南大历年真题不完全解答（每“一”题都是来自百度）

1996 年操作系统

二、某单处理器

题目：

(2)有一多道程序设计系统，采用不允许移动的可变分区方式管理主存空间，设主存空间为 100KB，采用最先适应分配算法分配主存，作业调度和进程调度均采用先来先服务算法．今有作业序列如表 6-14 所示。

表 6-14 作业的情况表

作业号	提交时刻/时	运行时间/h	主存要求
1	10.1	0.3 小时	15KB
2	10.3	0.5 小时	60KB
3	10.5	0.4 小时	50KB
4	10.6	0.4 小时	10KB
5	10.7	0.2 小时	20KB

假定所有作业都是计算型作业忽略系统调度时间，回答：

a．作业被装入主存的次序为(　　)；

b．把各个作业被装入主存的时间填入表 6-15 中。

表 6-15　作业装入情况表

作业	装入时间	作业	装入时间
1		4	
2		5	
3			

c．请计算作业的平均周转时间。

答案：

(2)a．作业被装入的次序为：1，2，4，5，3

b．如表 6-19 所示。

表 6-19　作业装入情况

作业	装入时间	作业	装入时间
1	10.1	4	10.6
2	10.3	5	10.7
3	10.9		

c．根据上面的解答，可以得到每个作业的周转时间以及平均周转时间，如表 6-20 所示。

表 6-20　作业执行情况

作业号	提交时刻/时	运行时间/h	开始时刻/时	完成时刻/时	周转时间/h
1	10.1	0.3	10.0	10.4	0.3
2	10.3	0.5	10.4	10.9	0.6
3	10.5	0.4	11.5	11.9	1.4
4	10.6	0.4	10.9	11.3	0.7
5	10.7	0.2	11.3	11.5	0.8

三、无

五、并发系统中诸进程由于资源共享、进程合作，而产生进程之间的相互制约；又因共享资源的方式不同，而导致两种不同的制约关系：

1 间接制约关系（进程互斥）

由于共享资源而引起的临界区内不允许并发进程交叉执行的现象。由共享公有资源而造成的对并发进程执行速度的间接制约

2 直接制约关系（进程同步）

由于并发进程互相共享对方的私有资源所引起的直接制约。

1996 年数据结构

一、

2、4、不清楚

3、（1）s1=substr(s,3,1)　//取出字符：'y'

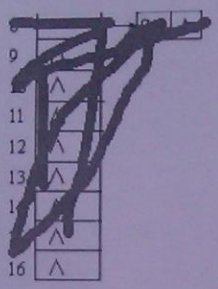
(2) s2=substr(s,6,1)　//取出字符：'+'

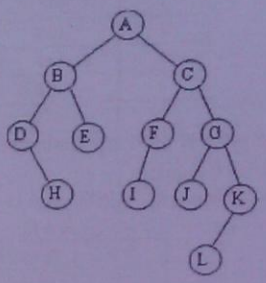
(3) s3=substr(s,1,5)　//取出字串：'xyz'

(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'





第九题

9. 试推导含有 12 个结点的平衡二叉树的最大深度，并画出原理这样的树。

【解答】令 Fk 表示含有最少结点的深度为 k 的平衡二叉树的结点树目，则：

F1=1，F2=2，…，Fn= Fn-2+Fn-1+1　含有 12 个结点的平衡二叉树的最大深度为 5。例如：

5、

6、因为叶子结点不包含关键字，所以可以把叶子结点看成在树里实际上并不存在外部结点，指向这些外部结点的指针为空，叶子结点的数目正好等于树中所包含的关键字总个数加 1。

7、有序？

二、

```
struct node    //邻接表的结构
{ ……
  Type data[i];
  link data[i].adj;
}
int max=0, temp=0;
int maxlenth(int i)
{
 if(i->adj==null) return 0;
 else
 {
  j=i->adj; max=1+maxlenth(j);
  while(j->adj!=null)
  {
   j=j->adj;
   temp=1+maxlenth(j);
   if(max<temp) max=temp;
  }
  return max;
 }
}
```

1997 年操作系统

5、semaphore mutex=1,empty=n,full=0;

```
item buffer[n];
int in=out=0;
void sender(int i)
{
 while (1)
 {
…
Writean mail;
   wait(empty);
   wait(mutex);
   buffer[in]=mail;
   in=(in+1) mod n;
   signal(mutex);
   signal(full);
 }
}
void receiver(int i)
{
while(1)
{
 ...
   wait(full);
   wait(mutex);
   mail=buffer[out];
   out=(out+1) mod n;
   signal(mutex);
   signal(empty);
   ...
   Deal with the mail;
   …
 }
}
main()
{
cobegin{
sender(i);
receiver(i);
}
}
```

