问题与练习答案

第 1 章

1.1 节

1. 上面的两个输入中有且只有一个必须为1，且最下面的输入必须为1。
2. 下面的输入1被NOT门取反为0，使得AND门的输出变为0。因此，OR门的两个输入均为0（记住，触发器上面的输入保持为0），因此OR门的输出变成0。这就意味着，当触发器下面的输入变回0，AND门的输出仍将保持0。
3. 上面的OR门的输出将变为1，使得上面的NOT门得到一个输出0。这会使得下面的OR门得到一个输出0，并使得下面的NOT门得到一个输出1。这个1被看做是触发器的输出，同时反馈给了上面的OR门，这时，它将该门的输出保持为1，即使在触发器的输入已经变回0以后也是如此。
4. a. 整个电路等同于单个XOR门。
   b. 这个电路也等同于单个XOR门。
5. a. 6AF2　　b. E85517　　c. 48
6. a. 0101111110110010111
   b. 0110000100001010
   c. 1010101111001101
   d. 0000000100000000

1.2 节

1. 在第一种情况下，地址为6的存储单元最后结果为值5。在第二种情况下，它的最后结果值为8。
2. 在步骤1当新值写入3号存储单元时，该单元的原始值被擦去了。因此，步骤2并没有将3号存储单元中原始值存入2号存储单元中。结果是：两个存储单元最后的值都是最初2号存储单元中的值。正确的步骤如下：
   步骤1，将2号存储单元中的内容移到1号存储单元。
   步骤2，将3号存储单元中的内容移到2号存储单元。
   步骤3，将1号存储单元中的内容移到3号存储单元。
3. 32 768位。

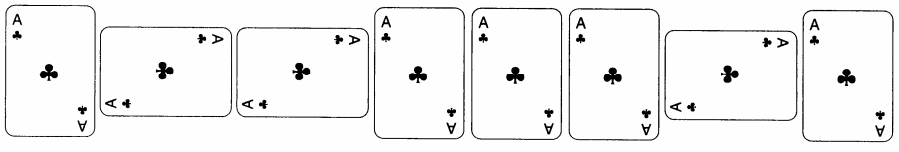
1.3 节

1. 有较快的数据检索速度以及较高的传输速率。
2. 这里要记住的一点是，与计算机内部运作速度相比较，机械动作的缓慢表明：我们应该把必须移动读/写磁头的次数减到最少。如果我们要在写满磁盘的一面后再开始下一面，那么当

我们在写满一个磁道时都必须移动一次读/写磁头。因此磁头移动的次数就大约等于磁盘两个盘面所有磁道的总和。不过，如果我们通过电动方式在磁盘表面之间切换读/写磁头，我们就只需要在每个柱面写满时才移动一次读/写磁头了。

3. 在这个应用中，必须从海量存储系统中随机地检索信息，而对于CD和DVD等设备中使用的螺旋系统，这种方法是很耗时的。（而且，现在的技术还无法对CD和DVD设备中的某部分数据进行更新。）
4. 存储空间是以物理扇区为单元分配的（事实上，在大多数情况下是以扇区组为单元）。如果最后一个物理扇区没有被写满，可以再填加新的文本，而不需要增加分配给该文件的存储空间。如果最后一个物理扇区已经被写满，那么无论要给该文档填加什么内容，都需要分配额外的物理扇区。
5. 闪存驱动器不需要物理运动，因此所需要的响应时间比较短，而且不会有物理损耗。
6. 缓冲区是一个临时的数据存储区域，通常用作解决数据源与最终目的地不一致的手段。

1.4 节

1. Computer Science。
2. 除了从低端数第6位对应的大写字母和小写字母分别是0和1外，两个位模式是相同的。
3. a.
   00100010 01010011 01110100 01101111
   01110000 00100001 00100010 00100000
   01000011 01101000 01100101 01110010
   01111001 01101100 00100000 01110011
   01101000 01101111 01110101 01110100
   01100101 01110100 00101110
   b.
   01000100 01101111 01100101 01110011
   00100000 00110010 00100000 00101011
   00100000 00110011 00100000 00111101
   00100000 00110101 00111111
4. 
5. a. 5　b. 9　c. 11　d. 6　e. 16　f. 18
6. a. 110　b. 1101　c. 1011　d. 10010　e. 11011　f. 100
7. 在24位中，我们利用ASCII码可以存储3个符号。因此，可存储的值最大能够达到999。不过，如果我们将这些位用作二进制数字，那么可存储的值则最大可达16 777 215。
8. a. 15.15　b. 51.0.128　c. 10.160
9. 几何表示法与位图编码的图像相比，更利于改变其尺寸。然而，几何表示法在图像的质量方面没有位图好。正如在1.8节所说的，JPEG位图表示方式在相片中很常见。
10. 以每秒44 100样本的采样频率，一小时的立体声音乐需要635 040 000个字节的存储空间。这也就差不多写满了一张容量略大于600 MB的CD。

1.5 节

1. a. 42 b. 33 c. 23 d. 6 e. 31
2. a. 100000 b. 1000000 c. 1100000 d. 1111 e. 11011
3. a. $3\frac{1}{4}$ b. $5\frac{7}{8}$ c. $2\frac{1}{2}$ d. $6\frac{3}{8}$ e. $\frac{5}{8}$
4. a. 100.1 b. 10.11 c. 1.001 d. 0.0101 e. 101.101
5. a. 100111 b. 1011.110 c. 100000 d. 1000.00

1.6 节

1. a. 3 b. 15 c. −4 d. −6 e. 0 f. −16
2. a. 00000110 b. 11111010 c. 11101111
   d. 00001101 e. 11111111 f. 00000000
3. a. 11111111 b. 10101011 c. 00000100
   d. 00000010 e. 00000000 f. 10000001
4. a. 4位时，最大值是7，最小值是−8。
   b. 6位时，最大值是31，最小值是−32。
   c. 8位时，最大值是127，最小值是−128。
5. a. 0111(5+2=7) b. 0100(3+1=4) c. 1111(5+(−6)=−1)
   d. 0001(−2+3=1) e. 1000(−6+(−2)=−8)
6. a. 0111 b. 1011（溢出） c. 0100（溢出）
   d. 0001 e. 1000（溢出）
7. a. 0110 b. 0011 c. 0100 d. 0010 e. 0001

+0001	+1110	+1010	+0100	+1011
0111	0001	1110	0110	1100

8. 不会。如果对于所使用的系统来说，某人数过大，那么试图存储这个数则会产生溢出现象。当一个正值与一个负值相加时，结果一定在这两个数值之间。于是，如果能够存储原始数值，那么也是可以存储结果。
9. a. 6，因为1110→14−8
   b. −1，因为0111→7−8
   c. 0，因为1000→8−8
   d. −6，因为0010→2−8
   e. −8，因为0000→0−8
   f. 1，因为1001→9−8
10. a. 1101，因为5+8=13→1101
    b. 0011，因为–5+8=3→0011
    c. 1011，因为3+8=11→1011
    d. 1000，因为0+8=8→1000
    e. 1111，因为7+8=15→1111
    f. 0000，因为–8+8=0→0000
11. 不可以。余8记数法可以存储的最大值是7，表示为1111。如果要表示更大的值，至少需要余16记数法（这个记数法采用5位模式）。类似地，6也无法用余4记数法表示。（能够用余4记数法表示的最大值是3。）

1.7 节

1. a. $\frac{5}{8}$　b. $3\frac{1}{4}$　c. $\frac{9}{32}$　d. $-1\frac{1}{2}$　e. $-\frac{11}{64}$
2. a. 01101011　b. 01111010（截断误差）
   c. 01001100　d. 11101110　e. 11111000（截断误差）
3. 01001001 $\frac{9}{16}$比00111101 $\frac{13}{32}$大。下面是确定哪个位模式表示更大值的一个简单方法。
   第一种情况：如果符号位不同，那么符号位为0的更大。
   第二种情况：如果两个符号位都是0，那么从左至右扫描这两个位模式的其余部分，一直到发现某一个二进制位，其位置上的位模式不同。在这个位置上包含1的位模式表示较大的值。
   第三种情况：如果两个符号位都是1，那么从左至右扫描这两个位模式的其余部分，一直到发现某一个二进制位，其位置上的位模式不同。在这个位置上包含0的位模式表示较大的值。这个比较过程的简单性是采用余码记数法（而不是二进制补码）表示浮点系统指数的一个原因。
4. 最大的数值是 $7\frac{1}{2}$，由位模式01111111表示。关于最小的正值，你们可以认为有2个“正确”答案。首先，如果你坚持文中所描述的编码过程，它要求小数部分的最高有效位必须为1（称为规格化格式），答案则为 $\frac{1}{32}$，由位模式00001000表示。不过大多数机器并不对接近0的值施加这样的限制，因此这时候的正确答案是 $\frac{1}{256}$，由位模式00000001表示。

1.8 节

1. 行程长度编码、频率相关编码、相对编码和字典编码。
2. 121321112343535
3. 彩色卡通是由边框清晰的单色块构成的，而且所包含的颜色数目是有限的。
4. 不是，GIF和JPEG都是有损压缩系统，也就是说，图像中的细节可能会丢失。
5. JPEG基准标准利用了这样一个事实：人眼对于颜色变化不如对光线的变化敏感。因此，减少表示颜色信息的位数，而没有明显地影响图像质量。
6. 暂时模糊和频率模糊。
7. 对信息编码时都要取近似值。对于数字数据，这些近似值在计算过程中会积累，这可能导致错误的结果。而对于图像和声音，取近似值就不会导致那么严重的后果，因为这些被编码的数据只是进行存储、传输以及再现。不过如果图像和声音反复地再现、存储，然后再编码，那么这些近似值就会积累，因此最终导致无用的数据。

1.9 节

1. b、c和e。
2. 有。如果在一个字节中出现了偶数个错误，那么奇偶校验技术就无法检测到它们。
3. 在这种情况下，问题1中的a和b字节中出现了错误。问题2的答案是一样的。

