2018 年硕士研究生入学考试试题（回忆版）

操作系统

一、填空题 4 个（2 分*5=10 分）

1、__pcb__是进程存在的唯一标志 1 分（考察进程的文件控制块 PCB）

2、逻辑页面 16 个，物理块 64 个，页的大小为 4KB，逻辑页号占___4____位，物理页号占____6___位。 2 分（考察分页页号和物理块号所占位数）

3、若信号量的初始值为 4，当前值为-1，则说明有（1）个进程在等待此信号量（考察信号量的值，负值代表什么，正值代表什么）

4、设文件索引结点有 4 个地址项，2 个直接地址，2 个一级间接地址，每个地址项大小为 4B，若每个磁盘索引块的大小均为 4KB，则可表示的单个文件的最大长度为（8MB+8KB）

答：每个磁盘索引块可以存放 4KB/4B=1K 个地址项

2 个直接地址对应文件大小：2*1K*4B=8KB

2 个一级间接地址对应文件：2*1K*1K*4B=8MB

所以单个文件最大长度为 8MB+8KB

（考察根据文件直接地址、一次间接地址和二次间接地址计算文件大小）

二、选择题 2 分*5=10 分

1. （考了个**磁盘黏着**的概念）不会产生磁道胶着的是

（这是比较新的概念，主要考察磁盘，在王道上没有提到，应该去看看课本）

2. 隐式链接（不要记得具体考什么，大概是隐式连接与显式连接的区别）

3. 6 个进程，每个需要 4 台打印机，不会产生死锁的最少台数是（19）

（考察造成死锁的最低的设备个数）

4. 哪种不属于 IO 控制方式（不是那四个之中的一个）

5. 进程状态转换（哪种状态转换是不能发生的：阻塞-运行）

三、简答题 20 分

1 死锁预防与死锁避免的区别（6 分）

答：（1）死锁预防是通过破坏死锁的四个必要条件之一来防止死锁的发生。

破坏互斥条件：在系统里取消互斥。若资源不被一个进程独占使用，那么死锁是肯定不会发生的。但一般来说在所列的四个条件中，“互斥”条件是无法破坏的。

破坏请求和保持条件：即在进程在其运行之前一次性申请说需求的所有资源，在它的资源未满足前，不分配资源，不投入运行

破坏不可剥夺条件：即破坏“不可抢占”条件就是允许对资源实行抢夺。

破坏循环等待条件：破坏“循环等待”条件的一种方法，是将系统中的所有资源统一编号，进程可在任何时刻提出资源申请，但所有申请必须按照资源的编号顺序（升序）提出。这样做就能保证系统不出现死锁

（2）死锁避免是在资源的动态分配过程中，用某种方法防止系统进入不安全状态，从而避免死锁。一般使用的是银行家算法。

2 抖动的原因，怎样解决（7 分）

答：（1）抖动的定义：在页面置换过程中，刚刚换出的页面马上又要换入主存，刚刚换入的页面又要换出主存，这种频繁的页面调度行为叫做抖动

（2）抖动的原因：某个进程频繁访问的页面数目高于可用的物理块数。

（3）解决方案：增加物理块（对先进先出算法不管用）；选择适当的页面置换算法等。

3 系统怎样实现文件共享（7 分）

答：（1）基于索引结点的共享方式(硬连接)

不同目录下使用相同索引结点，在索引结点中再增加一个计数值来统计指向该索引结点的目录项个数，只有计数值为 1 时才可删除该索引结点，若计数值大于 1 删除时把计数值减 1 即可，增加共享时计数值加 1。

优点：实现文件的异名共享

缺点：当文件被多个用户共享时，文件拥有者不能删除文件。

在硬连接中，目录项中只有文件名，指向的是共同的索引结点，索引结点再指向文件，所以一个文件名修改索引结点，其他所有文件名的索引结点都被修改了，因为所有文件名指向同一个索引结点。

（2）利用符号链实现文件共享(软连接)

相当于把共享文件的路径副本复制过来(并不拥有文件)，并且只有在文件试图去访问时才会更新其路径副本。

优点：解决了文件拥有者不能够删除共享了的文件的问题

缺点：当其他用户要访问共享文件时，需要逐层查找目录，开销较大。

（具体解释见王道书上关于这两个方式的讲解）

四、计算题（35 分）

1、（红果研相似题）某计算机的逻辑地址空间和物理地址空间均为 64KB，按字节编址。若某进程最多需要 6 页存储空间，页的大小为 1KB。操作系统采用固定分配局部置换为此进程分配四个页框，如下表所示。

