2019年计算机学科专业基础综合试题参考答案

一、单项选择题

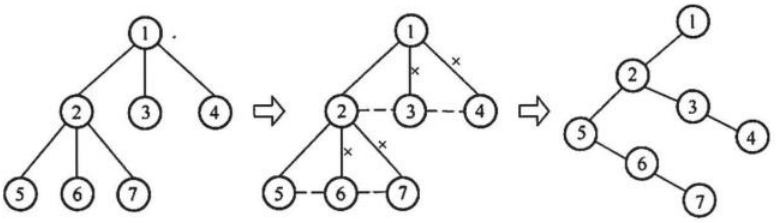
1.	B	2.	B	3.	C	4.	A	5.	C	6.	A	7.	D	8.	C
9.	B	10.	D	11.	B	12.	C	13.	A	14.	D	15.	D	16.	D
17.	B	18.	C	19.	B	20.	C	21.	A	22.	D	23.	B	24.	C
25.	C	26.	B	27.	C	28.	B	29.	C	30.	B	31.	A	32.	C
33.	C	34.	A	35.	B	36.	B	37.	B	38.	C	39.	D	40.	B

1．解析：

假设第 k 次循环终止，则第 k 次执行时，$(x+1)^2>n$，x 的初始值为 0，第 k 次判断时，$x=k-1$，即 $k^2>n$，$k>\sqrt{n}$，因此该程序段的时间复杂度为 $O(\sqrt{n})$。因此选 B。

2．解析：

后根遍历树可分为两步：① 从左到右访问双亲结点的每个孩子（转化为二叉树后就是先访问根结点再访问右子树）；② 访问完所有孩子后再访问它们的双亲结点（转化为二叉树后就是先访问左子树再访问根结点），因此树 T 的后根遍历序列与其相应二叉树 BT 的中序遍历序列相同。对于此类题，采用特殊值法求解通常会更便捷，左下图树 T 转换为二叉树 BT 的过程如下图所示，树 T 的后序遍历序列显然和其相应二叉树 BT 的中序遍历序列相同，均为 5, 6, 7, 2, 3, 4, 1。因此选 B。



3．解析：

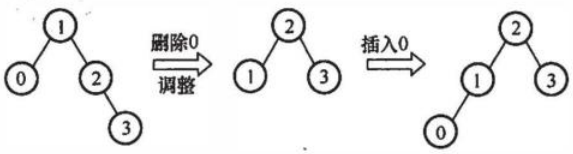
n 个符号构造成哈夫曼树的过程中，共新建了 $n-1$ 个结点（双分支结点），因此哈夫曼树的结点总数为 $2n-1=115$，n 的值为 58，答案选 C。

4．解析：

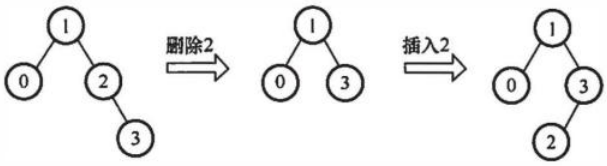
在非空平衡二叉树中插入结点，在失去平衡调整前，一定插入在叶结点的位置。

若删除的是 T_1 的叶结点，则删除后平衡二叉树不会失去平衡，即不会发生调整，再插入此结点得到的二叉平衡树 T_1 与 T_3 相同；若删除后平衡二叉树失去平衡而发生调整，再插入结点得到的二叉平衡树 T_3 与 T_1 可能不同。I 正确。例如，如下图所示，删除结点 0，平衡二叉树失衡调整，再插入结点 0 后，平衡二叉树和以前不同。

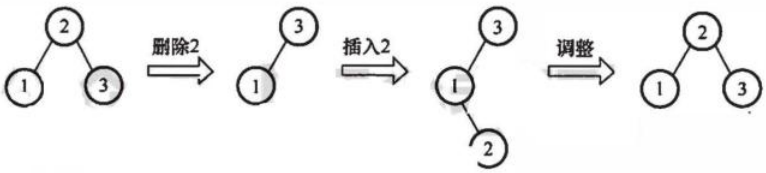
对于比较绝对的说法 II 和 III，通常只需举出反例即可。



若删除的是 T_1 的非叶结点，且删除和插入操作均没有导致平衡二叉树的调整（这时可以首先想到删除的结点只有一个孩子的情况），则该结点从非叶结点变成了叶结点，T_1 与 T_3 显然不同。例如，如下图所示，删除结点 2，用右孩子结点 3 填补，再插入结点 2，平衡二叉树和以前不同。



若删除的是 T_1 的非叶结点，且删除和插入操作后导致了平衡二叉树的调整，则该结点有可能通过旋转后继续变成非叶结点，T_1 与 T_3 相同。例如，如下图所示，删除结点 2，用右孩子结点 3 填补，再插入结点 2，平衡二叉树失衡调整，调整后的平衡二叉树和以前相同。