4. a. 001010111 001101000 101100101
   101110010 101100101 000100000
   001100001 101110010 101100101
   000100000 001111001 101101111
   001110101 100111111
   b. 100100010 101001000 101101111
   101110111 100111111 100100010
   000100000 001000011 001101000
   101100101 101110010 001111001
   101101100 000100000 001100001
   001110011 001101011 101100101
   001100100 100101110
   c. 000110010 100101011 100110011
   000111101 100110101 100101110
5. a. BED b. CAB c. HEAD
6. 一个解如下:
   A 0 0 0 0 0
   B 1 1 1 0 0
   C 0 1 1 1 1
   D 1 0 0 1 1

第 2 章

2.1 节

1. 对于一些机器，这个过程包含两步：首先是从第一个单元读取内容到寄存器，然后将内容从寄存器写入目标单元。对于大多数机器，这个过程只是当作一个活动被实现的，而不需要中间寄存器。
2. 要写入的值、要写入的单元的地址以及要写入的命令。
3. 通用寄存器用于存储操作中马上用到的数据，主存储器用于存储不久就要用到的数据，海量存储器用于存储暂时不会用到的数据。

2.2 节

1. move常用来表示从一个位置移到另外一个位置，因此后面留下一个空位。不过，在一个机器中大多数情况下是不会发生这种移动的。相反，被移动的目标通常是被复制（或克隆）到新的位置。
2. 称为相对寻址的常用技术指的是跳多远而不是跳到哪里。例如，一条指令可能向前跳3条指令，或者向后跳2条指令。不过，你应该知道，如果后来在转移（JUMP）指令的起点及目的地之间加入了额外的指令，那么就要改变这些指令了。
3. 从两方面讲都可以。这条指令是以条件转移的形式声明的。不过，由于0等于0这样的条件总是可以满足的，所以这个转移一定会发生，就好像根本没有条件一样。你经常会遇到带有这

种指令的机器，因为这样的设计有效。例如，如果一台机器设计成可以执行带有“if…jump to...”结构的指令，那么这个指令就既可以用于表示条件转移，也可以表示无条件转移。

4. 156C=0001010101101100
   166D=0001011001101101
   5056=0101000001010110
   306E=0011000001101110
   C000=1100000000000000
5. a. 将寄存器6的内容存入（STORE）地址为8A的存储单元。
   b. 如果寄存器A的内容与寄存器0的内容相同，转移（JUMP）到位置DE。
   c. 将寄存器3和寄存器C的内容进行与（AND）运算，并将结果存入寄存器0。
   d. 将寄存器F的内容移动（MOVE）至寄存器4。
6. 指令15AB要求CPU查询存储电路，查找地址为AB的存储单元的内容。当这个值从内存中获得时，要存入寄存器5。指令25AB并没有这样的存储器要求，而是将值AB存入寄存器5。
7. a. 2356　b. A503　c. 80A5

2.3 节

1. 十六进制值34。
2. a. 0F　b. C3
3. a. 00　b. 01　c. 4次
4. 它停机了。这是一个通常称为自修改代码的例子。也就是说，程序可以自我修改。需要注意的是，前两条指令将十六进制C0存入存储单元F8，接下来的两条指令将00存入存储单元F9。因此，当机器到达地址为F8的指令时，停止指令（C000）已经在那里了。

2.4 节

1. a. 00001011　b. 10000000　c. 00101101
   d. 11101011　e. 11101111　f. 11111111
   g. 11100000　h. 01101111　i. 11010010
2. 00111100，AND运算
3. 00111100，XOR运算
4. a. 如果该串包含偶数个1，最后结果就为0；否则为1。
   b. 结果是偶校验的校验位的值。
5. 逻辑XOR运算类似于加法，除了两个操作数都为1的情况，这时XOR运算的结果为0，而加法运算的结果为10。（于是XOR运算可以被看做是没有进位的加法运算。）
6. 用掩码11011111进行AND运算，可以将小写改成大写。用00100000进行OR运算，可以将大写改成小写。
7. a. 01001101　b. 11100001　c. 11101111
8. a. 57　b. B8　c. 6F　d. 6A
9. 5
10. 用二进制补码为00110110，用浮点记数法为0101110。关键点是：由于位模式表示的不同，用于值相加的过程也就不同。

11. 一个解如下：
    12A7（将地址为7的存储单元的内容加载（LOAD）到寄存器2。）
    2380（将值80加载（LOAD）到寄存器3。）
    7023（将寄存器2和寄存器3的内容进行OR运算，并将结果存入寄存器0。）
    30A7（将寄存器0的内容存入（STORE）地址为A7的存储单元。）
    C000（停止。）
12. 一个解如下：
    15E0（将地址为E0的内容加载（LOAD）到寄存器5。）
    A502（将寄存器5的内容向左循环移动（ROTATE）2位。）
    260F（将值0F加载（LOAD）到寄存器6。）
    8056（将寄存器5和寄存器6进行AND运算，并将结果存入寄存器0。）
    30E1（将寄存器0的内容存入地址为E1的存储单元。）
    C000（停止。）

2.5 节

1. a. 37B5
   b. 100万次
   c. 不能。一个典型的文本页包含不超过4000个字符。因此，每分钟打印5页文本的能力表示，其打印速率不超过每分钟20 000个字符，这是远远低于每秒钟100万个字符的速率。（关键点是，计算机传输给打印机字符的速度要远远超过打印机打印的速度，因此，打印机需要一种告知机器等待的方式。）
2. 该磁盘每秒钟要转50转，这就意味着在一秒钟之内，有800个扇区要通过读/写磁头。因为每个扇区包含1024字节，所以通过读/写磁头的二进制位速度大约为6.5 Mbit/s。因此，如果控制器打算与磁盘中读取到的数据同步，那么控制器与磁盘驱动器之间的通信速度至少要这么快。
3. 用Unicode码表示的300页的小说大概有2 MB，即16 000 000位。因此，如果要以54 Mbit/s的速度传输整部小说，大约需要0.3 s。

2.6 节

1. 该管道会包含指令B1B0（正在执行）、5002甚至B0AA。如果寄存器1中的值与寄存器0中的值相等，那么就会执行向地址B0转移的指令。那么对于流水线中指令所做的努力则白费了。另一方面，并没有浪费时间，因为对于这些指令所做的努力并没有花费额外的时间。
2. 如果不采取任何预防措施，那么在该程序的前面部分有机会对存储单元F8和F9进行修改之前，这两个单元的信息已经作为指令被读取出来了。
3. a. 试图给该单元加1的CPU可以在该单元中首先读取值。接着，另外一个CPU读取该单元的值。（注意，在这个时候，两个CPU检索到的是相同的值。）如果在第一个CPU完成它的加法并将其结果写入该单元之后，第二个CPU才完成它的减法，并记下结果，那么单元中最后的值只是反映了第二个CPU的活动。
   b. 两个CPU可能还像以前一样从存储单元中读取数据，但是，第二个CPU这次可能会在第一个CPU之前写下结果。因此，该单元的最后值就只反映了第一个CPU的活动。

第3章

3.1 节

1. 一个传统的例子是，人们为购买门票而排的队列。在这种情况中，可能某些人想“插队”，这就破坏了FIFO结构。
2. 选项（b）、（c）和（e）。
3. 嵌入式通常着重处理专用任务，而PC是通用型计算机。嵌入式系统与同一时期的PC相比，资源更加有限，但在人为干预最少的情况下，嵌入式系统可能面临严格的截止日期。
4. 分时是指多个用户同时访问一台机器，多任务是指一个用户同时执行多项任务。

3.2 节

1. 外壳（shell）：与机器环境进行通信。
   文件管理程序：协调机器的海量存储器的使用。
   设备驱动程序：处理与机器的外围设备的通信。
   内存管理程序：协调机器主存的使用。
   调度程序：协调系统中的进程。
   分派程序：控制进程的CPU时间的分配。
2. 它们之间的界线比较模糊，其差别通常在于持有者的观点。粗略来说，实用软件完成的是基本的、通用的任务，而应用软件则通常只完成针对机器某一特定应用的任务。
3. 虚拟存储器是虚构的存储空间，其表面上的存在是通过这样的过程创建的，即数据和程序在主存和海量存储器之间来回交换。
4. 当机器接通电源时，CPU就开始执行驻留在ROM中的引导程序。这个引导程序引导CPU完成这样一个过程，即把操作系统从海量存储器传送到主存的易失存储区内。当这个传送操作完成时，引导程序就指引CPU跳转至操作系统。

3.3 节

1. 程序是指令的集合，而进程是遵循这些指令的操作。
2. CPU完成它的当前机器周期，保存当前进程的状态，并把它的程序计数器设为一个预定的值（即中断处理程序的位置）。这样一来，将要执行的下一条指令就是中断处理程序中的第一条指令。
3. 一种方法是给它们较高的优先权，使它们可以被分派程序优先考虑。另一种方法是给优先权较高的进程以较长的时间片。
4. 如果每个进程都用完它的整个时间片，那么机器每秒钟可以给大约20个进程提供完整的时间片。如果一些进程没有用完它们的时间片，那么这个进程数目的值可能还会大些，但是花在实现进程上下文切换所需的时间可能会更多（见第5题）。
5. 总共$\frac{5000}{5001}$的机器时间实际将花在进程执行中。然而，当一个进程请求一个I/O活动时，它的时间片在控制器完成这个请求时就终止了。这样一来，如果每个进程都在它的时间片只用了1μs时就作出这样的请求，那么机器的效率将降至$\frac{1}{2}$。也就是说，机器执行进程上下文的时间切换和执行进程的时间一样多。

3.4 节

1. 这个系统保证，该资源一次不会被多个进程使用。然而，它表明了该资源是严格按照交替方式分配的。一个进程用完并释放了这个资源以后，那么如果该进程要再次访问这个资源，它就必须得等待其他进程用完这个资源。即使是第一个进程马上需要这个资源，而其他进程在一段时间内不需要这个资源，情况依然如此。
2. 如果两辆汽车同时进入这个隧道的两端，那么它们都不知道对方的存在。汽车进入和灯的打开过程是临界区的另一个例子，或者说，在这种情况下，我们可以称它为临界过程。在这个术语中，我们可以概括出这个系统的缺点，即隧道两端的汽车能够同时执行临界过程。
3. a. 这保证了不可共享的资源不能被部分地请求和分配；也就是说，要么将整个桥梁给一辆汽车，要么就什么也不给。
   b. 这就意味着：不可共享的资源可以被强制收回。
   c. 这就使得不可共享资源成为可共享的，这样就消除了竞争。
4. 箭头序列在这个有向图中形成了一个闭环。根据这个观察的结果，已经开发出了一些技术，使得某些操作系统能够识别出死锁的存在，并据此采取适当的改正措施。

