2014年计算机学科专业基础综合试题参考答案

一、单项选择题

1.	C	2.	B	3.	A	4.	D	5.	C	6.	D	7.	D	8.	D
9.	D	10.	B	11.	C	12.	D	13.	C	14.	A	15.	A	16.	D
17.	A	18.	C	19.	C	20.	C	21.	D	22.	B	23.	A	24.	B
25.	D	26.	A	27.	A	28.	C	29.	B	30.	A	31.	C	32.	D
33.	C	34.	B	35.	D	36.	C	37.	B	38.	A	39.	B	40.	D

1．解析：

内层循环条件 $j<=n$ 与外层循环的变量无关，每次循环 j 自增 1，每次内层循环都执行 n 次。外层循环条件为 $k<=n$，增量定义为 k*=2，可知循环次数为 $2^k<=n$，即 $k<=\log_2 n$。所以内层循环的时间复杂度是 $O(n)$，外层循环的时间复杂度是 $O(\log_2 n)$。对于嵌套循环，根据乘法规则可知，该段程序的时间复杂度 $T(n)=T_1(n)T_2(n)=O(n)O(\log_2 n)=O(n\log_2 n)$，选 C。

2．解析：

将中缀表达式转换为后缀表达式的算法思想如下：

从左向右开始扫描中缀表达式；

遇到数字时，加入后缀表达式；

遇到运算符时：

a. 若为‘(’，入栈；

b. 若为‘)’，则依次把栈中的运算符加入后缀表达式中，直到出现‘(’，从栈中删除‘(’；

c. 若为除括号外的其他运算符，当其优先级高于除‘(’以外的栈顶运算符时，直接入栈；否则从栈顶开始，依次弹出比当前处理的运算符优先级高和优先级相等的运算符，直到一个比它优先级低的或者遇到了一个左括号为止。

当扫描的中缀表达式结束时，栈中的所有运算符依次出栈加入后缀表达式。

待处理序列	栈	后缀表达式	当前扫描元素	动作
a/b+(c*d-e*f)/g			a	a 加入后缀表达式
/b+(c*d-e*f)/g		a	/	/入栈
b+(c*d-e*f)/g	/	a	b	b 加入后缀表达式
+(c*d-e*f)/g	/	ab	+	+优先级低于栈顶的/，弹出/
+(c*d-e*f)/g		ab/	+	+入栈
(c*d-e*f)/g	+	ab/	(	(入栈
c*d-e*f)/g	+(	ab/	c	c 加入后缀表达式
*d-e*f)/g	+(	ab/c	*	栈顶为(，*入栈
d-e*f)/g	+(*	ab/c	d	d 加入后缀表达式
-e*f)/g	+(*	ab/cd	-	-优先级低于栈顶的*，弹出*
-e*f)/g	+(	ab/cd*	-	栈顶为(，-入栈

（续表）

待处理序列	栈	后缀表达式	当前扫描元素	动作
e*f)/g	+(-	ab/cd*	e	e 加入后缀表达式
*f)/g	+(-	ab/cd*e	*	*优先级高于栈顶的-，*入栈
f)/g	+(-*	ab/cd*e	f	f加入后缀表达式
)/g	+(-*	ab/cd*ef	)	把栈中(之前的符号加入表达式
/g	+	ab/cd*ef*-	/	/优先级高于栈顶的+，/入栈
g	+/	ab/cd*ef*-	g	g 加入后缀表达式
	+/	ab/cd*ef*-g		扫描完毕，运算符依次退栈加入表达式
		ab/cd*ef*-g/+		完成

由此可知，当扫描到 f 的时候，栈中的元素依次是+(-*，选 B。

在此，再给出中缀表达式转换为前缀或后缀表达式的手工做法，以上面给出的中缀表达式为例：

第一步：按照运算符的优先级对所有的运算单位加括号。

式子变成((a/b)+(((c*d)-(e*f))/g))。

第二步：转换为前缀或后缀表达式。

前缀：把运算符号移动到对应的括号前面，则变成+(/(ab)/(-(*(cd)*(ef))g))。

把括号去掉：+/ab/-*cd*efg 前缀式子出现。

后缀：把运算符号移动到对应的括号后面，则变成((ab)/(((cd)*(ef)*)-g)/)+。

把括号去掉：ab/cd*ef*-g/+后缀式子出现。

当题目要求直接求前缀或后缀表达式时，这种方法会比上一种快捷得多。

3．解析：

end1 指向队头元素，那么可知出队的操作是先从 A[end1]读数，然后 end1 再加 1。end2 指向队尾元素的后一个位置，那么可知入队操作是先存数到 A[end2]，然后 end2 再加 1。若把 A[0]储存第一个元素，当队列初始时，入队操作是先把数据放到 A[0]，然后 end2 自增，即可知 end2 初值为 0；end1 指向的是队头元素，队头元素的在数组 A 中的下标为 0，所以得知 end1 初值也为 0，可知队空条件为 end1==end2。然后考虑队列满时，因为队列最多能容纳 $M-1$ 个元素，假设队列存储在下标为 0 到下标为 $M-2$ 的 $M-1$ 个区域，队头为 A[0]，队尾为 A[$M-2$]，此时队列满，考虑在这种情况下 end1 和 end2 的状态，end1 指向队头元素，可知 end1=0，end2 指向队尾元素的后一个位置，可知 end2 = $M-2+1=M-1$，所以可知队满的条件为 end1==(end2 + 1) mod M，选 A。