5．解析：

活动 d 的最早开始时间等于该活动弧的起点所表示的事件的最早发生时间，活动 d 的最早开始时间等于事件 2 的最早发生时间 $\max\{a,b+c\}=\max\{3,12\}=12$。活动 d 的最迟开始时间等于该活动弧的终点所表示的事件的最迟发生时间与该活动所需时间之差，先算出图中关键路径长度为 27（对于不复杂的选择题，找出所有路径计算长度），那么事件 4 的最迟发生时间为 $\min\{27-g\}=\min\{27-6\}=21$，活动 d 的最迟开始时间为 $21-d=21-7=14$。

常规方法：按照关键路径算法得到下表。

	v_1	v_2	v_3	v_4	v_5	v_6
$v_{e(i)}$	0	12	8	19	18	27
$v_{l(i)}$	0	12	8	21	18	27

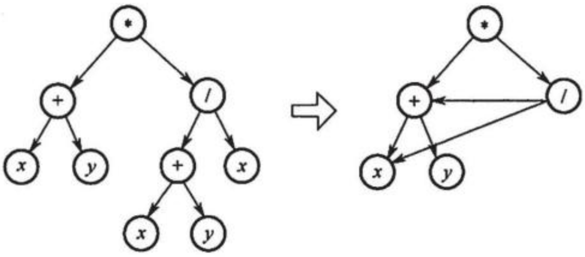
	a	b	c	d	e	f	g	h
$e(i)$	0	8	0	12	12	8	19	18
$l(i)$	9	8	0	14	12	8	21	18
$l(i)-e(i)$	9	0	0	2	0	0	3	0

从表中可知，活动 d 的最早开始时间和最迟开始时间分别为 12 和 14，故选 C。

6．解析：

先将该表达式转换成有向二叉树，注意到该二叉树中有些顶点是重复的，为了节省存储空间，可以去除重复的顶点（使顶点个数达到最少），将有向二叉树去重转换成有向无环图，如下

图所示。答案选 A。



7. 解析：

当数据规模较小时可选择复杂度为 $O(n^2)$ 的简单排序方法，当数据规模较大时应选择复杂度为 $O(n\log_2 n)$ 的排序方法，当数据规模大到内存无法放下时需选择外部排序方法，I 正确。数据的存储方式主要分为顺序存储和链式存储，有些排序方法（如堆排序）只能用于顺序存储方式，II 正确。若对数据稳定性有要求，则不能选择不稳定的排序方法，III 显然正确。当数据初始基本有序时，直接插入排序的效率最高，冒泡排序和直接插入排序的时间复杂度都是 $O(n)$，而归并排序的时间复杂度依旧是 $O(n\log_2 n)$，IV 正确。所以选 D。

8. 解析：

采用线性探查法计算每个关键字的存放情况，如下表所示。

散列地址	0	1	2	3	4	5	6	7	8	9	10
关键字	98	22	30	87	11	40	6	20			

由于 $H(\text{key}) = 0\sim6$，查找失败时可能对应的地址有 7 个，对于计算出地址为 0 的关键字 key0，只有比较完 0～8 号地址后才能确定该关键字不在表中，比较次数为 9；对于计算出地址为 1 的关键字 key1，只有比较完 1～8 号地址后才能确定该关键字不在表中，比较次数为 8；以此类推。需要特别注意的是，散列函数不可能计算出地址 7，因此有

$$\text{ASL}_{\text{失败}} = (9+8+7+6+5+4+3)/7 = 6$$

因此选 C。

9. 解析：

假设位序都是从 0 开始的，按照 next 数组生成算法，对于 S 有

编号	0	1	2	3	4	5
S	a	b	a	a	b	c
next	-1	0	0	1	1	2

根据 KMP 算法，第一趟连续比较 6 次，在模式串的 5 号位和主串的 5 号位匹配失败，模式串的下一个比较位置为 next[5]，即下一次比较从模式串的 2 号位和主串的 5 号位开始，然后直到模式串 5 号位和主串 8 号位匹配，第二趟比较 4 次，模式串匹配成功。单个字符的比较次数为 10 次，所以选 B。

10. 解析：

要理解清楚排序过程中一"趟"的含义，题干也进行了解释。一个初始无序序列，所有元素都没有确定最终位置，对所有元素做一次（称为趟）快速排序后一个元素确定最终位置，且将原序列划分成了前后两块，此时前后两块子表是无序的。按"趟"的解释——对尚未确定最终位置的所有元素都处理一遍才是一趟，所以此时要对前后两块子表各做一次快速排序才是一