3.5 节

1. 姓名和日期被认为是不好的候选对象，这是因为它们是常用的选择，所以容易成为密码猜测者的目标。使用完整的单词也不好，这是因为密码猜测者能够很容易编写一个程序，用来尝试字典里的所有单词。而且，也不鼓励使用只包含字符的密码，这是因为它们是由有限的字符集中构成的。
2. 利用两位构成的不同位模式的数目是4。如果需要更多的特权级别，那么设计者至少需要三位来表示不同的级别，这样一来，总共就有8个级别可供选择。按照同样的方式，对于少于4个特权级别的自然选择就会是2，它是用一位可表示的位模式的数目。
3. 这个进程可以更改操作系统程序，使得分派程序把每个时间片都分配给该进程。

第 4 章

4.1 节

1. 开放式网络是这样的一种网络：它的规格说明以及协议是公开的，于是不同的厂商可以生产相互兼容的产品。
2. 两者都是通过连接两个总线以形成一个更大的总线网络。不过网桥只传输那些目的地是该网桥另一端的信息，而交换机有多个连接，每个连接都可以充当网桥。
3. 路由器是一台把网络连结起来，用来传送消息的设备。
4. 邮购业务以及它的客户，银行出纳员以及银行的客户，或者药剂师以及他的顾客。
5. 在交通流量、口头电话通信以及礼仪上有许多的协议。

4.2 节

1. 第一层ISP和第二层ISP提供因特网通信的核心设备，而因特网接入服务提供商则为客户提供到核心设备的接入服务。

2. DNS收集因特网上的域名服务器的信息，以此来把助记网址转换成IP地址（也可以由IP地址转换成助记网址）。
3. 3.6.9表示3字节的位模式000000110000011000001001。位模式0001010100011100用点分十进制法表示为21.28。
4. 这个问题可能有几个答案。其中一个就是，它们都是从特殊到一般。助记形式的因特网地址都是以一台特定机器的名字开始，然后是TLD的名字。邮政地址是从个人的名字开始，然后逐渐到比较大的区域，例如城市、州、国家。这个顺序与IP地址是相反的，它最开始是标识域的位模式。
5. 域名服务器有助于将助记地址翻译成IP地址。邮件服务器传输、接收以及存储邮件信息。FTP服务器提供文档传输服务。
6. SSH提供加密以及认证。
7. 它们使得初始服务器从向每个客户端发送单个消息的负担中解脱出来，P2P方法把这个负担转移给客户(端)，而多点传送则把这个负担转移给因特网中的路由器。
8. 应当考虑的标准可能包括成本、便携性、将计算机用作手机的实用性、保留任何现有模拟电话的需要（如紧急911服务），以及各种相关提供商的可靠性和服务领域。

4.3 节

1. URL本质上是一个文档在万维网上的地址。浏览器是帮助用户访问超文本的程序。
2. 标记语言是在文档中加入解释信息的一个系统。
3. HTML是一种特殊的标记语言。XML是产生标记语言的标准。
4. a. `<html>`表示一个HTML文档的开始。
   b. `<head>`表示一个段落的结束。
   c. `</p>`表示一个文档主体的结束。
   d. `</a>`表示链接到另一个文档的项目的结束。
5. *客户端*与*服务器端*是两个术语，用于标识一个活动是在客户的机器上实现的，还是在服务器的机器上实现的。

4.4 节

1. 链路层接收报文并将其传给网络层。网络层决定报文的转发方向，并将报文传送回链路层再进行新的转发。更高的分层并不是路由选择所必需的，但高级的路由器可能会使用传输层或应用层来提供额外的服务，如选择性过滤或服务的分层性。
2. 不同于TCP，UDP是一个无连接协议，它不能保证报文会在目的地被接收到。
3. 传输层使用传输协议端口号来确定应用层中的哪个单元应当接收到来的消息。
4. 实际上没有办法。任何主机的程序员都可以修改该主机的软件，以保存这些记录。这就是给敏感数据加密的原因。

4.5 节

1. 钓鱼是指伪装成合法实体（如用户的银行或校园IT部门），就用户的密码、信用卡号等通过电子邮件寻问用户，以此方式来获得机密信息的一种手法。计算机不能抵御钓鱼术，而当用户向未经正确核实的他人透露机密数据时，他必须依赖自己良好的判断能力。
2. 一个区域的网关是一个路由器，它只转发传递进来的包（报文的一部分）。因此，网关上的防火墙不能通过它的内容而只能通过它的地址信息过滤通信流。

3. 使用口令可以保护数据（当然也包括信息）。加密的使用可以保护信息。
4. 对于公钥加密系统，知道报文如何被加密并不能够对报文进行解密。
5. 这些问题是国际化的，因此不隶属于某一个政府的法律。此外，法律补救只是给那些受伤害人一些帮助，并不能真正防止伤害。

第 5 章

5.1 节

1. 进程是执行算法的活动。程序是算法的表示。
2. 在绪论里，我们引证了演奏音乐、操作洗衣机、构造模型、表演魔术以及欧几里得算法等算法。在日常生活中遇到的许多“算法”按照我们的正式定义都不能算是算法。本书引证的长除算法就是一个例子。另一个例子是时钟执行的算法：它的指针日复一日地走动，奏鸣钟声。
3. 非正式定义没有要求步骤是有序的和无歧义的，它只在要求里暗示，步骤是可执行的且能终止的。
4. 这里存在两点。一是这些指令定义了一个不可终止的过程。但事实上，这个过程最终到达这样的状态：你的口袋里没有硬币。实际上，这可能是个初始状态。二是算法是有歧义的。这个算法正像所表示的，它没有告诉我们在这种情况下该怎么做。

5.2 节

1. 以物质的组成为例。在一个层面上，原语被认为是分子，而分子实际是由原子组成的，原子又是由电子、质子和中子组成的。今天，我们知道，甚至这些“原语”也是合成物。
2. 一个过程被正确地构建以后，它就可以用作较大程序结构的构件块，不必再重新考虑该过程的内部构成。

3. X ← 较大的输入;
   Y ← 较小的输入;
   while(Y不是0) do
   （Remainder ← X被Y除后的余数;
   X ← Y;
   Y ← Remainder);
   GCD ← X

4. 光的所有其他颜色都可由红、蓝和绿组合产生。所以，电视机的显像管被设计成能产生这三种基色。

5.3 节

1. a. if (n = 1 or n=2)
   then (答案是含有一个值n的列表)
   else (n除以3，得到商q和余数r.
   if (r=0)
   then (答案是含有q个3的列表)
   if (r=1)
   then (答案是含有（q-1）个3和两个2的列表)
   if (r=2)

```
            then (答案是含有q个3和1个2的列表)
        )
```

b. 结果是含有667个3的列表。

c. 用小的输入值来试验，直到看出一个模式。

2. a. 可以。提示：把第一个棋子放在中心，这样使得覆盖其他各个象限的一个正方形时它能避免该象限含有那个洞。每个象限是原来问题的较小版本。

b. 有一个洞的棋盘含有$2^{2n}-1$个正方形，而每个棋子实际覆盖3个正方形。

c. 知道一个问题的解如何能够帮助解决其他问题？问题a和问题b提供了极好的例子。见Ploya的第4阶段。

3. This is the correct answer.

4. 简单地设法去拼装图片是一个自底向上的方法。然而，通过观察拼图盒来看图形是什么样子，为你的方法增加了自顶向下的成分。

5.4 节

1. 把`while`语句的测试修改为“目标值不等于当前表项并且还有表项要检查”。

2.
```
Z ← 0;
X ← 1;
repeat ( Z ← Z + X;
         X ← X + 1)
until (X = 6)
```

3. 这是C语言中的一个问题。当关键字`do`距`while`若干行时，读程序的人常常会在对`while`语句的正常解释上遇到障碍。特别是，一个`do`语句结尾处的`while`常常被解释为一个`while`语句的开始。所以，经最好使用不同的关键字来表示先测试循环结构和后测试循环结构。

4.

Chery1	Alice	Alice
Gene	Chery1	Brenda
Alice	Gene	Chery1
Brenda	Brenda	Gene

5. 坚持把主元放到列表里一个相同表项的上面是浪费时间。例如，按建议进行修改，然后对所有表项都相同的列表试用这个修改后的新程序。

6.
```
procedure sort (List)
N ← 1;
 while(N小于List的长度) do
    (J ← N + 1;
        while( J不大于List的长度) do
           ( if (位置J里的表项小于位置N里的表项)
               then(交换两个表项)
           J ← J + 1)
N ← N + 1)
```

7. 下面这个解决方案的效率不是很高，你能使其效率更高吗？

```
procedure sort (List)
N ← List的长度;
While (N大于1) do
```

```
    (J ← List的长度;
      while(J大于1) do
        ( if (位置J里的表项小于位置J-1里的表项)
          then(交换两个表项)
    J ← J - 1)
N ← N - 1)
```

5.5 节

1. 考虑的第一个名字是Henry，接下来是Larry，最后是Joe。
2. 8，17
3. 1，2，3，3，2，1
4. 终止条件是“N大于等于3”（或“N不小于3”）。该条件的前提是没有创建额外的激活。