注意：考虑这类具体问题时，用一些特殊情况判断往往比直接思考问题能更快地得到答案，并可以画出简单的草图以方便解题。

4．解析：

线索二叉树的线索实际上指向的是相应遍历序列特定结点的前驱结点和后继结点，所以先写出二叉树的中序遍历序列 debxac，中序遍历中在 x 左边和右边的字符，就是它在中序线索化的左、右线索，即 b、a，选 D。

5．解析：

将森林转化为二叉树即相当于用孩子兄弟表示法表示森林。在变化过程中，原森林某结点的第一个孩子结点作为它的左子树，它的兄弟作为它的右子树。那么森林中的叶结点由于没有孩子结点，那么转化为二叉树时，该结点就没有左结点，所以 F 中叶结点的个数就等于 T 中左孩子指针为空的结点个数，选 C。此题还可以通过一些特例来排除 A、B、D 选项。

6．解析：

前缀编码的定义是在一个字符集中，任何一个字符的编码都不是另一个字符编码的前缀。选项 D 中编码 110 是编码 1100 的前缀，违反了前缀编码的规则，所以 D 不是前缀编码。

7．解析：

按照拓扑排序的算法，每次都选择入度为 0 的结点从图中删去，此图中一开始只有结点 3 的入度为 0；删掉结点 3 后，只有结点 1 的入度为 0；删掉结点 1 后，只有结点 4 的入度为 0；删掉结点 4 后，结点 2 和结点 6 的入度都为 0，此时选择删去不同的结点，会得出不同的拓扑序列，分别处理完毕后可知可能的拓扑序列为 3, 1, 4, 2, 6, 5 和 3, 1, 4, 6, 2, 5，选 D。

8．解析：

产生堆积现象，即产生了冲突，它对存储效率、散列函数和装填因子均不会有影响，而平均查找长度会因为堆积现象而增大，选 D。

9．解析：

关键字数量不变，要求结点数量最多，那么即每个结点中含关键字的数量最少。根据 4 阶 B 树的定义，根结点最少含 1 个关键字，非根结点中最少含 $\lceil 4/2 \rceil - 1 = 1$ 个关键字，所以每个结点中，关键字数量最少都为 1 个，即每个结点都有 2 个分支，类似于排序二叉树，而 15 个结点正好可以构造一个 4 层的 4 阶 B 树，使得叶结点全在第四层，符合 B 树定义，因此选 D。

10．解析：

首先，第二个元素为 1，是整个序列中的最小元素，所以可知该希尔排序为从小到大排序。然后考虑增量问题，若增量为 2，第 1+2 个元素 4 明显比第 1 个元素 9 要大，A 排除；若增量为 3，第 i、$i+3$、$i+6$ 个元素都为有序序列（$i=1,2,3$），符合希尔排序的定义；若增量为 4，第 1 个元素 9 比第 1 + 4 个元素 7 要大，C 排除；若增量为 5，第 1 个元素 9 比第 1 + 5 个元素 8 要大，D 排除，选 B。

11．解析：

快排的阶段性排序结果的特点是，第 i 趟完成时，会有 i 个以上的数出现在它最终将要出现的位置，即它左边的数都比它小，它右边的数都比它大。题目问第二趟排序的结果，即要找不存在两个这样的数的选项。A 选项中 2、3、6、7、9 均符合，所以 A 排除；B 选项中，2、9 均符合，所以 B 排除；D 选项中 5、9 均符合，所以 D 选项排除；最后看 C 选项，只有 9 一个数符合，所以 C 不可能是快速排序第二趟的结果。

12．解析：

不妨设原来指令条数为 x，那么原 CPI 就为 $20/x$，经过编译优化后，指令条数减少到原来的 70%，即指令条数为 $0.7x$，而 CPI 增加到原来的 1.2 倍，即 $24/x$，那么现在 P 在 M 上的执行时间就为指令条数×CPI = $0.7x \times 24/x = 24 \times 0.7 = 16.8$s，选 D。

13．解析：

8 位定点补码表示的数据范围为–128～127，若运算结果超出这个范围则会溢出，A 选项 $x + y = 103 - 25 = 78$，符合范围，A 排除；B 选项 $-x + y = -103 - 25 = -128$，符合范围，B 排除；D 选项 $-x - y = -103 + 25 = -78$，符合范围，D 排除；C 选项 $x - y = 103 + 25 = 128$，超过了 127，选 C。

该题也可按照二进制写出两个数进行运算，观察运算的进位信息得到结果，不过这种方法更为麻烦和耗时，在实际考试中并不推荐。

14．解析：

(f1)和(f2)对应的二进制分别是$(110011001001\cdots)_2$和$(101100001100\cdots)_2$，根据 IEEE 754 浮点数标准，可知(f1)的数符为 1，阶码为 10011001，尾数为 1.001，而(f2)的数符为 1，阶码为 01100001，

尾数为 1.1，则可知两数均为负数，符号相同，B、D 排除。(f1)的绝对值为 1.001×2^{26}，(f2)的绝对值为 1.1×2^{-30}，则(f1)的绝对值比(f2)的绝对值大，而符号为负，真值大小相反，即(f1)的真值比(f2)的真值小，即 $x<y$，选 A。