“趟”快速排序，如果只对一块子表进行了排序，而未处理另一块子表，就不能算是完整的一趟。

选项 A，第一趟匹配 72，只余一块无序序列，第二趟匹配 28，选项 A 可能。选项 B，第一趟匹配 2，第二趟匹配 72，选项 B 可能。选项 C，第一趟匹配 2，第二趟匹配 28 或 32，选项 C 可能。选项 D，无论先匹配 12 还是先匹配 32，都会将序列分成两块，那么第二趟必须有两个元素匹配，所以选项 D 不可能。故选 D。

11．解析：

在 12 路归并树中只存在度为 0 和度为 12 的结点，设度为 0 的结点数、度为 12 的结点娄和要补充的结点数分别为 n_0，$n_{12}, n_{补}$，则有 $n_0 = 120 + n_{补}$，$n_0 = (12-1)n_{12} + 1$，可得 $n_{12} = (120 - 1 + n_{补})/(12-1)$。

由于结点数 n_{12} 为整数，所以 $n_{补}$ 是使上式整除的最小整数，求得 $n_{补} = 2$，所以答案选 B。

12．解析：

冯·诺依曼结构计算机的功能部件包括输入设备、输出设备、存储器、运算器和控制器，程序的功能都通过中央处理器（运算器和控制器）执行指令，选项 A 正确。指令和数据以同等地位存于存储器内，形式上无差别，只在程序执行时具有不同的含义，选项 B 正确。指令按地址访问，数据由指令的地址码指出，除立即寻址，数据均存放在存储器内，选项 C 错误。在程序执行前，指令和数据需预先存放在存储器中，中央处理器可以从存储器存取代码，选项 D 正确。

13．解析：

unsigned short 类型为无符号短整型，长度为 2 字节，因此 unsigned short usi 转换为二进制代码即 1111 1111 1111 1111。short 类型为短整型，长度为 2 字节，在采用补码的机器上，short si 的二进制代码为 1111 1111 1111 1111，因此 si 的值为-1，所以选 A。

14．解析：

在请求分页系统中，每当要访问的页面不在内存中时，CPU 检测到异常，便会产生缺页中断，请求操作系统将所缺的页调入内存。缺页处理由缺页中断处理程序完成，根据发生缺页故障的地址从外存读入所缺失的页，缺页处理完成后回到发生缺页的指令继续执行。选项 D 中描述回到发生缺页的指令的下一条指令执行，明显错误，所以选 D。

15．解析：

注意，内存地址是无符号数。

操作数采用基址寻址方式，EA = (BR) + A，基址寄存器 BR 的内容为 F000 0000H，形式地址用补码表示为 FF12H 即 1111 1111 0001 0010B，因此有效地址为 F000 0000H + (–00 EEH) = EFFF FF12H。计算机采用大端方式编址，故低位字节存放在字的高地址处，机器数一共占 4 字节，该操作数的 LSB 所在的地址是 EFFF FF12H + 3 = EFFF FF15H，所以选 D。

16．解析：

时钟脉冲信号的宽度称为时钟周期，时钟周期是 CPU 工作的最小时间单位，时钟周期的倒数为机器主频。时钟脉冲信号是由机器脉冲源发出的脉冲信号经整形和分频后形成的，时钟周期以相邻状态单元间组合逻辑电路的最大延迟为基准确定。CPU 从内存中取出并执行一条指令所需的全部时间称为指令周期，指令周期又由若干机器周期来表示，一个机器周期又包含若干时钟周期，选项 D 显然错误。

17．解析：

该指令的两个源操作数分别采用寄存器、寄存器间接寻址方式，因此在取数阶段需要

用到通用寄存器组（GPRs）和存储器（Memory）；在执行阶段，两个源操作数相加需要用到算术逻辑单元（ALU）。而指令译码器（ID）用于对操作码字段进行译码，向控制器提供特定的操作信号，在取数及执行阶段用不到，所以答案选 B。

18．解析：

画出这四条指令在流水线中执行的过程如下图所示。

指令	1	2	3	4	5	6	7	8	9	10	11	12	13	14
add s2, s1, s0	取指	译码/取数	执行	访存	写回									
load s3, 0(t2)		取指	译码/取数	执行	访存	写回								
add s2, s2, s3			取指				译码/取数	执行	访存	写回				
store s2, 0(t2)							取指				译码/取数	执行	访存	写回

数据冒险即数据相关，指在程序中存在必须等前一条指令执行完才能执行后一条指令的情况，此时这两条指令即为数据相关。其中 I1 和 I3、I2 和 I3、I3 和 I4 均发生了写后读相关，因此必须等相关的前一条指令执行完才能执行后一条指令。只有 I2 和 I4 不存在数据冒险，所以答案选 C。

19．解析：

由题目可知，计算机采用 3 通道存储器总线，存储器总线的工作频率为 1333MHz 即 1 秒内传送 1333M 次数据，总线宽度为 64 位，即单条总线工作一次可传输 8 字节（B），因此存储器总线的总带宽为 3×8×1333MB/s，约为 32GB/s，故答案选 B。