5.6 节

1. 如果该机器1 s可以排序100个名字，那么它1 s可以进行$\frac{1}{4}(10\,000-100)$次比较。这意味着，每次比较所花费的时间近似于0.0004 s。因此，排序1000个名字平均需要$\frac{1}{4}(1\,000\,000-1\,000)$次比较，大概需要100 s或$1\frac{2}{3}$ min。
2. 二分搜索法属于$\Theta(\lg n)$、顺序搜索法属于$\Theta(n)$，而插入排序法属于$\Theta(n^2)$。
3. $\Theta(\lg n)$ 类是效率最高的，接着是$\Theta(n)$、$\Theta(n^2)$和$\Theta(n^3)$。
4. 回答是不正确的，尽管听起来似乎是对的。事实是3张卡片中有两张两面是一样的。于是，取得这样一张牌的概率是2/3。
5. 不正确。如果被除数小于除数，如3/7，给出的答案是1，尽管正确结果应该是0。
6. 不正确。如果X的值是0，而Y的值不是0，那么所给出的答案是不正确的。
7. 每次构建终止测试时，语句“Sum=1+2+…+K并且K小于等于N”为真。把它与终止条件“K大于等于N”合并产生所预期的结论“Sum=1+2+…+N”。因为K被初始化为0，并且每进行一次循环K的值就增加1，所以它的值最终一定达到N。
8. 不能保证。不是硬件和软件设计所能控制的问题，如机械故障和电气问题等，都会影响计算。

第 6 章

6.1 节

1. 一个用第三代语言编写的程序，从某种意义上说它是独立于机器的，因为它的运行步骤不是按照寄存器和存储单元地址这样的机器特征来描述的。在另一方面，从某种意义上说，它又是依赖于机器的，因为算数溢出和截断误差还是会出现。
2. 主要差别是，汇编程序把源程序里的每条指令只翻译为一条机器指令，而编译器往往要产生多条机器语言指令才能等价于一条源程序指令。
3. 说明性范型基于开发所要解决的问题的描述。函数式范型使程序员根据较小问题的解决方案来描述待解决问题的解决方案。面向对象范型则强调描述问题的环境里的成分。
4. 与前几代语言相比，第三代语言更多是用问题的环境来表达程序，很少用计算机细节来表达。

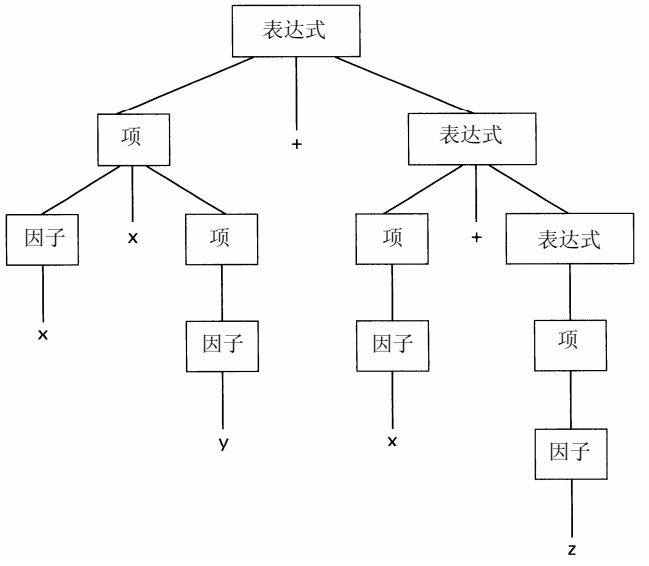
6.2 节

1. 使用描述性常量可以改进程序的可理解性。
2. 声明语句描述术语，命令语句描述算法中的步骤。
3. 整型、实型、字符型和布尔型。
4. `if-then-else`和`while`循环结构很常见。
5. 同构数组所有的成分有同样的类型。

6.3 节

1. 变量的作用域是指变量在程序中可使用的范围。
2. 函数是这样的一个过程，它返回一个与函数名相关联的值。
3. 因为它们就是这样的。I/O操作实际上是对该机器操作系统内例程的调用。
4. 形参是过程内的标识符。它是实参这个值的占位符，在该过程被调用时，实参才传递给该过程。
5. 过程用于执行一个操作，而函数用于产生一个值。于是，如果过程的名字反映它所执行的操作，函数名反映它所产生的值，那么这个程序就更可读。

6.4 节

1. *词法分析*：识别标记的过程。
   语法分析：识别程序的语法结构的过程。
   代码生成：产生目标程序指令的过程。
2. 符号表是语法分析程序从程序的声明语句中得到的信息的记录。
3. 

4. 符合Chacha结构的字符串由一个或几个下述子串构成：
   forward backward cha cha cha
   backward forward cha cha cha
   swing right cha cha cha
   swing left cha cha cha

6.5 节

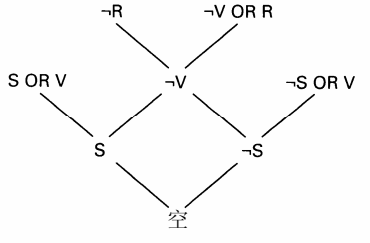
1. 类是对象的描述。
2. 大概是MeteorClass，由它可构造各种流星。在类LaserClass内，可以找到名为AimDirection的实例变量，它指示激光瞄准的方向。这个变量大概会用在fire、turnRight和turnLeft等方法中。
3. 类Employee可以包含与雇员的姓名、住址、服务年限等有关的特性。类FullTimeEmployee可以包含与退休津贴有关的特性。类PartTimeEmployee可以包含与每周工作的小时数和每小时佣金等有关的特性。
4. 构造器是类里的一个特殊方法，它在创建该类的一个实例时执行。
5. 一个类里的某些项被指定为私有，以防止其他程序单元直接访问这些项。如果一个项是私有的，那么修改这个项的影响应该限于这个类的内部。

6.6 节

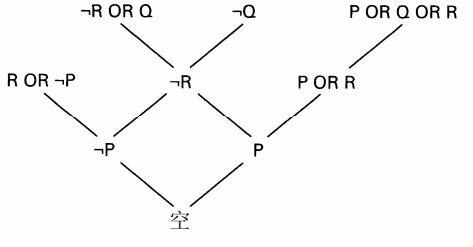
1. 包含初始化并发进程执行的技术以及实现进程间通信的技术。
2. 一个方法是把负担放在进程上，另一个方法是把负担放在数据上。后者的好处是把任务集中在该程序的一个点上。
3. 这包括天气预报、空中交通管制、复杂系统（从核反应到行人交通）的模拟、计算机网络以及数据库维护。

6.7 节

1. *R*、*T*和*V*。例如，我们可以证明，*R*是将¬*R*加到这个集合的结果，并且能够证明这个解可以得到空语句，证明如下：



2. 不是。这个集合是不一致的，因为消解可以得到空语句，证明如下：



3.
```
mother(X, Y) :- parent(X, Y), female(X)
father(X, Y) :- parent(X, Y), male(X)
```

4. Prolog将得出结论：carol是她的同胞。为了解决这个问题，规则需要包括x不能等于y这样的事实，在Prolog中写成`x \= y`。这样规则的改进版本是：

```
sibling(x, y) :- x \= y, parent(z, x), parent(z, y)
```

意思是：如果x和y不相同且其父母中有一方相同，那x就是y的同胞。下面的版本则坚持认为只有当x和y的父母双方都相同，那他们才是同胞：

```
sibling (x, y) :- x \= y, z \= w
                  parent (z, x), parent (z, y)
                  parent (w, x), parent (w, y)
```

第7章

7.1 节

1. 一长串赋值语句序列并不比设计成几个嵌套的if语句复杂。
2. 在使用了一段时间后，发现错误的数目为多少？这里的一个问题就是不能预先估算出这个值。
3. 这里的关键问题就是要考虑如何能对软件的属性进行度量。用于估算一款软件中错误数目的一种方法是，在设计这个软件时故意放进一些错误。然后，在认为软件已调试后，检查一下，看原先的错误还存在多少。例如，如果你在软件中故意放入7个错误，在调试后消除了5个错误，那么你就可以推测，软件中错误的总数只有$\frac{5}{7}$被排除。
4. 可能的答案包括：度量的发现、预制构件的开发、CASE工具的开发以及向标准靠近。另一个是（稍后将在7.5节介绍）建模和UML等符号系统的开发。

7.2 节

1. 在开发阶段稍作努力就能为将来的维护工作带来很多便利。
2. 需求分析阶段的重点在于，所推荐的系统必须实现哪些功能，设计阶段的重点在于系统完成这些目标的方式，实施阶段的重点在于系统的实际建设，而测试阶段的重点在于，保证建成的系统遵循原定的目标。
3. 软件需求规格说明文档的作用是：为客户和软件工程公司，在所要开发软件的需求和规格说明问题上达成一致而编写的文档。

7.3 节

1. 传统的瀑布方法要求需求分析、设计、实现和测试阶段按照线性方式实现，而较新的模型则是一种更为宽松的反复试验、不断探索的方法。
2. 增量模型、迭代模型和XP如何？
3. 传统的演化式原型开发是开发软件的组织所实现的，而开放源码开发的方法并不限制在一个组织内。在开放源码开发的情况中，管理软件开发过程的人没有必要决定报告哪些增强，而在传统的演化式原型开发中，管理软件开发的人员要为员工分配明确的增强软件的任务。
4. 对你而言，这是你要考虑的。如果你是软件开发公司的管理者，你能够对你的公司要销售的软件采用开放源码的方法进行开发吗？

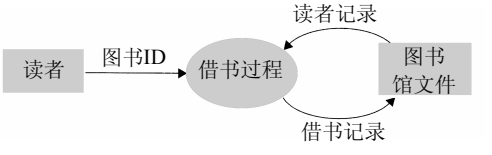
7.4 节

1. 一部小说的各章之间相互依存，而一部百科全书各个章节之间很大程度上是相互独立的。所以，小说的章节之间比百科全书的章节之间有更大的耦合度。然而，百科全书中的章节可能比小说里的章节有更高的内聚度。