此题还有更为简便的算法，(f1)与(f2)的前 4 位为 1100 与 1011，可以看出两数均为负数，而阶码用移码表示，两数的阶码头三位分别为 100 和 011，可知(f1)的阶码大于(f2)的阶码，又因为是 IEEE754 规格化的数，尾数部分均为 1.xxx，则阶码大的数，真值的绝对值必然大，可知(f1)真值的绝对值大于(f2)真值的绝对值，因为都为负数，则(f1)<(f2)，即 $x<y$。

15．解析：

4M×8 位的芯片数据线应为 8 根，地址线应为 $\log_2 4M = 22$ 根，而 DRAM 采用地址复用技术，地址线是原来的 1/2，且地址信号分行、列两次传送。地址线数为 22/2 = 11 根，所以地址引脚与数据引脚的总数为 11 + 8 = 19 根，选 A。

此题需要注意的是 DRAM 是采用传两次地址的策略，所以地址线为正常的一半，这是很多考生容易忽略的地方。

16．解析：

把指令 Cache 与数据 Cache 分离后，取指和取数分别到不同的 Cache 中寻找，那么指令流水线中取指部分和取数部分就可以很好地避免冲突，即减少了指令流水线的冲突，选 D。

17．解析：

采用 32 位定长指令字，其中操作码为 8 位，两个地址码一共占用 32-8 = 24 位，而 Store 指令的源操作数和目的操作数分别采用寄存器直接寻址和基址寻址，机器中共有 16 个通用寄存器，则寻址一个寄存器需要 $\log_2 16 = 4$ 位，源操作数中的寄存器直接寻址用掉 4 位，而目的操作数采用基址寻址也要指定一个寄存器，同样用掉 4 位，则留给偏移址的位数为 24-4-4 = 16 位，而偏移址用补码表示，16 位补码的表示范围为-32768～+32767，选 A。

18．解析：

计算机共有 32 条指令，各个指令对应的微程序平均为 4 条，则指令对应的微指令为 32×4 = 128 条，而公共微指令还有 2 条，整个系统中微指令的条数一共为 128 + 2 = 130 条，所以需要 $\log_2 130 = 8$ 位才能寻址到 130 条微指令，答案选 C。

19．解析：

数据线有 32 根，也就是一次可以传送 32bit/8 = 4B 的数据，66MHz 意味着有 66M 个时钟周期，而每个时钟周期传送两次数据，可知总线每秒传送的最大数据量为 66M×2×4B = 528MB，所以总线的最大数据传输率为 528MB/s，选 C。

20．解析：

猝发（突发）传输是在一个总线周期中，可以传输多个存储地址连续的数据，即一次传输一个地址和一批地址连续的数据，并行传输是在传输中有多个数据位同时在设备之间进行的传输，串行传输是指数据的二进制代码在一条物理信道上以位为单位按时间顺序逐位传输的方式，同步传输是指传输过程由统一的时钟控制，选 C。

21．解析：

采用统一编址时，CPU 访存和访问 I/O 端口用的是一样的指令，所以访存指令可以访问 I/O 端口，D 选项错误，其他三个选项均为正确陈述，选 D。

22．解析：

每 400ns 发出一次中断请求，而响应和处理时间为 100ns，其中允许的延迟为干扰信息，因为在 50ns 内，无论怎么延迟，每 400ns 还是要花费 100ns 处理中断的，所以该设备的 I/O 时

间占整个 CPU 时间的百分比为 100ns/400ns = 25%，选 B。

23．解析：

采用静态优先级调度时，当系统总是出现优先级高的任务时，优先级低的任务会总是得不到处理机而产生饥饿现象；短任务优先调度不管是抢占式或是非抢占的，当系统总是出现新来的短任务时，长任务会总是得不到处理机，产生饥饿现象，因此 B、C、D 都错误，选 A。

24．解析：

三个并发进程分别需要 3、4、5 台设备，当系统只有(3−1) + (4−1) + (5−1) = 9 台设备时，第一个进程分配 2 台，第二个进程分配 3 台，第三个进程分配 4 台。这种情况下，三个进程均无法继续执行下去，发生死锁。当系统中再增加 1 台设备，也就是总共 10 台设备时，这最后 1 台设备分配给任意一个进程都可以顺利执行完成，因此保证系统不发生死锁的最小设备数为 10。

25．解析：

trap 指令、跳转指令和压栈指令均可以在用户态执行，其中 trap 指令负责由用户态转换成为内核态，而关中断指令为特权指令，必须在核心态才能执行，选 D。

26．解析：

进程申请读磁盘操作的时候，因为要等待 I/O 操作完成，会把自身阻塞，此时进程就变为了阻塞状态，当 I/O 操作完成后，进程得到了想要的资源，就会从阻塞态转换到就绪态（这是操作系统的行为）。而降低进程优先级、分配用户内存空间和增加进程的时间片大小都不一定会发生，选 A。

27．解析：

簇的总数为 10GB/4KB = 2.5M，用一位标识一簇是否被分配，则整个磁盘共需要 2.5M 位，即需要 2.5M/8 = 320KB，因此共需要 320KB/4KB = 80 个簇，选 A。

28．解析：

虚实地址转换是指逻辑地址和物理地址的转换。增大快表容量能把更多的表项装入快表中，会加快虚实地址转换的平均速率；让页表常驻内存可以省去一些不在内存中的页表从磁盘上调入的过程，也能加快虚实地址转换；增大交换区对虚实地址转换速度无影响，因此 I、II 正确，选 C。