20．解析：

磁盘存储器的最小读写单位为一个扇区，即磁盘按块存取，选项 C 错误。磁盘存储数据之前需进行格式化，将磁盘分成扇区，并写入信息，因此磁盘的格式化容量比非格式化容量小，选项 A 正确。磁盘扇区中包含数据、地址和校验等信息，选项 B 正确。磁盘存储器由磁盘控制器、磁盘驱动器和盘片组成，选项 D 正确。

21．解析：

设备接口中的数据缓冲寄存器为 32 位，即一次中断可以传输 4B 数据，设备数据传输率为 50kBps，共需要 12.5k 次中断，每次中断开销为 1000 个时钟周期，CPU 主频为 1GHz，则 CPU 用于该设备输入/输出的时间占整个 CPU 时间的百分比最多是(12.5×1000)/1G = 1.25%。

22．解析：

每类设备都配置一个设备驱动程序，设备驱动程序向上层用户程序提供一组标准接口，负责实现对设备发出各种具体操作指令，用户程序不能直接和 DMA 打交道。DMA 的数据传送过程分为预处理、数据传送和后处理 3 个阶段。预处理阶段由 CPU 完成必要的准备工作，数据传送前由 DMA 控制器请求总线使用权；数据传送由 DMA 控制器直接控制总线完成；传送结束后，DMA 控制器向 CPU 发送中断请求，CPU 执行中断服务程序做 DMA 结束处理。

23．解析：

应用程序没有进行线程管理的代码，只有一个到内核级线程的编程接口，内核为进程及其内部的每个线程维护上下文信息，调度也是在内核中由操作系统完成的，选项 A 正确。在多线程模型中，用户级线程和内核级线程的连接方式分为多对一、一对一、多对多，“操作系统为每个用户线程建立一个线程控制块”属于一对一模型，选项 B 错误。用户级线程

的切换可以在用户空间完成，内核级线程的切换需要操作系统帮助进行调度，故用户级线程的切换效率更高，选项 C 正确。用户级线程的管理工作可以只在用户空间中进行，故可以在不支持内核级线程的操作系统上实现，选项 D 正确。

24．解析：

当被阻塞进程等待的某资源（不包括处理机）为可用时，进程将会被唤醒。I/O 结束后，等待该 I/O 结束而被阻塞的有关进程就会被唤醒，I 正确；某进程退出临界区后，之前因需要进入该临界区而被阻塞的有关进程就会被唤醒，II 正确；当前进程的时间片用完后进入就绪队列等待重新调度，优先级最高的进程将获得处理机资源从就绪态变成执行态，III 错误。

25．解析：

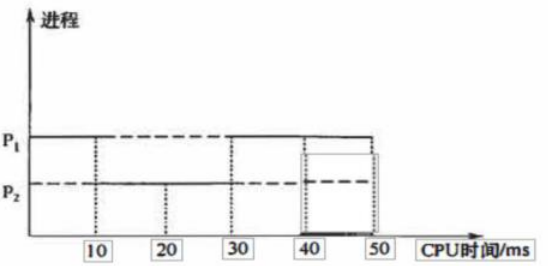
用户可以在用户态调用操作系统的服务，但执行具体的系统调用服务程序是处于内核态的，I 正确；设备管理属于操作系统的职能之一，包括对输入/输出设备的分配、初始化、维护等，用户程序需要通过系统调用使用操作系统的设备管理服务，II 正确；操作系统不同，底层逻辑、实现方式均不相同，为应用程序提供的系统调用接口也不同，III 错误；系统调用是用户在程序中调用操作系统提供的子功能，IV 正确。

26．解析：

传统的文件系统管理空闲磁盘的方法包括空闲表法、空闲链表法、位图法和成组链接法，I、III 正确。文件分配表（FAT）的表项与物理磁盘块一一对应，并且可以用一个特殊的数字-1 表示文件的最后一块，用-2 表示这个磁盘块是空闲的（当然，规定用-3、-4 来表示也是可行的）。因此 FAT 不仅记录了文件中各个块的先后链接关系，同时还标记了空闲的磁盘块，操作系统可以通过 FAT 对文件存储空间进行管理，IV 正确。索引结点是操作系统为了实现文件名与文件信息分开而设计的数据结构，存储了文件描述信息，索引结点属于文件目录管理部分的内容，II 错误。