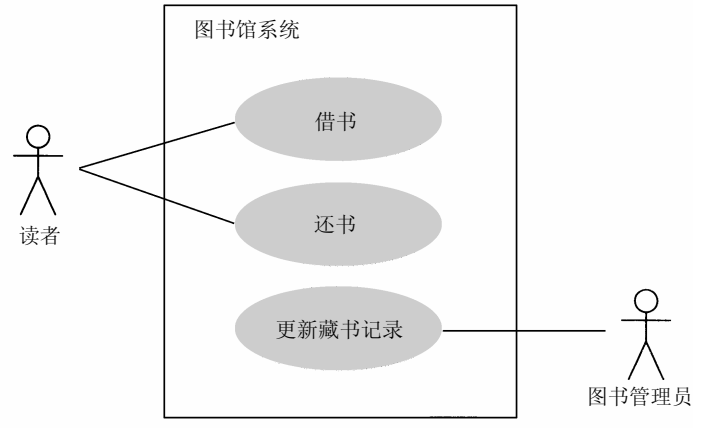
2. 累计积分可能是数据耦合的一个例子。其他可能存在的“耦合”包括疲劳、冲力、对对手策略的了解，可能还有自信心。在许多体育运动项目中，因一局比赛的结束而重新开始下一局，局的内聚度会增加。例如，在棒球运动中，一个队即使以满垒结束了上轮击球，每次轮到击球也都不以跑垒选手开始。在其他有些运动中，各单元分别记分，如网球比赛中，每局的输赢与其他局无关。
3. 这是一个难题。从一种观点来看，我们可以从把每件事情放在单个模块中开始。这就造成了较低的内聚度而根本没有耦合。然后，如果开始把单一模块分割成一些较小的模块，结果就会使得耦合度增加。所以可能会得出结论：内聚度的增加易于导致耦合度的增加。从另一方面讲，假设手头上的问题自然地分割成3个内聚度较高的模块，称为*A*、*B*和*C*。如果原始的设计没有注意到这种自然的分割（例如，*A*的一半任务可能与*B*的一半任务放在了一起，等等），我们希望内聚度低而耦合度高。在这种情况下，重新设计系统，将任务*A*、*B*和*C*分离至不同的模块中，这就极有可能在增加模块内的内聚度的同时降低模块间的耦合度。
4. 耦合是模块之间的链接，内聚则是模块内部的连接关系。信息隐藏是信息共享的约束。
5. 你可以添加一个箭头，用来说明`ControlGame`模块必须告知`UpdateScore`模块，谁赢得了比赛。然后，再在其另外一个方向上添加一个箭头，用来说明当`UpdateScore`模块把控制权移交到`ControlGame`模块时，它就将报告当前的状态（如“局结束”或“比赛结束”等）。
6. 删除图7-5中除第一个和最后一个以外的所有水平箭头。也就是说，裁判应该评判`PlayerA`的发球，并且直接将`updateScore`消息发送给`Score`。（当然，这也就忽视了第二发球的机会。考虑到双误时，应该如何修改程序设计呢？）
7. 传统的程序员是在语句的基础上写程序，如第6章所介绍的；而组件装配员则是通过链接称为构件的预制块来构建程序。

7.5 节

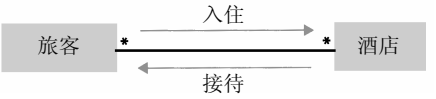
1. 首先确定的是，你的图处理的是数据流（不是书本的流动）。下图就表明：图书ID（来源于读者）和读者记录（来源于图书馆文件）结合成借书记录，并存放在图书馆文件中。



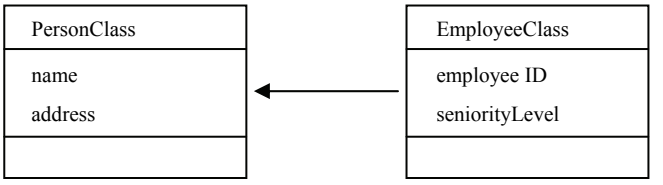
2.



3.



4.



5. 如图7-13所示，在图的周围画一个矩形，并且在左上角加上“sd”标识。
6. 设计模式为实现反复出现的软件主题提供了标准的、成熟的方法。

7.6 节

1. SQA（软件质量保证）小组负责监督和实施组织所采用的质量控制系统。
2. 人们一般不会记录他们在一个项目中所采取的步骤措施（如决定、操作等）。（这里还存在性格冲突、嫉妒、自我冲突等问题）
3. 记录保存和评审。
4. 软件测试的目的是为了找出错误。那么，从这个意义上讲，没有发现错误的测试就是失败的。
5. 办法之一就是考虑模块中的分支数。例如，一个过程模块，它包含了大量的循环和`if-then-else`语句，那么它很可能比一个带有简单的逻辑结构的模块更容易出错。
6. 边界值分析会建议你用一个有100个数据项的表和一个没有数据项的表对这个软件进行测试。你还可以用一个已经排好序的表进行测试。

7.7 节

1. 文档采用的形式有用户文档、系统文档和技术文档。通常出现在随付的手册中、以注释形式出现在源程序和编写规范的代码中，或者以程序写到终端上的交互消息的形式出现，还有可能是数据字典、结构图、类图、数据流图和实体-联系图之类的设计文档。
2. 在开发阶段和修改阶段。问题在于，修改必须要像原始程序一样完全文档化。（同样，软件在其使用阶段也要文档化。例如，系统的一个用户可能会发现问题，这个问题不是确定的，而仅仅会在系统的用户手册的以后版本中得到反映。而且有时候，有关软件如何使用的书籍是在该软件已经使用了一段时间并开始流行后写的。）
3. 不同的人对此有不同的观点。有些人认为程序是整个项目的关键，所以自然更重要。而另一部分人则认为，如果程序没有文档，则它什么也不是，因为如果你不能理解一个程序，你就不会使用和修改它。而且，如果有良好的文档，创建程序的任务就“容易”被再创造。

7.8 节

1. a. 调节显视器屏幕的倾角或者鼠标形状的能力如何？在智能手机上，用触摸屏来替代鼠标，或者通过倾斜手机来提供输入的做法怎么样？

b. 显示器屏幕上的窗口版面布局如何？包括工具栏、滚动条和下拉菜单的设计。在智能手机上，通过倾斜照相机来对准相关物体的方式难道不是与人们的思考方式一致吗？

2. a. 在智能手机上使用鼠标（甚至手写笔）是不切实际且不方便的。此外，缩小了尺寸的触摸屏要求不重要的元素所占据的空间必须很有限。因此，智能手机上通常没有滚动条。即使有，也将其显示为细线。

   b. 对于我们来说，智能手机屏幕上的滑动触摸是很自然的手势。现实中，我们可能通过在桌面上滑动物体（如纸张等）来移动它们。更进一步说，这比使用台式计算机上的滚动条更为自然。虽然滚动条确实可以按预期移动，但正被滚动的区域会向相反的方向移动。对于从未使用过计算机的人来说，这一行为可能是违反人类直觉的。

3. 你可以回答“人类特征的作用”。另一个不错的答案是界面设计关注的是软件系统的外部特性，而不是内部特性。
4. 文中讨论了三项，即习惯的形成、狭窄的注意力和有限的多重处理能力。你能想到其他的特质吗？做出假设的倾向算不算？

7.9 节

1. 版权声明声明作品的所有权并确定被授权使用该作品的个体。所有的作品（包括需求规格说明、设计文档、源代码和最终的产品）通常都需要大量的投入才能产生。个人或者企业应该采取措施来确保他们的所有权，以及确保其知识产权不被不当使用。
2. 版权法和专利法使社会受益，因为它们鼓励新产品的发明者将产品公之于众。如果没有这些保护，很多公司会犹豫是否对新产品开发进行较多的投资。
3. 免责申明并不能保护公司由于疏忽而给他人造成伤害。

第 8 章

8.1 节

1. 表：运动队队员的名单。
   栈：自助餐厅里的一叠盘子。
   队列：食堂里排的队。
   树：许多政府部门的组织图。
2. 栈和队列可以看成是特殊类型的表。在广义表中，项能在任何位置插入和移除；在栈中，项只能在头部被插入和移除；在队列中，项只在尾部插入，在头部移除。
3. 栈中自顶向下的字母是E、D、B和A，如果有字母出栈，那就是字母E。
4. 队列中从表头至表尾的字母是B、C、D和E，如果有字母从队列中移出，那就是字母B。
5. D和C是叶子（或终端）节点。B一定是根节点，因为所有其他结点都有父节点。

8.2 节

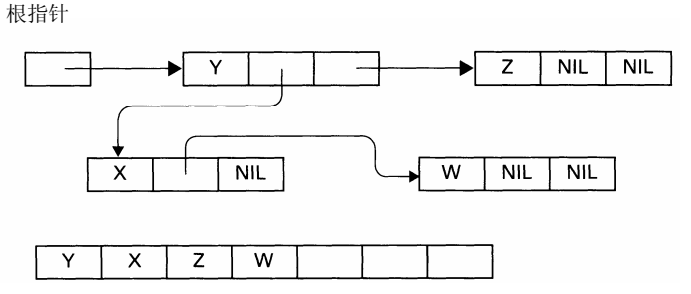
1. 计算机主存中的数据实际上是存储在单独的可寻址的存储单元中的。模仿数组、表和树这样的结构是为了方便数据用户存取数据。
2. 如果你想编写一个下跳棋的游戏程序，那么表示棋盘的数据结构将会是一个静态数据结构，这是因为棋盘的大小在游戏过程中是不会改变的。然而，如果你要编写一个玩多米诺游戏的程序，那么表示根据表构建的多米诺模式的数据结构将会是一个动态数据结构，这是因为这

个模式的大小是可变的，而且不能预先确定。

3. 电话簿实质上是一个用来指向人的指针（电话号码）集合。犯罪现场留下来的线索是（可能加密过了）指向罪犯的指针。

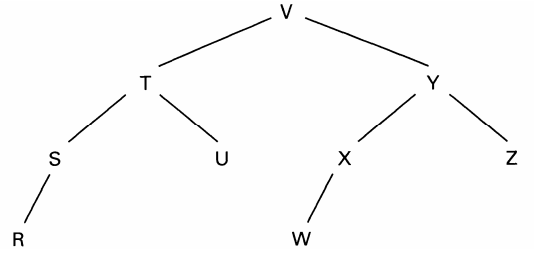
8.3 节

1. 5 3 7 4 2 8 1 9 6
2. 如果R为矩阵的行数，那么公式就为R（$J-1$）+ （$I-1$）
3. （$c \times i$）+j
4. 头指针包含NIL值。
5. Last ← 要打印的最后一个名字
   Finished ← false
   Current Pointer ← 头指针;
   while (Current Pointer非NIL and Finished = false) do
     (打印Current Pointer指向的项,
     if (正在打印的名字=last)
       then (Finished ← true)
    Current Pointer ← 当前指针所指项中该指针单元中的值)
6. 栈指针指向与栈底直接相邻的那个单元。
7. 把栈表示为一个一维数组，并把栈指针表示为一个整型变量。然后利用这个栈指针来维护栈顶在数组中位置的一个记录，而不是实际的内存地址。
8. 空和满这两种情况都是由相等的头指针和尾指针来指示。这样一来，需要一些附加信息来区分这两种情况。
9.