29．解析：

一个文件被用户进程首次打开即被执行了 Open 操作，会把文件的 FCB 调入内存，而不会把文件内容读到内存中，只有进程希望获取文件内容的时候才会读入文件内容；C、D 明显错误，选 B。

30．解析：

只有 FIFO 算法才会导致 Belady 异常，选 A。

31．解析：

管道实际上是一种固定大小的缓冲区，管道对于管道两端的进程而言，就是一个文件，但它不是普通的文件，它不属于某种文件系统，而是自立门户，单独构成一种文件系统，并且只存在于内存中。它类似于通信中半双工信道的进程通信机制，一个管道可以实现双向的数据传输，而同一个时刻只能最多有一个方向的传输，不能两个方向同时进行。管道的容量大小通常为内存上的一页，它的大小并不是受磁盘容量大小的限制。当管道满时，进程在写管道会被阻塞，而当管道空时，进程在读管道会被阻塞，因此选 C。

32．解析：

多级页表不仅不会加快地址的变换速度，而且会因为增加更多的查表过程，使地址变换速度减慢；也不会减少缺页中断的次数，反而如果访问过程中多级的页表都不在内存中，会大大增加缺页的次数，也并不会减少页表项所占的字节数，而多级页表能够减少页表所占的连续内存空间，即当页表太大时，将页表再分级，可以把每张页表控制在一页之内，减少页表所占的连续内存空间，因此选 D。

33．解析：

直接为会话层提供服务的是会话层的下一层，即传输层，选 C。

34．解析：

主机 00-e1-d5-00-23-a1 向 00-e1-d5-00-23-c1 发送数据帧时，交换机转发表中没有 00-e1-d5-00-23-c1 这项，所以向除 1 接口外的所有接口广播这帧，即 2、3 端口会转发这帧，同时因为转发表中并没有 00-e1-d5-00-23-a1 这项，所以转发表会把（目的地址 00-e1-d5-00-23-a1，端口 1）这项加入转发表。而当 00-e1-d5-00-23-c1 向 00-e1-d5-00-23-a1 发送确认帧时，由于转发表已经有 00-e1-d5-00-23-a1 这项，所以交换机只向 1 端口转发，选 B。

35．解析：

由香农定理可知，信噪比和频率带宽都可以限制信道的极限传输速率，所以信噪比和频率带宽对信道的数据传输速率是有影响的，A、B 错误；信道的传输速率实际上就是信号的发送速率，而调制速度也会直接限制数据的传输速率，C 错误；信号的传播速度是信号在信道上传播的速度，与信道的发送速率无关，选 D。

36．解析：

考虑制约甲的数据传输速率的因素，首先，信道带宽能直接制约数据的传输速率，传输速率一定是小于等于信道带宽的；其次，主机甲、乙之间采用后退 N 帧协议，那么因为甲、乙主机之间采用后退 N 帧协议传输数据，要考虑发送一个数据到接收到它的确认之前，最多能发送多少数据，甲的最大传输速率受这两个条件的约束，所以甲的最大传输速率是这两个值中小的那一个。甲的发送窗口的尺寸为 1000，即收到第一个数据的确认之前，最多能发送 1000 个数据帧，也就是发送 1000×1000B = 1MB 的内容，而从发送第一个帧到接收到它的确认的时间是一个往返时延，也就是 50 + 50 = 100ms = 0.1s，即在 100ms 中，最多能传输 1MB 的数据，因此，此时的最大传输速率为 1MB/0.1s = 10MB/s = 80Mbps。信道带宽为 100Mbps，所以答案为 min{80Mbps,100Mbps} = 80Mbps，选 C。

37．解析：

把收到的序列分成每 4 个数字一组，即为(2, 0, 2, 0)、(0, -2, 0, -2)、(0, 2, 0, 2)，因为题目求的是 A 发送的数据，因此把这三组数据与 A 站的码片序列(1, 1, 1, 1)做内积运算，结果分别是(2, 0, 2, 0) • (1, 1, 1, 1)/4 = 1、(0, −2, 0, −2) • (1, 1, 1, 1)/4 = −1、(0, 2, 0, 2) • (1, 1, 1, 1)/4 = 1，所以 C 接收到的 A 发送的数据是 101，选 B。

38．解析：

当 t 时刻发生超时时，把 ssthresh 设为 8 的一半，即为 4，且拥塞窗口设为 1KB。然后经历 10 个 RTT 后，拥塞窗口的大小依次为 2, 4, 5, 6, 7, 8, 9, 10, 11, 12，而发送窗口取当时的拥塞窗口和接收窗口的最小值，而接收窗口始终为 10KB，所以此时的发送窗口为 10KB，选 A。

实际上该题接收窗口一直为 10KB，可知不管何时，发送窗口一定小于等于 10KB，选项中只有 A 选项满足条件，故选 A。

39．解析：

UDP 提供的是无连接的服务，I 正确；同时 UDP 也提供复用/分用服务，II 正确；UDP 虽然有差错校验机制，但是 UDP 的差错校验只是检查数据在传输的过程中有没有出错，出错的数据直接丢弃，并没有重传等机制，不能保证可靠传输，使用 UDP 协议时，可靠传输必须由应用层实现，III 错误。答案选 B。