27．解析：

进程 P_1、P_2 依次创建后进入队列 Q_1，根据时间片调度算法的规则，进程 P_1、P_2 将依次被分配 10ms 的 CPU 时间，两个进程分别执行完一个时间片后都会被转入队列 Q_2，就绪队列 Q_2 采用短进程优先调度算法，此时 P_1 还需要 20ms 的 CPU 时间，P_2 还需要 10ms 的 CPU 时间，所以 P_2 会被优先调度执行，10ms 后进程 P_2 执行完成，之后 P_1 再调度执行，再过 20ms 后 P_1 也执行完成。运行图表述如下。



进程 P_1、P_2 的等待时间分别为图中的虚横线部分，平均等待时间= (P_1 等待时间 + P_2 等待时间) /2 = (20 + 10) /2 = 15，故答案选 C。

28．解析：

段的共享是通过两个作业的段表中相应表项指向被共享的段的同一个物理副本来实现的，

因此在内存中仅保存一份段 S 的内容，选项 A 正确。段 S 对于进程 P_1、P_2 来说，使用位置可能不同，所以在不同进程中的逻辑段号可能不同，选项 B 错误。段表项存放的是段的物理地址（包括段始址和段长度），对于共享段 S 来说物理地址唯一，选项 C 正确。为了保证进程可以顺利使用段 S，段 S 必须确保在没有任何进程使用它（可在段表项中设置共享进程计数）后才能被删除，选项 D 正确。

29．解析：

最近最久未使用（LRU）算法每次执行页面置换时会换出最近最久没有使用过的页面。第一次访问 5 页面时，会把最久未被使用的 1 页面换出，第一次访问 3 页面时，会把最久未访问的 2 页面换出。具体的页面置换情况如下图所示。

访问页面	0	1	2	7	0	5	3	5	0	2	7	6
物理块 1	0	0	0	0	0	0	0	0	0	0	0	0
物理块 2		1	1	1	1	5	5	5	5	5	5	6
物理块 3			2	2	2	2	3	3	3	3	7	7
物理块 4				7	7	7	7	7	7	2	2	2
缺页否	√	√	√	√		√	√			√	√	√

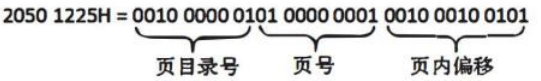
需要注意的是，题中问的是页置换次数，而不是缺页次数，所以前 4 次缺页未换页的情况不考虑在内，答案为 5 次，故选 C。

30．解析：

剥夺进程资源，将其分配给其他死锁进程，可以解除死锁，I 正确。死锁预防是死锁处理策略（死锁预防、死锁避免、死锁检测）中最为严苛的一种策略，破坏死锁产生的 4 个必要条件之一，可以确保系统不发生死锁，II 正确。银行家算法是一种死锁避免算法，用于计算动态资源分配的完全性以避免系统进入死锁状态，不能用于判断系统是否处于死锁，III 错误。通过简化资源分配图可以检测系统是否为死锁状态，当系统出现死锁时，资源分配图不可完全简化，只有两个或两个以上的进程才会出现“环”而不能被简化，IV 正确。

31．解析：

题中给出的是十六进制数地址，首先将它转化为二进制数地址，然后用二进制数地址去匹配题中对应的地址结构。转换为二进制数地址和地址结构的对应关系如下图所示。



前 10 位、11～20 位、21～32 位分别对应页目录号、页号和页内偏移。把页目录号、页号单独拿出，转换为十六进制数时缺少的位数在高位补零，0000 1000 0001、0001 0000 0001 分别对应 081H、101H，选项 A 正确。

32．解析：

最佳适应算法总是匹配与当前大小要求最接近的空闲分区，但是大多数情况下空闲分区的大小不可能完全和当前要求的大小相等，几乎每次分配内存都会产生很小的难以利用的内存块，所以最佳适应算法最容易产生最多的内存碎片，选项 C 正确。

33．解析：

OSI 参考模型自下而上分别为物理层、数据链路层、网络层、传输层、会话层、表示层和应用层。第 5 层为会话层，它的主要功能是管理和协调不同主机上各种进程之间的通信（对话），即负责建立、管理和终止应用程序之间的会话，这也是会话层得名的原因。

34．解析：

100BaseT 是一种以速率 100Mbps 工作的局域网（LAN）标准，它通常被称为快速以太网标准，并使用两对 UTP（非屏蔽双绞线）铜质电缆。100BaseT：100 表示数据传输速率为 100Mbps；Base 表示采用基带传输；T 表示传输介质为双绞线（包括 5 类 UTP 或 1 类 STP），T 为 F 时表示光纤。

35．解析：

从滑动窗口的概念来看，停止等待协议：发送窗口大小=1，接收窗口大小=1；后退 N 协议：发送窗口大小>1，接收窗口大小=1；选择重传协议：发送窗口大小>1，接收窗口大小>1。在选择重传协议中，需要满足：发送窗口大小+接收窗口大小$\leqslant 2^n$；接收窗口大小不应超过发送窗口大小，因此接收窗口大小不应超过序号范围的一半即$\leqslant 2^{(n-1)}$。根据以上规则，采用 3 比特编号，发送窗口大小为 5，接收窗口大小应$\leqslant 3$。