1997 年数据结构 三、无

1998 年操作系统

三、无

四、类似题目

假定系统有三个并发进程 read, move 和 print 共享缓冲器 B1 和 B2。进程 read 负责从输入设备上读信息，每读出一个记录后把它存放到缓冲器 B1 中。进程 move 从缓冲器 B1 中取出一记录，加工后存入缓冲器 B2。进程 print 将 B2 中的记录取出打印输出。缓冲器 B1 和 B2 每次只能存放一个记录。要求三个进程协调完成任务，使打印出来的与读入的记录的个数，次序完全一样。请用 PV 操作，写出它们的并发程序。

```
begin SR,SM1,SM2,SP:semaphore;
B1,B2:record;
SR:=1;SM1:=0;SM2:=1;SP:=0
cobegin
process read
X:record;
begin R: (接收来自输入设备上一个记录)
X:=接收的一个记录;
P(SR);
B1:=X;
V(SM1);
goto R;
end;
Process move
Y:record;
begin
M:P(SM1);
Y:=B1;
V(SR)
加工 Y
P(SM2);
B2:=Y;
V(SP);
goto M;
end;
Process print
Z:record;
begin
P:P(SP);
Z:=B2;
V(SM2)
打印 Z
goto P;
end;
coend;
end;
```

1998 年数据结构

一、无

三、LINK * temp;//中间变量

temp=malloc(sizeof(p));//初始化

memcpy(temp,p,sizeof(p));//把 P 位置的值保存起来

memcpy(p,&at,sizeof(p));//把 at 的值放进 p 位置

p->next=temp;//更改 P 的下一节点

或

1) 当前节点指针为 p

```
ListNode     *s = new ListNode;
s->next     =     p->next;
p->next     =     s;
ElemType     temp =p->data;
p->data = T;
s->data     = temp;
```

即在 p 指向结点后插一个结点，然后交换两结点的数据。

四、1、

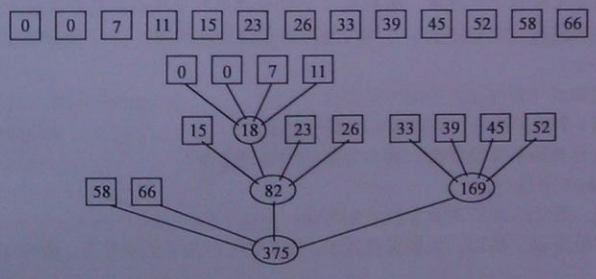
【解答】权值个数n = 11，扩充4 叉树的内结点的度都为4，而外结点的度都为0。设内结点个数为n4，外结点个数为n0，则可证明有关系n0 = 3 * n4 + 1。由于在本题中n0 = 11≠3 * n4 +1，需要补2个权值为0的外结点。此时内结点个数n4 = 4。

2、类似题目

给定一组权值: 23, 15, 66, 07, 11, 45, 33, 52, 39, 26, 58, 试构造一棵具有最小带权外部路径长度的扩充 4 叉树，要求该 4 叉树中所有内部结点的度都是 4，所有外部结点的度都是 0。这棵扩充 4 叉树的带权外部路径长度是多少?

【解答】权值个数 n = 11，扩充 4 叉树的内结点的度都为 4，而外结点的度都为 0。设内结点个数为 n_4，外结点个数为 n_0，则可证明有关系 $n_0 = 3 * n_4 + 1$。由于在本题中 $n_0 = 11 \neq 3 * n_4 + 1$，需要补 2 个权值为 0 的外结点。此时内结点个数 $n_4 = 4$。

仿照霍夫曼树的构造方法来构造扩充 4 叉树，每次合并 4 个结点。



此树的带权路径长度 WPL = 375 + 82 + 169 + 18 = 644。

五、1、强连通图必须从任何一点出发都可以回到原处,每个节点至少要一条出路(单节点除外),至少有 n 条边,正好可以组成一个环。

六、

3、

■ 10. 假设有n个关键字，它们具有相同的Hash函数值，用线性探测方法解决冲突，把这n个关键字散列到大小为n的地址空间中，共计需要做 n（n+1）次插入和探测操作。【武汉大学 2000 一、8】