40．解析：

当接入网络时可能会用到 PPP 协议，A 可能用到；当计算机不知道某主机的 MAC 地址时，用 IP 地址查询相应的 MAC 地址时会用到 ARP 协议，B 可能用到；当访问 Web 网站时，若 DNS 缓冲没有存储相应域名的 IP 地址，用域名查询相应的 IP 地址时要使用 DNS 协议，而 DNS 是基于 UDP 协议的，所以 C 可能用到；SMTP 只有使用邮件客户端发送邮件，或是邮件服务器向别的邮件服务器发送邮件时才会用到，单纯的访问 Web 网页不可能用到，选 D。

二、综合应用题

41．解答：

1）算法的基本设计思想：

① 基于先序递归遍历的算法思想是用一个 static 变量记录 wpl，把每个结点的深度作为递归函数的一个参数传递，算法步骤如下：

若该结点是叶子结点，则变量 wpl 加上该结点的深度与权值之积；

若该结点非叶子结点，则若左子树不为空，对左子树调用递归算法，若右子树不为空，对右子树调用递归算法，深度参数均为本结点的深度参数加 1；

最后返回计算出的 wpl 即可。

② 基于层次遍历的算法思想是使用队列进行层次遍历，并记录当前的层数，

当遍历到叶子结点时，累计 wpl；

当遍历到非叶子结点时，把该结点的子树加入队列；

当某结点为该层的最后一个结点时，层数自增 1；

队列空时遍历结束，返回 wpl。

2）二叉树结点的数据类型定义如下：

```
typedef struct BiTNode{
    int weight;
    struct BiTNode *lchild,*rchild;
}BiTNode,*BiTree;
```

3）算法代码如下：

① 基于先序遍历的算法：

```
int WPL(BiTree root){
    return wpl_PreOrder(root, 0);
}
int wpl_PreOrder(BiTree root, int deep){
    static int wpl = 0;                          //定义一个 static 变量存储 wpl
    if(root->lchild == NULL && root->rchild == NULL)
                                                 //若为叶子结点，累积 wpl
        wpl += deep*root->weight;
    if(root->lchild != NULL)                     //若左子树不空，对左子树递归遍历
        wpl_PreOrder(root->lchild, deep+1);
    if(root->rchild != NULL)                     //若右子树不空，对右子树递归遍历
        wpl_PreOrder(root->rchild, deep+1);
```

```
    return wpl;
}
```

② 基于层次遍历的算法：

```
#define MaxSize 100                    //设置队列的最大容量
int wpl_LevelOrder(BiTree root){
    BiTree q[MaxSize];                 //声明队列，end1 为头指针，end2 为尾指针
    int end1, end2;                    //队列最多容纳 MaxSize-1 个元素
    end1 = end2 = 0;                   //头指针指向队头元素，尾指针指向队尾的后一个元素
    int wpl = 0, deep = 0;             //初始化 wpl 和深度
    BiTree lastNode;                   //lastNode 用来记录当前层的最后一个结点
    BiTree newlastNode;                //newlastNode 用来记录下一层的最后一个结点
    lastNode = root;                   //lastNode 初始化为根结点
    newlastNode = NULL;                //newlastNode 初始化为空
    q[end2++] = root;                  //根结点入队
    while(end1 != end2){               //层次遍历，若队列不空则循环
        BiTree t = q[end1++];          //拿出队列中的头一个元素
        if(t->lchild == NULL & t->lchild == NULL){
            wpl += deep*t->weight;
        }                              //若为叶子结点，统计 wpl
        if(t->lchild != NULL){         //若非叶子结点把左结点入队
            q[end2++] = t->lchild;
            newlastNode = t->lchild;
        }                              //并设下一层的最后一个结点为该结点的左结点
        if(t->rchild != NULL){         //处理叶结点
            q[end2++] = t->rchild;
            newlastNode = t->rchild;
        }
        if(t == lastNode){             //若该结点为本层最后一个结点，更新 lastNode
            lastNode = newlastNode;
            deep += 1;                 //层数加 1
        }
    } //while
    return wpl;                        //返回 wpl
}
```

【评分说明】

① 若考生给出能够满足题目要求的其他算法且正确，可同样给分。

② 考生答案无论使用 C 或者 C++语言，只要答案正确同样给分。

③ 若对算法的基本设计思想和主要数据结构描述不十分准确，但在算法实现中能够清晰反映出算法思想且正确，参照①的标准给分。

④ 若考生给出的二叉树结点的数据类型定义和算法实现中，使用的是除整型之外的其他数值，可视同使用整型类型。

⑤ 若考生给出的答案中算法主要设计思想或算法中部分正确，可酌情给分。

注意：上述两个算法一个为递归的先序遍历，一个为非递归的层次遍历，读者应当选取自己最擅长的书写方式。直观看去，先序遍历代码行数少，不用运用其他工具，书写也更容易，希望读者能掌握。

在先序遍历的算法中，static 是一个静态变量，只在首次调用函数时声明 wpl 并赋值为 0，以后的递归调用并不会使得 wpl 为 0，具体用法请参考相关资料中的关键字 static 说明，也可以在函数之外预先设置一个全局变量，并初始化。不过考虑到历年真题算法答案通常都直接仅由