36．解析：

为了确保发送站在发送数据的同时能检测到可能存在的冲突，需要在发送完帧之前就能收到自己发送出去的数据，帧的传输时延至少要两倍于信号在总线中的传播时延，所以 CSMA/CD 总线网中的所有数据帧都必须要大于一个最小帧长，这个最小帧长=总线传播时延×数据传输速率×2。已知最小帧长为 128B，数据传输速率为 100Mbps = 12.5MB/s，计算得单向传播延时为 $128B/(12.5MB/s\times 2) = 5.12\times 10^{-6}s$，即 5.12μs。

37．解析：

网络前缀为 20 位，将 101.200.16.0/20 划分为 5 个子网，为了保证有子网的可分配 IP 地址数尽可能小，即要让其他子网的可分配 IP 地址数尽可能大，不能采用平均划分的方法，而要采用变长的子网划分方法，也就是最大子网用 1 位子网号，第二大子网用 2 位子网号，以此类推。

子网 1：101.200.00010000.00000001～101.200.00010111.11111110；地址范围为 101.200.16.1/21～101.200.23.254/21；可分配的 IP 地址数为 2046 个。

子网 2：101.200.00011000.00000001～101.200.00011011.11111110；地址范围为 101.200.24.1/22～101.200.27.254/22；可分配的 IP 地址数为 1022 个。

子网 3：101.200.00011100.00000001～101.200.00011101.11111110；地址范围为 101.200.28.1/23～101.200.29.254/23；可分配的 IP 地址数为 510 个。

子网 4：101.200.00011110.00000001～101.200.00011110.11111110；地址范围为 101.200.30.1/24～101.200.30.254/24；可分配的 IP 地址数为 254 个。

子网 5：101.200.00011111.00000001～101.200.00011111.11111110；地址范围为 101.200.31.1/24～101.200.31.254/24；可分配的 IP 地址数为 254 个。

综上所述，可能的最小子网的可分配 IP 地址数是 254 个。

38．解析：

TCP 规定当发送方收到对同一个报文段的 3 个重复的确认时，就可以认为跟在这个被确认报文段之后的报文已经丢失，立即执行快速重传算法。t_3 时刻连续收到了来自服务器的三个确认序列号 ack_seq = 100 的段，发送方认为 seq = 100 的段已经丢失，执行快速重传算法，重新发送 seq = 100 段。

39．解析：

根据 TCP 连接建立的“三次握手”原理，第三次握手时甲发出的确认序列号应为第二次握手时乙发出的序列号+1，即 2047。

40．解析：

在 P2P 模型中，每个结点的权利和义务是对等的。在 C/S 模型中，客户是服务发起方，服务器被动接受各地客户的请求，但客户之间不能直接通信，例如 Web 应用中两个浏览器之间并不直接通信。P2P 模型减轻了对某个服务器的计算压力，可以将任务分配到各个结点上，极大提高了系统效率和资源利用率。

二、综合应用题

41．解答：

（1）算法的基本设计思想：

先观察 $L=(a_1,a_2,a_3,\cdots,a_{n-2},a_{n-1},a_n)$ 和 $L'=(a_1,a_n,a_2,a_{n-1},a_3,a_{n-2},\cdots)$，发现 L' 是由 L 摘取第一个结点，再摘取倒数第一个结点……依次合并而成的。为了方便链表后半段取结点，需要先将 L 后半段原地逆置［题目要求空间复杂度为 $O(1)$，不能借助栈］，否则每取最后一个结点都需要遍历一次链表。①先找出链表 L 的中间结点，为此设置两个指针 p 和 q，指针 p 每次走一步，指针 q 每次走两步，当指针 q 到达链尾时，指针 p 正好在链表的中间结点；②然后将 L 的后半段结点原地逆置。③从单链表前后两段中依次各取一个结点，按要求重排。

（2）算法实现：

```
void change_list(NODE*h)
{   NODE *p,*q,*r,*s;
    p=q=h;
    while(q->next!=NULL)                    //寻找中间结点
    {   p=p->next;                          //p 走一步
        q=q->next;
        if(q->next!=NULL) q=q->next;  //q 走两步

    q=p->next;              //p 所指结点为中间结点，q 为后半段链表的首结点
    p->next=NULL;
    while(q!=NULL)          //将链表后半段逆置
    {   r=q->next;
        q->next=p->next;
        p->next=q;
        q=r;

    s=h->next;              //s 指向前半段的第一个结点，即插入点
    q=p->next;              //q 指向后半段的第一个结点
    p->next=NULL;
    while(q!=NULL)          //将链表后半段的结点插入指定位置
    {   r=q->next;          //r 指向后半段的下一个结点
        q->next=s->next;    //将 q 所指结点插入 s 所指结点之后
        s->next=q;
        s=q->next;          //s 指向前半段的下一个插入点
        q=r;
```