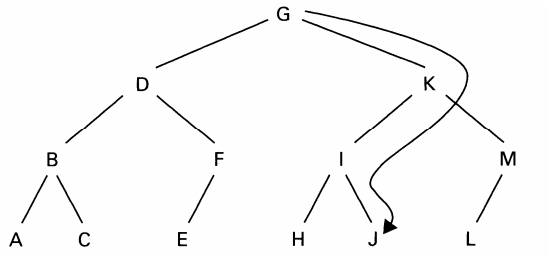


8.4 节

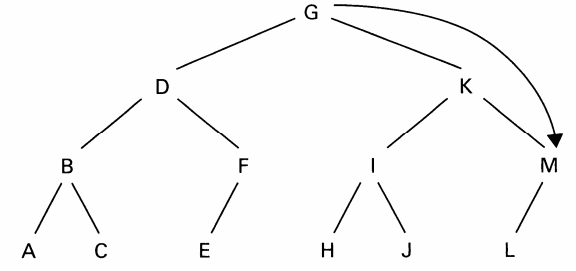
1.



2. 当查找J时：



当查找P时：



3.

```
procedure PrintTree(Tree)

if(Tree的根结点不为NIL)
  then(对以Tree的左分支出现的树应
    用PrintTree过程;
  Print Tree的根结点;
  对以Tree的右分支出现的树应用
    PrintTree过程。)
```

```
procedure PrintTree(Tree)

if(Tree的根结点不为NIL)
  then(对以Tree的左分支出现的树应
    用PrintTree过程;
  Print Tree的根结点;
  对以Tree的右分支出现的树应用
    PrintTree过程。)
```

当K被打印时，在这里

4. 在每个结点上，每个孩子指针可用来表示字母表中的一个字母。沿着表示一个单词的拼写次序的指针序列遍历树的路径，就可以表示一个单词。如果一个结点表示的是拼写正确的单词的结束，那么它就可以用一种特殊的方式进行标记。

8.5 节

1. 类型是一个模板；而这个类型的实例是通过这个模板构建出的一个真实的实体。打个比方，狗是动物的一类，而Lassie和Rex则是该类型的实例。
2. 用户自定义数据类型是数据组织的一种描述，而抽象数据类型则包括了对数据进行处理的操作。
3. 这里的问题在于，你是选择用邻接表还是用链表来实现。你的选择会影响到插入新项、删除旧项以及找到感兴趣的项这些操作过程的结构。然而，该抽象数据类型的实例的用户看不到这种选择。
4. 这个抽象数据类型至少得包含一个数据结构的描述，用来存储账户余额，以及通过支票存取款的过程。

8.6 节

1. 抽象数据类型和类都是构建类型实例的模板。然而，类更为普遍，这就在于它与继承性相关

联，可以只描述一组过程的集合。
2. 类是模板，通过它来构建对象。
3. 这个类可以包含一个循环队列，以及用于添加项、删除项、测试队列为满和测试队列为空的这些操作的过程。

8.7 节

1. a．A5　　b．A5　　c．CA
2. D50F，2EFF，5FFE
3. 2EA0，2FB0，2101，20B5，D50E，E50F，5EE1，5FF1，BF14，B008，C000
4. 一个链表的项由两个存储单元组成（一个数据单元，后跟一个指向下一项的指针），当遍历这个链表时，指令形式DR0S能用来读取数据，而指令形式DR1S可用来检索下一项的指针。如果用指令形式DRTS，那么可以通过修改寄存器T中的值来调整所访问的实际存储单元。

第 9 章

9.1 节

1. 对于库存信息而言，采购部门的兴趣在于存放更多的原材料订单的库存记录，而财务部门则需要平衡账目的信息。
2. 数据库模型提供了数据库的一种组织观点，它更切合于应用的角度，而不是实际组织的角度。所以，数据库模型的定义是数据库作为抽象工具来使用的第一步。
3. 应用软件把用户的请求从应用的术语翻译成数据库管理系统支持的数据库模型的术语。数据库管理系统又把这些请求转化为实际数据库中的操作。

9.2 节

1. a．G. Jerry Smith
   b．Cheryl H. Clark
   c．S26Z
2. 一种解答是：

```
TEMP ← SELECT from JOB
                where Dept = "PERSONNEL"
LIST ← PROJECT JobTitle from TEMP
```

在有些系统中，这会导致表中的职务重复，因其在人事部门出现的次数而异。也就是说，秘书这个职务可能在我们的表中出现很多次。然而，更为常见的方法就是将PROJECT操作设计成能把结果关系中重复的元组去掉。

3. 一种解答是：

```
TEMP1 ← JOIN JOB and ASSIGNMENT
                where JOB.JobId = ASSIGNMENT.JobId
TEMP2 ← SELECT from TEMP1
                where TermDate = "*"
```

```
TEMP3 ← JOIN EMPLOYEE and TEMP2
                    where EMPLOYEE.EmplId = TEMP2.EmplId
RESULT ← PROJECT Name,Dept from TEMP3
```

4.

```
Select JobTitle
  from JOB
  where Dept = "PERSONNEL"
Select EMPLOYEE.Name, JOB.Dept
  from JOB,ASSIGNMENT,and EMPLOYEE
  where (JOb.Job = ASSIGNMENT.JobId) and
      (ASSIGNMENT.EmplId = EMPLOYEE.EmplID)
      and (ASSIGNMENT.TermDate = "*")
```

5. 模型本身并不提供数据独立性。这是数据库管理系统的一个属性。即使是实际的组织发生了变化，只要数据管理系统有能力为应用软件提出始终如一的关系组织，那么就能获得数据独立性。
6. 通过公共属性。例如，本节中的EMPLOYEE关系是通过属性`EmplId`与关系ASSIGNMENT相联系的，而关系ASSIGNMENT是通过属性`JobId`与关系`JOB`相联系的。像这样的用来连接关系的属性，有时候称为连接属性。

9.3 节

1. 应该有对`StartDate`和`TermDate`赋值和检索的方法。另一种方法可能是为报告服务的总时间而提供。
2. 持久对象就是一个无限期存储的对象。
3. 一种方法就是为库存中的每一类产品建立一个对象。每个这样的对象能够维护该产品的总库存量、产品的总成本以及到未交付的产品定单的链接。
4. 正如本节开始所讨论的，就处理复合型数据类型而言，使用面向对象数据库比关系型数据库要容易得多。而且，事实上，对象可包含在解决问题上起着能动作用的方法，而关系数据只包含有数据，因而，这就使得面向对象数据库优于关系数据库。

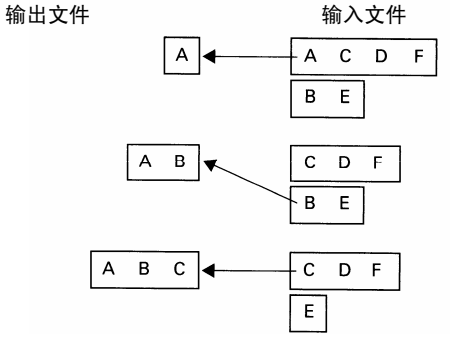
9.4 节

1. 事务到达了它的提交点以后，数据库管理系统就承担起了查看在数据库上执行的完整事务的责任，而没有达到提交点的事务就没有这样的保证。如果出现了问题，就不得不重新提交。
2. 一种方法就是在某一时刻停止交叉的事务，使得当前所有的事务都能完全完成。这就可以建立一个点，在这个点后的级联回滚将终止。
3. 如果一次执行一个事务，可以使账户的最终余额为100美元。如果第一个事务是在第二个事务读取了初始的余额而尚未存入新的余额期间执行的，那么就可以使账户的最终余额为200美元。如果第二个事务是在第一个事务读取了初始的余额而尚未存入新的余额期间执行的，那么就可以使账户的最终余额为300美元。
4. a. 如果没有其他事务互斥访问，则准许共享访问。
   b. 如果另一个事务已经做了某种形式的访问，那么通常数据库管理系统会让新事务等待，或者将其他的事务回滚而让新事务访问。

5. 如果两个事务中的每一个都获得了对不同数据项的互斥访问，然后请求访问另一项，那么就会出现死锁。
6. 通过回滚其中一个事务（利用日志），并让另一个事务访问原来被第一个事务占用的数据项，上述的死锁就能被消除。