一个函数构成，所以参考答案使用 st at ic 若对 st at ic不熟悉的同学可以使用以下形式的递归：

```
int wpl_PreOrder(BiTree root,int deep){
    int lwpl,rwpl;                    //用于存储左子树和右子树的产生的 wpl
    lwpl=rwpl=0;
    if(root->lchild==NULL&&root->lchild==NULL)
                                      //若为叶结点，计算当前叶结点的 wpl
        return deep*root->weight;
    if(root->lchild!=NULL)            //若左子树不空，对左子树递归遍历
        lwpl = wpl_PreOrder(root->lchild, deep+1);
    if(root->rchild!=NULL)            //若右子树不空，对右子树递归遍历
        rwpl=wpl_PreOrder(root->rchild,deep+1);
    return lwpl+rwpl;
}
```

C/C++语言基础好的同学可以使用以下更简便的形式：

```
int wpl_PreOrder(BiTree root,int deep){
    if(root->lchild==NULL&&root->lchild==NULL)   //若为叶子结点，累积 wpl
        return deep*root->weight;
    return (root->lchild!=NULL ? wpl_PreOrder(root->lchild, deep+1):0)
    + (root->rchild!=NULL ? wpl_PreOrder(root->rchild, deep+1):0);
}
```

这个形式只是上面方法的简化而已，本质是一样的，而这个形式的代码更短，在时间有限的情况下更具优势，相比写层次遍历能节约很多时间，所以读者应当在保证代码正确的情况下，尽量写一些较短的算法，为解答其他题目赢得更多的时间。但是，对于基础不扎实的考生，还是建议使用把握更大的方法，否则可能会得不偿失。例如在上面的代码中，考生容易忘记三元式(x?y:z)两端的括号，若不加括号，则答案就会是错误的。

在层次遍历的算法中，读者要理解 lastNode 和 newlastNode 的区别，last Node 指的是当前遍历层的最后一个结点，而 newlast Node 指的是下一层的最后一个结点，是动态变化的，直到遍历到本层的最后一个结点，才能确认下层真正的最后一个结点是哪个结点，而函数中入队操作并没有判断队满，若考试时用到，读者最好加上队满条件，这里队列的队满条件为 end1==(end2+1) mod *M*。同时，考生也可以尝试使用记录每层的第一个结点来进行层次遍历的算法，这里不再给出代码，请考生自行练习。

42．解答：

很多考生乍看之下以为是网络的题目，其实该题本身并没有涉及太多的网络知识点，只是应用了网络的模型，实际上考查的还是数据结构的内容。

（1）图（1 分）

题中给出的是一个简单的网络拓扑图，可以抽象为无向图。

【评分说明】

只要考生的答案中给出与图含义相似的描述，例如，“网状结构”“非线性结构”等，同样给分。

（2）链式存储结构的如下图所示。

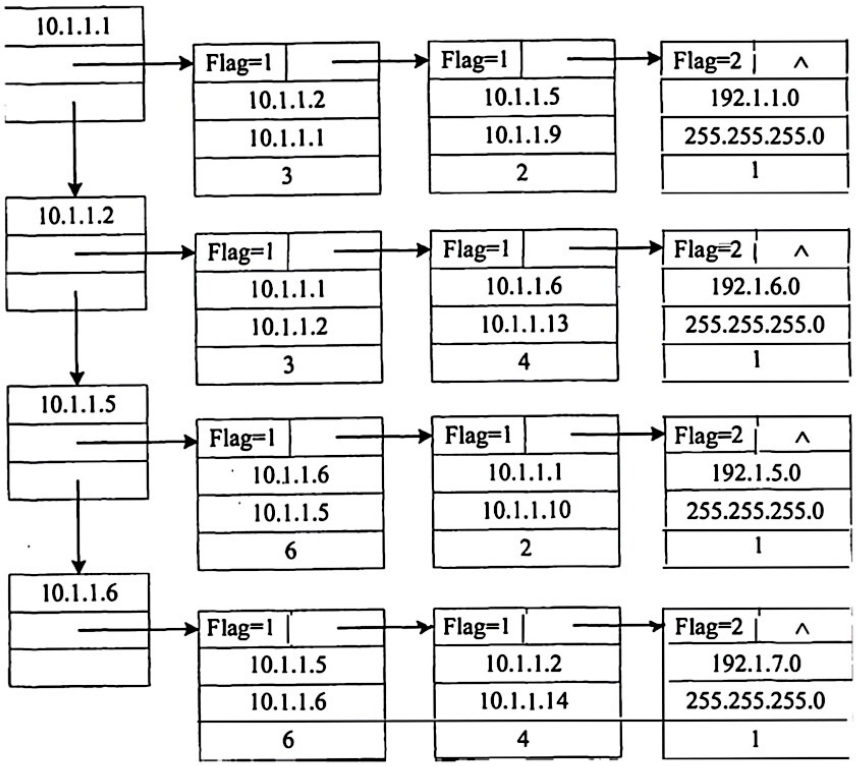
弧结点的两种基本形态



其数据类型定义如下：（3 分）

```
typedef struct{
    unsigned int ID, IP;
}LinkNode;                    //Link 的结构
typedef struct{
    unsigned int Prefix, Mask;
}NetNode;                     //Net 的结构
typedef struct Node{
    int Flag;          //Flag=1 为 Link;Flag=2 为 Net
    union{
        LinkNode Lnode;
        NetNode Nnode
    }LinkORNet;
    unsigned int Metric;
    struct Node *next;
}ArcNode;                     //弧结点
typedef struct HNode{
    unsigned int RouterID;
    ArcNode *LN_link;
    struct HNode *next;
}HNODE;                       //表头结点
```