1999 年数据结构

一、

```
#include<stdio.h>
float f(int a[],int n)
{
  float avg;
  if(n==1) avg=a[0];
  else
  {
    avg = (a[n-1] + f(a,n-1)*(n-1))/n;
  }
  return avg;
}

void main()
{
  int a[10] = {0,1,2,3,4,5,6,7,8,9};
  int sum;
  float avg;
  avg = f(a,10);
  printf("The avg = %f\n",avg);
}
```

二、n、(n-1)(n+1)/6、(n+1)n/6

不对，例如序列{3、3、4、2、1}的“逆序元素”个数是2，2和1是“逆序元素”；但是将第二个3和2交换后，成为{3、2、4、3、1}，此时“逆序元素”个数是3，2、3四、和1是“逆序元素”。

五、

2、一棵深度为H的满k叉树有如下性质：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。如果按层次顺序从1开始对全部结点编号，回答：

①各层的结点数目是多少？

②编号为p的结点的父结点（若存在）的编号是多少

③编号为p的结点的第i个儿子结点（若存在）的编号是多少？

④编号为p的结点有右兄弟的条件是什么？其右兄弟的编号是多少？

【解答】 ①第i层有ki-1个结点。

②p=1时，该结点为根，无父结点,否则其父结点编号为：p+(k−2)k（k≥2）。

③其第k-1个儿子的编号为pk。所以，如果它有儿子，则其第i个儿子的编号为：pk+(i-(k-1))。

④(p-1)%k≠0 时，该结点有右兄弟，其右兄弟的编号为 p+1。

3、解答：二叉树中叶结点必在某结点的左或右子树中，三种遍历方法对左右子树的遍历都是按先左后右的原则进行。所以在三种遍历序列中叶结点的相对次序是不会发生变化的

2000 年操作系统

二、1、进程有四个基本属性

1.多态性 从诞生、运行，直至消灭。

2.多个不同的进程可以包括相同的程序

3.三种基本状态 它们之间可进行转换

4.并发性 并发执行的进程轮流占用处理器属性

四、14

答：用户进程进入临界区时屏蔽所有中断，应当也包括系统程序。若屏蔽的是用户进程，的确可以保护临界资源，但若系统所发出的中断也被屏蔽，则会引起错误，因为系统的中断往往与当前运行的程序无关，却可能是一些重要的硬件中断，如电源故障等，故不可盲目屏蔽所有中断；又或者当时发出故障中断的中断源恰好是该临界资源，则更应该及时响应处理

2000 年数据结构

一、1、无

2、变量或表达式，变量?

3、函数和对应地址?

4、指当调用函数时，除了返回函数值之外，还对主调用函数产生附加的影响。例如修改全局变量（函数外的变量）或修改参数

四、1、无

五、

迭代：

```
int fun2(int n)
        {
            if ( n == 1 || n == 2 ) return 1;
            int tmpe, f1 = 1, f2 = 1;
            for (int i = 2; i <= n; i++ )
            {
                tmpe = f1 + f2;
                f1=f2;
                f2=tmpe;
            }
            return f2;
        }
```

f1+f2 + f2

或者

利用辅助变量 x=FIBi 与 y=FIBi-1，就可以用迭代方案来计算 Fibonicca 数，而避免同一值的重复计算。

```
{ 对 n>0 计算 x=FIBn}
i:=1;x:=1;y:=0;
while i<n do
   begin z:=x; i:=i+1;
      x:=x + y; y:=z
end
```

六、

4、一个结点可能有若干个前驱，也可能有若干个后继

6、表中元素必须按关键字有序

8、无

10、一棵含有 n 个结点的 k 叉树，可能达到的最大深度为 n ，最小深度为 2 。

答：当 k=1(单叉树)时应该最深，深度＝n（层）；当 k=n-1（n-1 叉树）时应该最浅，深度＝2（层），但不包括 n=0 或 1 时的特例情况。教材答案是“完全 k 叉树”，未定量。）