9.5 节

1. 你应当经过这样一些步骤：



2. 该算法的思想是：首先将文件进行分割，存放在许多独立的文件中，每个文件包含一条记录。接下来，把这些单记录文件组成对，并对每个对应用归并算法。其结果就使得文件数减半，而每个文件有两条记录。此外，每个这样的双记录文件是排序过的，我们就能再把这些文件组成对，并应用归并算法。于是我们就发现，文件数越来越小，而文件越来越大，每个文件都是排序过的。就这样一直继续下去，我们最终会发现只剩一个文件，且该文件包含所有的初始记录，但已经排过序了。（如果在这个过程的任意一步中出现了奇数个文件，我们只需把那个不成对的文件放在一边，在下一步中将其与一个较大的文件配成对。）
3. 如果文件是存放在磁带或CD上，那么它的实际组织最有可能是顺序的。然而，如果文件是存放在磁盘上，那么它最有可能是分散在磁盘的不同扇区，这时，文件的顺序特性就是一个概念属性，并得到指针系统或者是某种形式的表（表中记录着存放文件的扇区）的支持。
4. 首先找到文件索引中的目标键，从那可以得到目标记录的地址。这样就可以在这个地址上读取记录。
5. 一个选取不恰当的散列算法会比正常情况造成更多的群集，因而也就会出现更多的溢出。因为海量存储器每个部分的溢出组成了一个链表，所以对溢出记录的搜索本质上就是搜索一个顺序文件。
6. 桶的分配如下：

   a. 0　　b. 0　　c. 3　　d. 0　　e. 3

   f. 3　　g. 3　　h. 3　　i. 3　　j. 0

   这样一来，所有的记录都散列到桶0和桶3中，剩下桶1、2、4和5为空。这里的问题就在于所使用的桶数是6，而键值有公因子3。（你可以试着用7个桶来对这些键值重新散列，然后看看有什么改进。）
7. 这里的关键就在于我们实际上使用散列算法将一群人分成365类。当然，散列算法是用来计算每个人的生日。令人惊奇的是，只需23个人，就能出现至少两个人生日相同的情况。从散列文件的角度来看，这就表明当将记录散列到海量存储器中可用的365个桶时，只需要输入23条记录，就可能出现群集。

9.6 节

1. 这是因为在动态数据中搜索模式还存在问题。
2. 类型描述：确定出某种杂志订阅者的特征。
   类型识别：确定出两种杂志订阅者之间不同的特征。
   群集分析：确定出容易吸引类似订阅者的那些杂志。
   关联分析：确定出不同杂志的订阅者与不同的购物习惯间的联系。
   孤立点分析：确定出不符合一般订阅者特征的某杂志的订阅者。
   序列模式分析：确定出杂志订阅方面的趋势。
3. 多维数据集可以让我们从这样几个角度来看待销售数据，如按月销售的角度、按地域销售的角度和按产品种类销售的角度等。
4. 传统的数据库查询是检索存储在数据库中的事实，而数据挖掘则是寻找这些事实中的模式。

9.7 节

1. 这里的问题是将你的回答与下一题的回答进行比较。你就会发现这两个问题实质上是相同的，只是放在了不同的语境中而已。
2. 见上一题。
3. 你可能会收到一些原先不会收到的通知或广告，但是你也可能会成为筹款对象或犯罪目标。
4. 这里的问题就在于新闻自由能使公众警惕滥用或潜在的滥用，这样一来，就能使公众的观点发挥作用。文中所列举的大多数情况，正是新闻自由通过对公众的警告而启动了矫正行动。

第 10 章

10.1 节

1. 图像处理对二维图像进行分析，2D图形学把二维形状转化为图像，3D图形学将三维场景转化为图像。
2. 传统的摄影生成实际场景的图像，而3D图形学生成虚拟场景的图像。
3. 第一步是“构建”虚拟场景，第二步是捕捉图像。

10.2 节

1. 步骤是建模（构建场景）、渲染（生成图像）和显示（显示图像）。
2. 图像窗口是构成图像的投影平面的一部分。
3. 帧缓冲区是包含图像编码版本的存储区域。

10.3 节

1. 它是菱形（被压扁的正方形）。
2. 程序化模型是指导物体构建的程序段。
3. 列出的物体可能包括：青草覆盖的地面、用石头铺的道路、露台、树、灌木林、云、太阳和演员。这里的重点是强调场景图的范围（它包含许多细节）。
4. 用多边形网格表示所有物体，这提供了渲染处理的统一方法。（在大多数情况下，可认为渲染过程渲染的是平面片，而不是渲染物体。）
5. 纹理映射是将二维图像和物体表面相关联的一种方法。

10.4 节

1. 镜面反射光是“直接”从表面反射的光，散射光是从表面“散开”的光，环境光是没有特定光源的光。
2. 裁剪是去除那些与视体不相交的物体（或物体的一部分）的过程。
3. 假设高光应该出现在面片的中间，高光是由于面片这一点的特定表面朝向引起的。因为Gouraud描影只考虑了沿着碎片边界的表面朝向，所以它将丢失高光部分。但是，因为Phong描影试图决定碎片内部的表面朝向，所以它或许能检测高光部分。
4. 渲染流水线提供了渲染的标准化方法，它最终产生了更高效的绘制系统。特别地，渲染流水线可以在固件中实现，这意味着执行渲染过程可以比用传统软件实现任务更快。
5. 这个问题的目的是让你考虑局部照明模型和全局照明模型间的区别，而不是去产生一个预定的答案。你可以提出的潜在解决方案包括：当考虑镜子是透明的或试着把镜子中的图像以一种阴影的形式来处理时，把适当修改过的要反射的物体的副本放置在镜子后面。

10.5 节

1. 我们只对最终达到图像窗口的光线感兴趣。如果我们从光源开始，我们将不知道沿着哪条光线。
2. 分布式光线跟踪试图避免传统光线跟踪多条光线时产生的固有的有光泽的外观。
3. 辅射度是费时的，并且它捕获不到镜面反射光。
4. 光线跟踪和辅射度都实现了全局照明模式，二者都是计算密集型的。但是光线跟踪往往生成有光泽的表面，而辅射度则会生成灰暗的表面。

10.6 节

1. 没有标准的答案。如果图像逗留200 ms，我们每秒投影5帧，每一帧在下一帧被投影的时候刚刚消退完。这将可能导致跳动的图像，虽然观看200 ms跳动的图像并不舒服，但它还是产生了动画效果。（事实上，慢速率可以产生粗糙的动画。）注意，每秒5帧的速率远低于每秒24帧的电影标准。
2. 故事板是所需动画序列的“图示的轮廓”。
3. 补间是创建填充到关键帧间的空白帧的过程。
4. 动力学是机械学的分支，它把运动作为力作用的结果进行分析；运动学是机械学的分支，它分析运动，而不考虑引起运动的力。

第 11 章

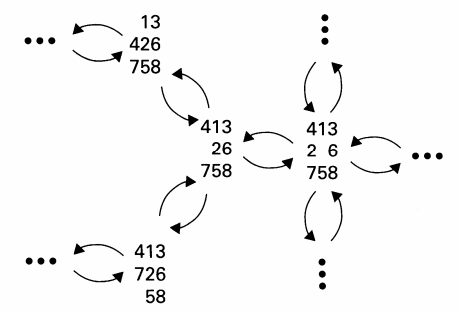
11.1 节

1. 本章介绍的有反射动作、基于真实世界知识的行为、寻找目标的行为、学习以及感知。
2. 这里的目的并非一定要给出明确的回答，而是通过这个问题表明，关于智能存在形式的争论是多么的微妙。
3. 虽然我们大多数人会说不是，但是如果在类似的场景中由人来分发相同的物品，即使区分不出两者的差异，我们大概仍会说人是知道的。
4. 这也不是一个回答是或否的问题。大多数人也许会同意至少机器看上去是有智能的。答案并非简单的对或错。值得注意的是，设计用来模拟人类聊天的程序难以进行有意义的聊天，即使持续的时间很短也不容易做到。这种程序很容易被认出是机器。

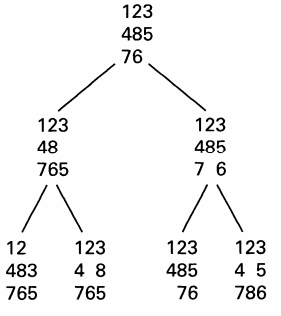
11.2 节

1. 在遥控的情况下，系统只需要转播画面就可以了；而利用画面来控制，机器人就必须能“理解”画面的意思。
2. 这个图的一个部分的各种可能的诠释都不能与其他部分的诠释相匹配。为了把这个见解置入程序，要把各种不同的线条交叉点的解释独立出来，然后写一个程序，力图去找出一组相容的诠释（每个交叉点一个）。其实，如果静下心想一想，这可能就是你在判断这幅画时自己的感觉所做的事情。当你的感觉试图将可能的解释合并在一起时，是否察觉到自己的眼睛在画面两端来回扫视？（若你对此感兴趣，可以阅读D. A. Huffman、M. B. Clowes和D. Waltz等人的著作。）
3. 栈中有4个方块，但只看得见3个。问题是理解这种显然很简单的概念，却需要有大量的“智能”。
4. 有趣，不是吗？这种意思上的微妙差别在自然语言理解领域呈现了具有重大意义的问题。
5. 该句是描述了他们是哪种骑兵，还是在说一些人在干什么？
6. 语法分析产生了同样的结构，但语义分析分辨出，介词短语在第一句里告诉篱笆建在何处，而在第二句里告诉篱笆建在何时。
7. 他们是兄妹。

11.3 节

1. 产生式系统提供了对于各种不同问题的一个统一的方法。也就是说，所有的问题都可以用产生式系统这个术语重新表达为在状态图中寻找一条路径的问题，尽管它们的原始形式有很明显的不同。
2. 