对应上述链式存储结构的示意图如下。（2 分）



【评分说明】

① 若考生给出的答案是将链表中的表头结点保存在一个一维数组中（即采用邻接表形式），同样给分。

② 若考生给出的答案中，弧结点没有使用 union 定义，而是采用两种不同的结构分别表示 Link 和 Net，同时在表头结点中定义了两个指针，分别指向由这两种类型的结点构成的两个链表，同样给分。

③ 考生所给答案的弧结点中，可以在单独定义的域中保存各直连网络 IP 地址的前缀长度，也可以与网络地址保存在同一个域中。

④ 数据类型定义中，只要采用了可行的链式存储结构，并保存了题目中所给的 LSI 信息，如将网络抽象为一类结点，写出含 8 个表头结点的链式存储结构，均可参照①～③的标准给分。

⑤ 若考生给出的答案中，图示部分与其数据类型定义部分一致，图示只要能够体现链式存储结构和网络连接关系（可以不给出结点内细节信息），即可给分。

⑥ 若解答不完全正确，酌情给分。

（3）计算结果如下表所示。（4 分）

	目的网络	路径	代价（费用）
步骤 1	192.1.1.0/24	直接到达	1
步骤 2	192.1.5.0/24	R1→R3→192.1.5.0/24	3
步骤 3	192.1.6.0/24	R1→R2→192.1.6.0/24	4
步骤 4	192.1.7.0/24	R1→R2→R4→192.1.7.0/24	8

【评分说明】

① 若考生给出的各条最短路径的结果部分正确，可酌情给分。

② 若考生给出的从 R1 到达子网 192.1.x.x 的最短路径及代价正确，但不完全符合代价不减的次序，可酌情给分。

43．解答：

（1）因为题目要求路由表中的路由项尽可能少，所以这里可以把子网 192.1.6.0/24 和 192.1.7.0/24 聚合为子网 192.1.6.0/23。其他网络照常，可得到路由表如下：（6 分）

目的网络	下一条	接口
192.1.1.0/24	-	E0
192.1.6.0/23	10.1.1.2	L0
192.1.5.0/24	10.1.1.10	L1

【评分说明】

① 每正确解答一个路由项，给 2 分，共 6 分。

② 路由项解答不完全正确，或路由项多于 3 条，可酌情给分。

（2）通过查路由表可知：R1 通过 L0 接口转发该 IP 分组。（1 分）因为该分组要经过 3 个路由器（R1, R2, R4），所以主机 192.1.7.211 收到的 IP 分组的 TTL 是 64−3 = 61。（1 分）

（3）R1 的 LSI 需要增加一条特殊的直连网络，网络前缀 Prefix 为“0.0.0.0/0”，Metric 为 10。（1 分）

【评分说明】

考生只要回答：增加前缀 Prefix 为“0.0.0.0/0”，Metric 为 10，同样给分。

44．解答：

该题涉及指令系统、存储管理以及 CPU 三个部分内容，考生应注意各章节内容之间的联系，才能更好地把握当前考试的趋势。

1）已知计算机 M 采用 32 位定长指令字，即一条指令占 4B，观察表中各指令的地址可知，每条指令的地址差为 4 个地址单位，即 4 个地址单位代表 4B，一个地址单位就代表了 1B，所以该计算机是按字节编址的。（2 分）

2）在二进制中某数左移两位相当于乘以 4，由该条件可知，数组间的数据间隔为 4 个地址

单位，而计算机按字节编址，所以数组 A 中每个元素占 4B。（2 分）

3）由表可知，bne 指令的机器代码为 1446FFFAH，根据题目给出的指令格式，后 2B 的内容为 OFFSET 字段，所以该指令的 OFFSET 字段为 FFFAH，用补码表示，值为–6（1 分）。当系统执行到 bne 指令时，PC 自动加 4，PC 的内容就为 08048118H，而跳转的目标是 08048100H，两者相差了 18H，即 24 个单位的地址间隔，所以偏移址的一位即是真实跳转地址的–24/–6 = 4 位（1 分）。可知 bne 指令的转移目标地址计算公式为(PC) + 4 + OFFSET×4（1 分）。

4）由于数据相关而发生阻塞的指令为第 2、3、4、6 条，因为第 2、3、4、6 条指令都与各自前一条指令发生数据相关。（3 分）

第 6 条指令会发生控制冒险。（1 分）

当前循环的第五条指令与下次循环的第一条指令虽然有数据相关，但由于第 6 条指令后有 3 个时钟周期的阻塞，因而消除了该数据相关。（1 分）

【评分说明】

对于第 1 问，若考生回答：因为指令 1 和 2、2 和 3、3 和 4、5 和 6 发生数据相关，因而发生阻塞的指令为第 2、3、4、6 条，同样给 3 分。答对 3 个以上给 3 分，部分正确酌情给分。

45．解答：

1）R2 里装的是 i 的值，循环条件是 $i < N$（1000），即当 i 自增到不满足这个条件时跳出循环，程序结束，所以此时 i 的值为 1000。（1 分）

2）Cache 共有 16 块，每块 32 字节，所以 Cache 数据区的容量为 16×32B = 512B。（1 分）

P 共有 6 条指令，占 24 字节，小于主存块大小（32B），其起始地址为 0804 8100H，对应一块的开始位置，由此可知所有指令都在一个主存块内。读取第一条指令时会发生 Cache 缺失，故将 P 所在的主存块调入 Cache 某一块，以后每次读取指令时，都能在指令 Cache 中命中。因此在 1000 次循环中，只会发生 1 次指令访问缺失，所以指令 Cache 的命中率为(1000×6−1)/(1000×6) = 99.98%。（2 分）