七、5、无

2001 年编译及操作系统

5、无

二、1、 (1)进程：系统进行资源管理和保护的单位，与中级调度相关的实体。

内核级线程：进程的一条执行路径，操作系统进行处理器调度的实体。

用户级线程：进程的一条执行路径，操作系统不知道它的存在，在执行时映射到内核级线程上，用户调度的实体。

(2)划分成实时优先级层次和交互式优先级层次，其中实时优先级层次较高。

实时优先级层次包括多个优先级，可以组织成多个就绪线程队列，也可以组织成一个优先队列；可以采用抢占式优先数调度策略，如果分配时间片，应该较长。

交互式优先级层次可以划分成 3 个就绪线程队列，按照优先级从高到低依次为访问字符设备的就绪线程队列、访问块设备的就绪线程队列、时间片到的就绪线程队列，优先级较高的就绪线程队列具有较短的时间片。

周转时间＝完成时间 - 提交时间

带权周转时间＝周转时间 / 运行时间 （化为统一单位（分或秒）计算）

响应比＝等待时间 / 要求运行时间

2、无

三、

```
var buf:array[0…9] of integer;
    count,getptr,putptr:integer;
    count:=1
    getptr:=0
    putptr:=0;
    semphore:s1,s2,sput,sget;
    s1:=1;
    s2:=1;
    sput:=3;
     sget:=0
Process producer
 begin
   L1:生产 3 个整数
    p(sput)
    p(s1)
    buf[putptr]:=整数
    putptr:=(putptr+1)mod10
     v(sget)
    v(s1)
     goto   L1
```

```
 end
Process consumer
 var y:integer
 begin
   L2: p(sget)
    p(s2)
    y:=buf[getptr]      getptr:=(getptr+1)mod10
    count:=count+1
    if count=3 then begin
         count:=0
         v(sput)
    end
    v(s2)
     消费一个整数 y;
     goto   L2
 end
```

2002 年软件基础

四、1、无

五、2、无

六、

【例题分析】

- 本试题考核二叉树的基本概念以及基本操作，要熟悉二叉树递归定义及路经的概念；
- 问题⑵的设计思想为：若根结点为空，则返回假；若根结点的数域值为 value 且左右孩子为空，则返回为真；否则递归判断

HasPathSum(t->left, value – t->info) 或者

HasPathSum(t->right, value – t->info)

【例题解答】

（1）判断 t 所指结点是否为叶子结点算法:

```
#define          true       1
#define          false      0
Typedef        int              Boolean ;
typedef   struct BTNode{
int                  info ;                    // 二叉树的数据元素
BTNode * Left , *Right ; // 指向左子树和右子树的根结点
}BTNode, *BiTree ;
Boolean      Isleaf( BTNode   * t )
{
if( t == NULL )
return   false ;
else
if(t->Left == NULL && t->Right == NULL )
return   true ;
else
return   false ;
}
```

（2）判断是否存在从根结点到某个叶子结点的路经上，各结点的 info 字段之和等于 value 算法:

```
Boolean     HasPathSum( BTNode   *t , int   value)
{ //t 为指向根结点的指针
Boolean   result ;
if( t == NULL )
return   false ;
else
if(t->Left==NULL && t->Right==NULL && t->info==value)
return   true ;
else {
result = HasPathSum(t->Left, value-t->info) ;
result = result || HasPathSum(t->Right, value-t->info) ;
return   result ;
}
}
```

（3）在最坏情况下，需要检测结点的个数 O(n)，即要检测树中的所有路经.

2003 年软件基础

五、2、

```
1.    #include <iostream>
2.    #include <stack>
3.    using namespace std;
4.
5.    const int _N = 1000;
6.    int list[_N];
7.    int n;
8.
9.    void dfs(int i, int n, stack<int> s)
10.   {
11.       if(i == n)
12.       {
13.           cout<<"{";
14.           while(!s.empty())
15.           {
16.               cout<<" "<<s.top();
17.               s.pop();
18.           }
19.           cout<<"}"<<endl;
20.       }else
21.       {
22.           s.push(list[i]); //取
23.           dfs(i + 1, n, s);
24.           s.pop(); //舍
25.           dfs(i + 1, n, s);
26.       }
```

```
    }
    int main()
    {
        int i;
        while(cin>>n)
        {
            stack<int> s;
            for(i = 0; i < n; i++)
                list[i] = i + 1; //初始化
            dfs(0, n, s);
        }
        return 0;
    }
```

六、1、无

九、无

十、

设置 4 个信号量：

Mutex：表示桥的互斥使用的信号量，初值为 1；

Scounteast：表示由东向西方向的车辆计数器的互斥使用的信号量，初值为 1；

Scountwest：表示由西向东方向的车辆计数器的互斥使用的信号量，初值为 1；

Scount4：表示桥上车辆计数器的信号量，初值为 4。

算法过程如下：

semaphore Mutex，Scounteast，Scountwest，Scount4；

int Counteast，Countwest；

Mutex=1；Scounteast=1；Scountwest=1；Scount4=4；

Counteast=0；Countwest=0；

void P-east()

