=%d\n",max);
    if(max%2==0 && q->next!=NULL)
      {
         t=q->data;
         q->data=q->next->data;
         q->next->data=t;
   }
}
```

评分标准：

参数正确：2 分

找到最大值位置并输出：6 分

正确互换 2 分

16. 已知一个二叉树采用二叉链表存放，设计一个算法将所有结点的左、右子树相互交换。

参考答案：

```
void convert (BiTree root)
{
   BiTNode *temp;
   if(root!=NULL)
    {
      if(root->lchild!=NULL ||root->rchild!=NULL)
      {
         temp=root->lchild;
         root->lchild=root->rchild;
         root->rchild=temp;
       }
       convert(root->rchild);
```

```
            convert(root->lchild);
        }
    }
```

评分标准：
参数正确：2 分
正确交换：8 分

17．设有一个由正整数组成的无序单链表，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）若该数值是偶数，则将其与直接后继结点的数值交换；
单链表类型描述：

```
typedef struct Node
{   ElemType data;
   struct Node   * next;
}Node, *LinkList;
```

参考答案：

```
        void fun(LinkList    head)
        {
           int min;
           Node    *p, *q;
           p=head->next; q=p;
           if(p!=NULL)
           {
              min=p->data;
              while(p!=NULL)
             {
                 if(min>p->data) {min=p->data; q=p;}
                 p=p->next;
              }
             printf("min=%d\n",min);
             if(min%2==0 && q->next!=NULL)
                {
                 p=q->next;
                 q->next=p->next;
                 free(p);
                }
        }
```

评分标准：
参数正确：2 分
找到最小值位置并输出：6 分
正确删除 2 分

18 已知一个二叉树采用二叉链表存放，写一算法，统计出二叉树中叶子结点的个数。
参考答案：

```
      Int    node=0;        /*node 为保存叶子结点数目的全局变量，调用之前初始
```

```
化为 0 */
    void   CountNode(BinTree   root)        /*求二叉树中叶子结点个数*/
    {
        if (root!=NULL)
        {    CountNode(root->lchild);          /*  中序遍历左子树* /
             if ((root->lchild==NULL)&&(root->rchild==NULL))
                  node++;                      /*结点计数* /
             CountNode(root->rchild);          /*  中序遍历右子树* /
        }/*if*/
      }/* CountNode */
```

评分标准：
参数正确：1 分
计数器定义正确：2 分
正确计数：7 分

19.在带头结点的单链表 head 的结点 a 之后插入新元素 x。 （共10分）

```
typedef struct Node
{Elemtype data;
Struct Node *next;
}Node,*LinkList;
 Void InsertList(LinkList L,Elemtype x,Node *a)
{LinkList p=L->next;
while(p!=null&&p!=a)p=p->next;
if(p)
{s=(LinkList)malloc(sizeof(Node));
s->data=x;
s->next=p->next;
p->next=s;
}
}
```

评分标准：写出数据类型定义2分；找到插入结点 a3分；生成新结点 s2分；修改链完成插入3分。

20. 二叉树采用链接存储结构，试设计一个按层次顺序（同一层次自左至右）遍历二叉树的算法。（共 10 分）

```
 typedef char datatype;
typedef struct node
{datatype data;
struct node *lchild,*rchild;
} bitree;
int   LayTraverse(bitree *bt)
{InitQueue( Q);
p=bt;
 if(bt==null)return error;
EnterQueue(Q,bt);
while(!IsEmpty(Q))
```

```
{DeleteQueue(Q,p);
visit(p);
if(p->lchild) EnterQueue(Q, p->lchild);
f(p->rchild) EnterQueue(Q, p->rlchild);
}
return ok;
}
}
```

评分标准：写出数据类型定义 2 分；初始化队列，根指针进队 2 分；结点的邻接点依次进队，进行循环 6 分。