【评分说明】若考生给出正确的命中率，而未说明原因和过程，给 1 分。若命中率计算错误，但解题思路正确，可酌情给分。

3）指令 4 为加法指令，即对应 sum+=A[i]，当数组 A 中元素的值过大时，则会导致这条加法指令发生溢出异常；而指令 2、5 虽然都是加法指令，但它们分别为数组地址的计算指令和存储变量 i 的寄存器进行自增的指令，而 i 最大到达 1000，所以它们都不会产生溢出异常。（2 分）

只有访存指令可能产生缺页异常，即指令 3 可能产生缺页异常。（1 分）

因为数组 A 在磁盘的一页上，而一开始数组并不在主存中，第一次访问数组时会导致访盘，把 A 调入内存，而以后数组 A 的元素都在内存中，则不会导致访盘，所以该程序一共访盘一次。（2 分）

每访问一次内存数据就会查 TLB 一次，共访问数组 1000 次，所以此时又访问 TLB1000 次，还要考虑到第一次访问数组 A，即访问 A[0]时，会多访问一次 TLB（第一次访问 A[0]会先查一次 TLB，然后产生缺页，处理完缺页中断后，会重新访问 A[0]，此时又查 TLB），所以访问 TLB 的次数一共是 1001 次。（2 分）

【评分说明】

① 对于第 1 问，若答案中除指令 4 外还包含其他运算类指令（即指令 1、2、5），则给 1 分，其他情况，则给 0 分。

② 对于第 2 问，只要回答“load 指令”，即可得分。

③ 对于第 3 问，若答案中给出的读 TLB 的次数为 1002，同样给分。若直接给出正确的

TLB 及磁盘的访问次数，而未说明原因，给 3 分。若给出的 TLB 及磁盘访问次数不正确，但解题思路正确，可酌情给分。

46．解答：

1）系统采用顺序分配方式时，插入记录需要移动其他的记录块，整个文件共有 200 条记录，要插入新记录作为第 30 条，而存储区前后均有足够的磁盘空间，且要求最少的访问存储块数，则要把文件前 29 条记录前移，若算访盘次数移动一条记录读出和存回磁盘各是一次访盘，29 条记录共访盘 58 次，存回第 30 条记录访盘 1 次，共访盘 59 次。（1 分）

F 的文件控制区的起始块号和文件长度的内容会因此改变。（1 分）

2）文件系统采用链接分配方式时，插入记录并不用移动其他记录，只需找到相应的记录，修改指针即可。插入的记录为其第 30 条记录，那么需要找到文件系统的第 29 块，一共需要访盘 29 次，然后把第 29 块的下块地址部分赋给新块，把新块存回内存会访盘 1 次，然后修改内存中第 29 块的下块地址字段，再存回磁盘（1 分），一共访盘 31 次。（1 分）

4 字节共 32 位，可以寻址 $2^{32}=4G$ 块存储块，每块的大小为 1KB，即 1024B，其中下块地址部分占 4B，数据部分占 1020B，那么该系统的文件最大长度是 4G×1020B＝4080GB。（2 分）

【评分说明】

① 第 1）小题的第 2 小问，若答案中不包含文件的起始地址和文件大小，则不给分。

② 若按 $1024\times2^{32}B=4096GB$ 计算最大长度，给 1 分。

47．解答：

这是典型的生产者和消费者问题，只对典型问题加了一个条件，只需在标准模型上新加一个信号量，即可完成指定要求。

设置四个变量 mutex1、mutex2、empty 和 full，mutex1 用于一个控制一个消费者进程一个周期（10 次）内对于缓冲区的控制，初值为 1；mutex2 用于进程单次互斥的访问缓冲区，初值为 1；empty 代表缓冲区的空位数，初值为 0；full 代表缓冲区的产品数，初值为 1000，具体进程的描述如下：

```
semaphore mutex1=1;
semaphore mutex2=1;
semaphore empty=n;
semaphore full=0;
producer(){
 while(1){
     生产一个产品;
     P(empty);                  //判断缓冲区是否有空位
     P(mutex2);                 //互斥访问缓冲区
     把产品放入缓冲区;
     V(mutex2);                 //互斥访问缓冲区
     V(full);                   //产品的数量加 1
 }
}

consumer(){
 while(1){
     P(mutex1)                          //连续取 10 次
         for(int i = 0; i < 10; ++i){
             P(full);                   //判断缓冲区是否有产品
             P(mutex2);                 //互斥访问缓冲区
```

```
                从缓冲区取出一件产品;
                V(mutex2);              //互斥访问缓冲区
                V(empty);               //腾出一个空位
                消费这件产品;
            }
        V(mutex1)
    }
}
```

【评分说明】

① 信号量的初值和含义都正确给 2 分。

② 生产者之间的互斥操作正确给 1 分；生产者与消费者之间的同步操作正确给 2 分；消费者之间互斥操作正确给 1 分。

③ 控制消费者连续取产品数量正确给 2 分。

④ 仅给出经典生产者–消费者问题的信号量定义和伪代码描述最多给 3 分。

⑤ 若考生将题意理解成缓冲区至少有 10 件产品，消费者才能开始取，其他均正确，得 6 分。

⑥ 部分完全正确，酌情给分。