{

wait(Scounteast)；

if (Counteast= =0) wait(Mutex)；

Counteast++；

signal(Scounteast)；

wait(Scount4)；

过桥；

signal(Scount4)；

wait(Scounteast)；

Counteast--；

if (Counteast= =0) signal (Mutex)；

signal(Scounteast)；

}

void P-west()

{

wait(Scountwest)；

if (Countwest = =0) wait(Mutex)；

```
Countwest ++;
signal(Scountwest);
wait(Scount4);
过桥;
signal(Scount4);
wait(Scountwest);
Count west --;
if (Count west = =0) signal (Mutex);
signal(Scountwest);

}
```

2004 软件基础

```
四、2、O(n*n*n)
3、无
五、无
六、#include "stdio.h"
#include "stdlib.h"
#define MAXSIZE 12500
typedef struct BitNode{
        char c;
        BitNode * lchild,rchild;
}*BitTree;
非递归算法中用栈来存放结点
用 w 来计算叶子的个数
int w=0;
typedef struct stack{
         int top;
         BitTree Maxsize[MAXSIZE];
}*Stack;
创建二叉树
void creat(BitNode * T){
char c;
scanf("%d",&c);
if(c==' ')
T=null;
else{
T=(BitNode *)malloc(sizeof(BitNode *));
T->data=c;
creat(T->lchild);
creat(T->rchild);
}
}

使用非递归算法求叶子节点数
void printTree(BitTree T){
```

初始化栈

```
Stack s;
s=(Stack*)malloc(sizeof(Stack *));
s->top=0;
while(T!=null && s->top!=0){
if(T!=null){
printf(T->data);
s->Maxsize[s->top]=T->data;
s->top++;
T=T->lchild;
}
else{
T=s->Maxsize[s->top];
s->top--;
if(T->lchild==null && T->rchild==null){
w++;}
T=T->rchild;
}
}
}
//递归算法求叶子节点的个数
int leaf(BitTree T){
if(T==null)
  return 0;
else if(T->lchild==null && T->rchild==null)
  return 1;
else return leaf(T->lchild)+leaf(T->rchild);
}
```

2、是最小堆的插入，0-n-1 的最小堆插入 n，调整为 0-n 的最小堆。

九、无

2005 年软件基础

七、

```
void printLeavesDepth(TreeNode * pNode, size_t depth = 0)
{
 if (pNode == NULL) return;
 if (pNode->left == NULL && pNode->right == NULL)
 {
  std::cout << "node " << pNode->name << ": " << depth << std::endl;
 }
 else
 {
  printLeavesDepth(pNode->left, depth+1);
  printLeavesDepth(pNode->right, depth+1);
 }
}
```

2006 年软件工程

一、1、无

二、3、2）

```
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
if(hc+1>hs)return(hc+1); else return (hs);
}
}//结束 height
```

二、7、无

2007 年软件工程

一、5、1100 次

原因：

1，x 为 101 时，执行循环中的 if 判断的内容，这时 y 自减 1；

2，x<=100 时，执行循环中的 else 半段的内容，这时 x 自加 1；

3，基于上述，x 自增 10 次之后，执行一次 y 自减动作；

4，由于 x 初始为 91，所以当循环执行 10 次 else 之后，才执行一次 if；

5，则计算结果为：10*100 + 10*10 = 1000 + 100 = 1100

6、如果一棵具有 n 个结点的深度为 k 的完全二叉树，其叶子结点数和总结点数有这样的关系：n（叶子）=（n 总+1）/2

二、

1、前面有过这个题

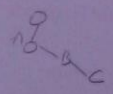
33. 由于地址空间为 10，且从 100 开始，散列函数为 H（key）=key%7+100，

散列地址	100	101	102	103	104	105	106	107	108	109
关键字	98	63	79	17	53	19	61	75	46	19
比较次数	1	2	1	1	1	1	2	3	5	10

用线性探测再散列解决冲突，ASL_succ=27/10

4、

5、n 个非叶节点则有 n+1 个右指针域为空的结点



解释一

森林转换为二叉树,遵循"左儿子右兄弟"的说法.