（3）第 1 步找中间结点的时间复杂度为 $O(n)$，第 2 步逆置的时间复杂度为 $O(n)$，第 3 步合并链表的时间复杂度为 $O(n)$，所以该算法的时间复杂度为 $O(n)$。

42．解答：

（1）顺序存储无法满足要求②的队列占用空间随着入队操作而增加。根据要求来分析：要求①容易满足；链式存储方便开辟新空间，要求②容易满足；对于要求③，出队后的结点并不真正释放，用队头指针指向新的队头结点，新结点入队时，有空闲结点则无须开辟新空间，赋

值到队尾后的第一个空闲结点即可，然后用队尾指针指向新的队尾结点，这就需要设计成一个首尾相接的循环单链表，类似于循环队列的思想。设置队头、队尾指针后，链式队列的入队操作和出队操作的时间复杂度均为 $O(1)$，要求④可以满足。

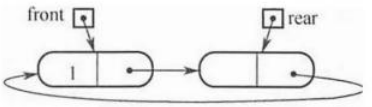
因此，采用链式存储结构（两段式单向循环链表），队头指针为 front，队尾指针为 rear。

（2）该循环链式队列的实现，可以参考循环队列，不同之处在于循环链式队列可以方便增加空间，出队的结点可以循环利用，入队时空间不够也可以动态增加。同样，循环链式队列也要区分队满和队空的情况，这里参考循环队列牺牲一个单元来判断。初始时，创建只有一个空闲结点的循环单链表，头指针 front 和尾指针 rear 均指向空闲结点，如下图所示。

队空的判定条件：front == rear。

队满的判定条件：front == rear->next。

（3）插入第一个结点后的状态如下图所示。



（4）操作的基本过程如下：

```
入队操作：
若（front == rear->next）                    //队满
    则在rear后面插入一个新的空闲结点；
入队元素保存到rear所指结点中；rear=rear->next；返回。
出队操作：
若（front == rear）                          //队空
    则出队失败，返回；
取front所指结点中的元素e；front = front->next；返回e。
```

43．解答：

回顾传统的哲学家问题，假设餐桌上有 n 个哲学家、n 根筷子，那么可以用这种方法避免死锁（王道单科书上提供了这一思路）：限制至多允许 $n-1$ 个哲学家同时“抢”筷子，那么至少会有 1 个哲学家可以获得两根筷子并顺利进餐，于是不可能发生死锁的情况。

本题可以用碗这个限制资源来避免死锁：当碗的数量 m 小于哲学家的数量 n 时，可以直接让碗的资源量等于 m，确保不会出现所有哲学家都拿一侧筷子而无限等待另一侧筷子进而造成死锁的情况；当碗的数量 m 大于等于哲学家的数量 n 时，为了让碗起到同样的限制效果，我们让碗的资源量等于 $n-1$，这样就能保证最多只有 $n-1$ 个哲学家同时进餐，所以得到碗的资源量为 $\min\{n-1, m\}$。在进行 PV 操作时，碗的资源量起限制哲学家取筷子的作用，所以需要先对碗的资源量进行 P 操作。具体过程如下：

```
//信号量
semaphore bowl;                    //用于协调哲学家对碗的使用
```

```
semaphore chopsticks[n];      //用于协调哲学家对筷子的使用
for(int i=0;i<n;i++)
    chopsticks[i]=1;          //设置两个哲学家之间筷子的数盘
bowl=min(n-1,m);              //bowl≤n-1，确保不死锁

CoBegin
while(TRUE){                  //哲学家 i 的程序
    思考;
    P(bowl);                  //取碗
    P(chopsticks[i]);         //取左边筷子
    P(chopsticks[(i+1)%n]);   //取右边筷子
    就餐;
    V(chopsticks[i]);
    V(chopsticks[(i+1)%n]);
    V(bowl);

CoEnd
```

44. 解答：

（1）磁盘容量 = 磁盘的柱面数×每个柱面的磁道数×每个磁道的扇区数×每个扇区的大小 = (300×10×200×512/1024) KB = 3×10^5KB。

（2）磁头在 85 号柱面上，对 SSTF 算法而言，总是访问当前柱面距离最近的地址。注意每个簇包含 2 个扇区，通过计算得到，85 号柱面对应的簇号为 85000～85999。通过比较得出，系统最先访问离 85000～85999 最近的 100260，随后访问离 100260 最近的 101660，然后访问 110560，最后访问 60005。顺序为 100260、101660、110560、60005。