页号	页框号	装入时刻	访问位
0	3	100	1
1	4	200	1
2	2	230	1
3	5	160	1

若该进程执行到 260 时刻，要访问逻辑地址为 104AH 的数据，请回答：

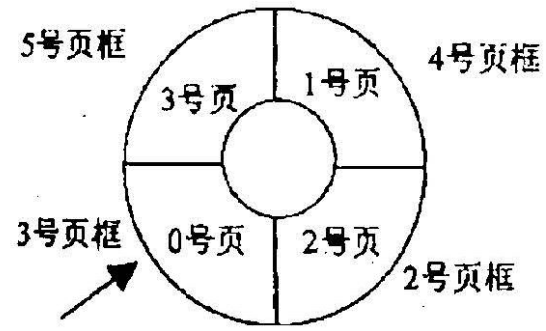
（1） 该逻辑地址对应的页号是多少，页内偏移量是多少？

答：104AH=0001 0000 0100 1010

页面大小 1KB，所以页内偏移量占据 10 位，即为 00 0100 1010=74

页号占据 6 位，即 0001 00=4

（2）若采用 clock 置换算法，该逻辑地址对应的物理地址是多少？要求给出计算过程（设搜索下一页的指针沿着顺时针方向移动，且当前指针指向 3 号页框，如下图所示）



答：Clock 算法从 0 号页开始查找，前 4 次查找页框顺序为 3-5-4-2，并将所有页框的使用位置为 0，在第五次查找中，指针指向 3 号页框，使用位为 0，故淘汰 3 号页框所对应的 0 号页面，把需要的页面转入 3 号页框，并把访问位置为 1.

对应的物理地址为 3*1KB（0000 1100 0000 0000）+74（0000 0000 0100 1010）=0000 1100 0100 1010=0C4AH

（原题模型为王道 190 页第 14 题，基本上原题，更改了访问位和修改位，本质是考察页面置换算法，看把那个逻辑页面置换出去，然后这个逻辑页面对应的物理帧给这个新来的页面）

2、订票与查询，可多个查询者，订票者与查询者不可同时操作且要求按请求顺序提供服务（互斥访问）订票者，不可与查票者同时访问，订票者之间，也不可以。要求，按用户请求次序执行操作（20 分）

答：（读写者问题，并且要保证读进程和写进程都不优先）

```
int count=0;   //用于记录当前的查询者数量
semaphore mutex=1;   //用于对 count 变量值的互斥访问
semaphore rw=1;   //用于保证查询者和订票者互斥地访问系统
semaphore w=1;   //用于保证查询者和订票者都不优先
Order () {   //订票者进程
while (1){
        P(w); //  在无查询者时请求进入
        P(rw); //  互斥访问系统
```

```
            Order ticket;   //订票
            V(rw) ;   //释放系统访问
            V(w) ;   //表示查询者可以进入
        }
}


Check () {   //  查询者进程
while(1){
            P(w); //  在无订票者时请求进入
            P (mutex) ;   //互斥访问 count 变量
            if (count==0)   //当第一个查询者进入系统时
                P(rw);   //阻止订票者订票
            count++;   //读者计数器加 1
            V (mutex) ;   //释放互斥变量 count
            V(w) ;   //恢复对系统的访问
            reading;   //查询
            P (mutex) ;   //互斥访问 count 变量
            count--; //读者计数器减 1
            if (count==0)   //当最后一个查询者离开系统时
                V(rw) ;   //允许订票者进入系统
            V (mutex) ;   //释放互斥变量  count
        }
}
```

数据结构

一、填空（10 分）

1. 数据的逻辑结构分为 线性结构 和 非线性结构 。

2. 对于快速排序和堆排序，如果一组数据处于基本上有序的状态时，应该用 快速排序 算法；如果处于基本上无序时，应选用 堆排序 算法。

3. 弗洛伊德求最短路径的时间复杂度为 O（|V^3|） 。

4. 对于快速排序，堆排序，简单选择排序，冒泡排序来说， 快速排序 的空间复杂度最高，其空间复杂度为 O（$\log_2 n$） 。

5. n 个顶点 e 条边的无向图，利用邻接矩阵存储时的空间复杂度为 O（n^2） ，利用邻接表存储时的空间复杂度 O(|n|+2|e|) 。

6. 图 G 有 n 个顶点和 e 条边，求 G 的最小生成树，采用 prim 算法的时间复杂度为（O(n^2)）

二、选择 （20 分）

1. 对于 n 个序列，每个序列里面含有 m 个有序序列，则将他们整体排列成有序序列时，其算法时间复杂度为 （ D ）（答案不确定是否正确）

A. O（m） B.O（n） C.O（m*n） D.O（$\log_2 n$）

2. 对于 5，7，8，1，2，3，4，11，12 第一个数字为准，对其进行一次快速排序的结果是（ A ）

A.4 3 2 1 5 8 7 11 12 B.4 2 3 1 5 8 7 11 12

C.4 1 2 3 5 7 8 11 12 D.4 1 3 2 5 7 8 11 12

3. 下列对于广义表的说法错误的是(A)