举个例子.树:根节点有三个儿子 A,B,C.那么转换为二叉树后,根节点只有一个儿子 A,然后 A 的兄弟 B 成为 A 的"儿子"(或者可以说是右指针域),C 成为 B 的右指针域,此时 C 已经没有兄弟了,所以到此的一个右指针域为空.(你可以画图体会一下.)

题目中说 F 有 n 个非终端节点,所以转换为二叉树后所有的空的右指针域(right)就是 n 个.

根节点没有兄弟,所以该右指针域也为空.(注:这里根节点也是一个有指针域.上文中根节点属于非终端节点,那里它所指向的右指针域不是它本身而是它的最右边的儿子.)

所以综上,二叉树中右指针域为空的节点有(n+1)个.

解释二

设叶子结点有 m 个，则总结点为（m+n）个，而森林转化成树是根据左孩子右兄弟存储原理转

化过来的，当转成树时，每个结点有两个指针域，分别是左孩子域和右兄弟域，所以总的左、右指针域分别有（m+n）个。并且有下面的等量关系：
空的右指针域个数+不为空的右指针域个数=总的右指针域个数=m+n；
不为空的右指针域个数+不为空的左指针域个数=总的不为空的指针域个数=m+n-1（因为总结点为 m+n 个）
而原来森林当中有 n 个非终端结点，所以有且仅有 n 个结点有孩子，因此转化成树时有 n 个不为空的左指针域个数，代入上面的第二条式子得不为空的右指针域个数为 m-1，再代入第一条式子，得：空的右指针域个数为 n+1
解释三
森林中的非终端节点都有孩子，这个是肯定的，如果只有一个孩子，那么这个孩子就变成对应的二叉树的左孩子，如果有 2 个孩子，那么这个节点的右孩子就变成了他左边孩子的右孩子....那么节点的右子树就没有孩子了，这里有 N 和终端，所以就有 N 个右子树为空的节点，还有一个是一环套一环以后，还有个非终端节点的右孩子成为在最下边的叶子，显然没有右节点，要算进去，所以有 N+1 个，语文能力有限，不知道楼主能看懂不，画画图帮忙理解吧，这个问题理解了，树，二叉树，森林的转换就没问题了

2008 年数据库&操作系统

四、3、
内存映射文件，是由一个文件到一块内存的映射。Win32 提供了允许应用程序把文件映射到一个进程的函数 (CreateFileMapping)。内存映射文件与虚拟内存有些类似，通过内存映射文件可以保留一个地址空间的区域，同时将物理存储器提交给此区域，内存文件映射的物理存储器来自一个已经存在于磁盘上的文件，而且在对该文件进行操作之前必须首先对文件进行映射。使用内存映射文件处理存储于磁盘上的文件时，将不必再对文件执行 I/O 操作，使得内存映射文件在处理大数据量的文件时能起到相当重要的作用。
六、在本题中，应设置三个信号量 S、So、Sa，信号量 S 表示盘子是否为空，其初值为 1；信号量 So 表示盘中是否有桔子，其初值为 0；信号量 Sa 表示盘中是否有苹果，其初值为 0。同步描述如下：

```
int S=5;
int Sa=0;
int So=0;
        main()
        {
          cobegin
              father();                 /*父亲进程*/
              son();                    /*儿子进程*/
             daughter();                /*女儿进程*/
          coend
        }
        father()
        {
           while(1)
             {
               P(S);
               将水果放入盘中;
               if（放入的是桔子）V(So);
               else                V(Sa);
```

```
        }
    }
son()
{
    while(1)
      {
        P(So);
        从盘中取出桔子;
        V(S);
       吃桔子;
      }
}
daughter()
{
     while(1)
       {
         P(Sa);
         从盘中取出苹果;
         V(S);
         吃苹果;
       }
}
```

七、3、无

5、数组Q［n］用来表示一个循环队列，f为当前队列头元素的前一位置，r为队尾元素的位置，假定队列中元素的个数小于n，计算队列中元素的公式为 （D）

（A）r－f;　　（B）（n＋f－r）％n;　（C）n＋r－f;　　（D）（n＋r－f）％n

九、2、无

十、1、

1) T->lchild == NULL && T->rchild == NULL //叶节点

2) n + T->weight * h　//加上当前叶节点的带权路径长度

3)WPL(T->lchild, h+1) //遍历左子树

4)WPL(T->rchild, h+1) //遍历右子树

2、

```
{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
```