（3）第 100530 簇在磁盘上的物理地址由其所在的柱面号、磁头号、扇区号构成。

柱面号 =⌊簇号/每个柱面的簇数⌋=⌊100530/(10×200/2)⌋= 100。

磁头号 =⌊(簇号%每个柱面的簇数)/每个磁道的簇数⌋=⌊530/(200/2)⌋= 5。

扇区号 = 扇区地址%每个磁道的扇区数 =(530×2)%200 = 60。

将簇号转换成磁盘物理地址的过程由磁盘驱动程序完成。

45. 解答：

（1）计算 $f(10)$需要调用函数 f1 共 10 次，执行第 16 行的 call 指令会递归调用 f1。

（2）第 12 行的 jle 指令是条件转移指令，其含义为小于等于时转移，本行代码的意义为：当 $n\leqslant1$ 时，跳转至地址 0040 1035H。第 16 行的 call 指令为函数调用指令，第 20 行的 jmp 指令为无条件转移指令，第 30 行的 ret 指令为子程序的返回指令，这三条指令一定会使程序跳转执行。

（3）其长度计算机 M 上按字节编址，第 16 行的 call 指令的虚拟地址为 0040 1025H，长度为 5 字节，故第 17 行的指令的虚拟地址为 0040 1025H + 5 = 0040 102AH。第 16 行的 call 指令采用相对寻址方式，即目标地址 =(PC)+ 偏移量，call 指令的目标地址为 0040 1000H，所以偏移量=目标地址 -(PC)= 0040 1000H - 0040 102AH = FFFF FFD6H。根据第 16 行的 call 指令的偏移量为 D6 FF FF FF，可以确定 M 采用小端方式。

4）因为 $f(13)$= 6227020800，其结果超出了 32 位 int 型数据可表示的最大范围，因此 $f(13)$的返回值是一个发生了溢出的错误结果。为使 f1(13)能返回正确结果，可将函数 f1 的返回值类型改为 double（或 long long，或 long double，或 float）型。

5）若乘积的高 33 位为非全 0 或非全 1，则 OF = 1。编译器应在 imul 指令后加一条“溢出自陷指令”，使得 CPU 自动查询溢出标志 OF，当 OF = 1 时调出“溢出异常处理程序”。

46．解答：

因为页大小为 4KB，所以虚拟地址的高 20 位为虚拟页号。第 1 行的 push 指令和第 30 行的 ret 指令的虚拟地址的高 20 位都是 00401H，因此两条指令在同一页中。

指令 Cache 有 64 块，采用 4 路组相联映射方式，故指令 Cache 共有 64/4 =16 组，Cache 组号共 4 位。主存块大小为 64B，所以块内地址为低 6 位。综上所述，在 32 位主存地址中，低 6 位为块内地址，中间 4 位为组号，高 22 位为标记。

因为页大小为 4KB，所以虚拟地址和物理地址的最低 12 位完全相同，因而 call 指令虚拟地址 0040 1025H 中的 025H = 0000 0010 0101B 为物理地址的低 12 位，对应的 7～10 位为组号，故对应的 Cache 组号为 0。

47．解答：

（1）以太网交换机（无 VLAN 功能）连接的若干 LAN 仍然是一个网络（同一个广播域），路由器可以连接不同的 LAN、不同的 WAN 或把 WAN 和 LAN 互连起来，隔离了广播域。IP 地址 192.168.1.2/26 与 192.168.1.3/26 的网络前缀均为 192.168.1.0，视为 LAN1。IP 地址 192.168.1.66/26 与 192.168.1.67/26 的网络前缀均为 192.168.1.64，视为 LAN2。所以设备 1 为路由器，设备 2、3 为以太网交换机。

（2）设备 1 为路由器，其接口应配置 IP 地址。IF1 接口与路由器 R 相连，其相连接口的 IP 地址为 192.168.1.253/30，253 的二进制数表示形式为 11111101，故 IF1 接口的网络前缀也应为 192.168.1.111111，已分配 192.168.1.253，去除全 0 全 1，IF1 接口的 IP 地址应为 192.168.1.254。LAN1 的默认网关为 192.168.1.1，LAN2 的默认网关为 192.168.1.65，网关的 IP 地址是具有路由功能的设备的 IP 地址，通常默认网关地址就是路由器中的 LAN 端口地址，设备 1 的 IF2、IF3 接口的 IP 地址分别设置为 192.168.1.1 和 192.168.1.65。

（3）私有地址段：C 类 192.168.0.0~192.168.255.255，即 H1～H4 均为私有 IP 地址，若要能够访问 Int ernet，R 需要提供 NAT 服务，即网络地址转换服务。

（4）主机 H3 发送一个目的地址为 192.168.1.127 的 IP 数据报，主机号全为 1，为本网络的广播地址，由于路由器可以隔离广播域，只有主机 H4 会接收到数据报。