3. 这个树有四层移动深度。上面部分如下图所示：

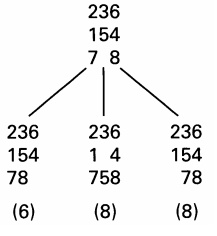


4. 此题需要太多的纸张和时间。
5. 解决八数码游戏的启发式系统基于对即时情况的分析，就像登山者一样。这种短识是使得本节例子中出现的算法一开始就沿着错误的路径进展的原因，正如登山者如果总是只根据当地地形设计路线就会陷入困境一样。（由于这种比喻，那种基于目前或者即时信息的启发式系统常称为登山系统。）
6. 系统按顺时针或逆时针顺序旋转方块5、6和8，直到达到目标状态为止。
7. 这里的问题是，我们的启发式方法忽略了让一些未到位的方块与空位相邻的启发值。如果空位被位于正确位置上的方块包围，那么为了让尚未找到正确位置的方块可以移动，就必须先移开某些已位于正确位置的方块。所以认为包围空位的所有方块实际上已经都在正确的位置上的想法是不对的。为了解决这个问题，先观察一个已在正确位置上却堵了别的尚未在正确位置上的方块的移动空位，就得从它的正确位置移开，以后再移回来。因此，如果某方块已位于正确位置上，但却堵在空位与最接近的未在正确位置上的方块之间的路径上，在以后的解题过程中这样的方块都至少还需要移动两次。于是可以这样来修改规划代价：

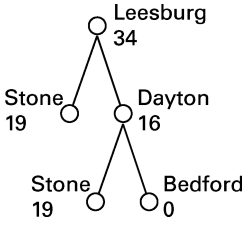
   首先，按原来那样计算规划代价。然而，如果空位完全与未到位的方块隔离，则在空位和一个未到位的方块之间找到一条最短路径，把这个路径上的方块个数乘以2，再把此结果值加到原来的规划代价上。

   对于这个系统，图11-10中的叶子结点有规划代价值6、6和4（从左到右），所以一开始就沿着正确的分支进展。

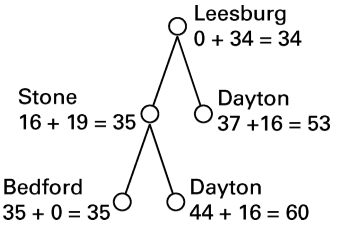
   新的系统十分简单。例如，考虑如下布局。答案是方块5下移，上面两排按顺时针方向移动，直到这些方块移到正确位置，方块5向上移回，最后把方块8移到正确位置。但是，我们新的启发式系统要求从移动方块8开始，因为从这个起始移动得到的状态其规划代价值仅为6，而其他选择代价值为8。



8. 通过最佳适应算法得到的解决方案是从Leesburg到Dayton再到Bedford。这一路线并不是最短的路线。



9. 我们得到的答案是从Leesburg到Stone再到Bedford的路线。这一路线是最短路线。



11.4 节

1. 真实世界知识是人用来理解和推理的关于环境的信息。开发用于表示、存储以及回想这些信息的方法是人工智能研究的一个主要目标。
2. 它使用封闭世界假设。
3. 框架问题是当事件发生时如何正确更新机器的知识存储的问题。由于许多事件有间接结果，所以这个任务非常复杂。
4. 模仿、监督学习以及强化学习。强化学习不涉及直接的人为干预。
5. 传统技术起源于单个计算机系统。进化技术不涉及审判系统的多代衍生，从这些衍生出的系统中，可能会找到“优秀的”系统。

11.5 节

1. 除了模式1、0产生输出1外，其他所有的模式都产生输出0。
2. 给每个输入赋权值1，并且给该单元赋阈值1.5。
3. 文中指出的主要问题是训练过程可能会周期性地震荡，一遍又一遍重复相同的调整。
4. 网络将会走向这样的布局，在该布局中中心神经元兴奋而其他神经元抑制。

11.6 节

1. 反应型方法是当选项出现时等待和作出决定，而不是形成一个完整的行动计划。
2. 这里的要点是考虑机器人学领域是多么的宽广。它涵盖了整个人工智能领域以及其他领域的众多论题。目标是开发真正能够走来走去并能够智能地对所处环境作出反应的自主机器。
3. 内部控制和物理结构。

11.7 节

1. 答案没有对错。
2. 答案没有对错。
3. 答案没有对错。

第 12 章

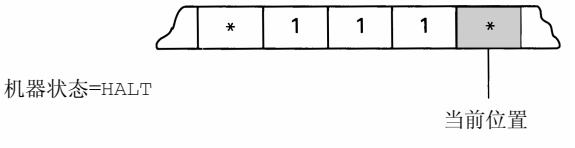
12.1 节

1. 布尔运算AND、OR及XOR。事实上，我们使用的是在第1章介绍这些函数时用过的表。
2. 贷款偿还额、圆的面积或汽车行驶里程的计算。
3. 数学家称这样的函数为超越函数。这样的例子有对数函数和三角函数。这些特殊函数还是能够计算的，只是不能用代数的方法。例如，三角函数能通过实际画出相关的三角形，测量它的边长，然后用代数运算中的除法运算来计算。

4. 一个例子是把角三等分的问题。也就是说，他们不能做到构建这样一个角，其大小是给定角的三分之一。问题在于希腊人的直尺和圆规计算系统是这种具有局限性系统的另一例子。

12.2 节

1. 结果如下图所示：



2.

当前状态	单元内容	写的值	移动方向	进入的新状态
START	*	*	左移	STATE 1
STATE 1	0	0	左移	STATE 2
STATE 1	1	0	左移	STATE 2
STATE 1	*	0	左移	STATE 2
STATE 2	0	*	右移	STATE 3
STATE 2	1	*	右移	STATE 3
STATE 2	*	*	右移	STATE 3
STATE 3	0	0	右移	HALT
STATE 3	1	0	右移	HALT

3.

当前状态	单元内容	写的值	移动方向	进入的新状态
START	*	*	左移	SUBTRACT
SUBTRACT	0	1	左移	BORROW
SUBTRACT	1	0	左移	NO BORROW
BORROW	0	1	左移	BORROW
BORROW	1	0	左移	NO BORROW
BORROW	*	*	右移	ZERO
NO BORROW	0	0	左移	NO BORROW
NO BORROW	1	1	左移	NO BORROW
NO BORROW	*	*	右移	RETURN
ZERO	0	0	右移	ZERO
ZERO	1	0	右移	ZERO
ZERO	*	*	不移动	HALT
RETURN	0	0	右移	RETURN
RETURN	1	1	右移	RETURN
RETURN	*	*	不移动	HALT

4. 这里的问题在于，图灵机的概念被认为是获得了“计算”的含义。也就是说，任何时候只要出现了有计算发生的情况，那么就一定会有图灵机成分和活动的出现。例如，一个人计算所得税，就是在做某种程度的计算。所以，这个人就是计算机器，而磁带则由纸表示，纸上记录着数值。

5. 下表所描述的机器，如果是以偶数作为输入开始，那么就能停止，而如果是以奇数作为输入开始，那么将永不会停止。

当前状态	单元内容	写的值	移动方向	进入的新状态
START	*	*	左移	STATE 1
STATE 1	0	0	右移	HALT
STATE 1	1	1	不移动	STATE 1
STATE 1	*	*	不移动	STATE 1

12.3 节

1.
```
clear   AUX;
incr   AUX;
while X not 0 do;
    clear X;
    clear AUX;
end;
while AUX not 0 do;
    incr X;
    clear AUX;
end;
```

2.
```
while X not 0 do;
    decr X;
end;
```

3.
```
copy X to AUX;
while AUX not 0 do;
    S1
    clear AUX;
end;
copy X to AUX;
invert   AUX;（见问题1）
while AUX not 0 do;
    S2
    clear AUX;
end;
while X not 0 do;
    clear AUX;
    clear X;
end;
```

4. 如果我们假设*X*为地址40的存储单元，且每个程序段从地址00开始，那么我们就有以下的转换表：

	地址	内容
clear X;	00	20
	01	00
	02	30
	03	40

	地址	内容
incr X;	00	11
	01	40
	02	20
	03	01
	04	50
	05	01
	06	30
	07	40

	地址	内容
decr X;	00	20
	01	00
	02	23
	03	00
	04	11
	05	40
	06	22
	07	01
	08	B1
	09	10
	0A	40
	0B	03
	0C	50
	0D	02
	0E	B1
	0F	06
	10	33
	11	40

	地址	内容
while X not	00	20
0 do;	01	00
.	02	11
.	03	40
.	04	B1
end;	05	WZ
	•	•
	•	•
	•	•
	WX	B0
	WY	00

5. 就像在实际的机器中那样，负数可以通过编码系统来处理。例如，把每个字符串的最右边一位用作符号位，而剩下的位用来表示该值的数量级。
6. 该函数表示的是乘以2的乘法运算。

12.4 节

1. 是的。事实上，这个程序不管其变量的初始值为多少，它都会停止。所以，如果它的变量初始值为程序本身的编码表示，那么它一定能终止。
2. 只有当x的初始值以1结束时，这个程序才会终止。因为分号的ASCⅡ码表示是00111011，所以这个程序的编码版本必须是以1结束的。因此，这个程序是自终止的。
3. 这里的问题就在于，其逻辑和我们所讨论的停机问题没有算法解的论述一样。如果房屋油漆工漆自己的房子，那么他就不能漆，反之亦然。

12.5 节

1. 我们只能得出这样的结论，即该问题的复杂性为$\Theta(2^n)$。如果能够证明：解决这个问题的“最优”算法是属于$\Theta(2^n)$的，那么就可以断定，该问题属于$\Theta(2^n)$。
2. 不是。作为一般的规律，属于$\Theta(n^2)$的算法要优于属于$\Theta(2^n)$的算法。但是，对于较小的输入值，指数算法通常优于多项式算法。事实上，当应用仅涉及较小的输入时，比起多项式算法，有时候首选指数算法。
3. 问题就在于小组的数目是以指数的形式增长的，所以从这点来看，要列出所有的可能就是一个比较费时的工作。
4. 排序问题就属于多项式问题类，它可以用多项式算法来解决，如插入排序等。
   从一个给定的委员会中列出所有可能的分组的任务就属于非多项式问题类。
   任何一个多项式问题都是NP问题。旅行商问题就是NP问题的一个例子，但还没有证明是一个多项式问题。
5. 不一定。我们所用的复杂性这个术语指的是执行一个算法所需要的时间，而不是要弄懂这个算法有多难。

12.6 节

1. $211\times313 = 66\,043$。
2. 消息101是5的二进制表示。$5^e = 5^5 = 15\,625$。$15\,625 \pmod{91} = 64$，其二进制表示是1000000。因此，1000000是该消息的加密表示。
3. 消息10是2的二进制表示。$2^d = 2^{29} = 536\,870\,912$。$536\,870\,912 \pmod{91} = 32$，其二进制表示是100000。因此，100000是该消息的解密表示。
4. $n = p\times q = 7\times19 = 133$。为了找到$d$值，我们需要一个正整数值$k$，使得$k(p-1)(q-1)+1 = k(6\times18)+1 = 108k+1$能被$e=5$整除。值$k=1$和$k=2$都不满足，但是$k=3$能够产生$108k+1=325$，这就可以被5整除。所以商65即为$d$的值。