A. 对于广义表进行取表尾操作时则得到该表表尾

B. 对于广义表取表头操作则得到的就是表头

C. 广义表是一种递归的数据结构

D. 广义表的定义为是无限的

4.无向完全图的邻接矩阵和邻接表表示法

5.各种排序算法需要比较多少次以及哪些算法不能保证每一次比较都有元素落在最终位置上

6.已知中序和后序，是否可以求前序

7、图的广度和深度优先搜索与二叉树的先序中序后序遍历

8、循环队列判断队空队满条件

9、栈的各种输出序列

10、广义表取表头表尾操作

三、简答题：（30 分）

1. 如何用双栈实现队列功能（5 分）

答：算法思想：先把元素按顺序进入栈 1，然后出栈 1 进入栈 2，再从栈 2 输出就是队列先进先出的特性。（自己写的时候稍加扩展）

2. 请简述求自然数 a，b 的最大公约数，最小公倍数算法思想（5 分）

答：

```
int main() //主函数
{
int w,t,n,m,z,p; //定义变量类型，在使用变量之前，一定要先定义变量，定义时要包括变量的类型
scanf("%d%d",&n,&m); //用输入函数，在键盘上输入两个值
w=n*m;
while(m!=0) //使用循环，用辗转相除法求出最大公约数
{
if(n<m) //判断两个数的大小，必须是较大的数除去较小的数
{
p=n;
n=m;
m=p;
}
```

```
t=n%m;
n=m;
m=t;
}
z=w/n; //利用求出的最大公约数求出最小公倍数
printf("最大公约数=%d\n 最小公倍数=%d",n,z); //输出求出的值
return 0;
}
```

（伪代码不会写，写出算法思路也会得分）

3. 对（5 2 3 8 11 7 13 20）进行排序要求不平衡时调整为平衡二叉树。（5 分）

答：（这类题王道和真题上很多，多做几道会了就可以）

4. 对（14 16 26 23 48 19 20）进行 hash 函数查找，且 hash（key）=key%7 用链地址法解决冲求出查找成功的 ASL.（5 分）

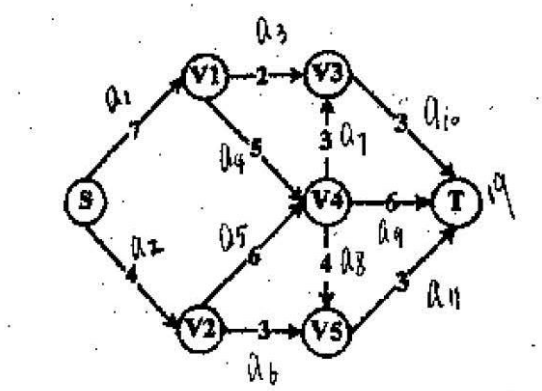
答：（这类题王道和真题上很多，多做几道会了就可以）

5. 对（2 3 5 7 8 20）构造最优二叉树，且求出其权值。（5 分）

答：（这类题王道和真题上很多，多做几道会了就可以）

6. 用双标记号法实现关键路径。（5 分）

答：以红果研例题为例：



顶点	Ve（最早开始时间）	Vl（最迟开始时间）
S	0	0
V1	7	7
V2	4	6
V3	15	16
V4	12	12
V5	16	16
T	19	19

关键路径为 S-V1-V4-V5-T

四、算法题：（15 分）

1 .merge 算法（按升序合并两个有序链表，并递减输出，是王道上原题）；（7 分）

答：void mergeList(LinkList &La,LinkList &Lb)

{//合并两个有序递增链表 La，Lb，并有序递减输出

Lnode *r,*pa=La->next,*pb=Lb->next;

La->next==null;

```
while(pa&&pb)//两链表都不为空
{
    if(pa->num<=pb->num)
    {
        r=pa->next;
        pa->next=La->next;
        La->next=pa;
    }
    else
    {
        r=pb->next;
        pb->next=La->next;
        La->next=pb;
        pb=r;
    }
}
if(pa)//最后剩下 pa 不为空
{
    pb=pa;
}
while(pb)//最后剩下 pb 不为空
{
    r=pb->next;
    pb->next=La->next;
    La->next=pb;
    pb=r;
}
free(pb);
```

```
    }
```

2 .判断二叉树 A 里面是否有子树的结构和数值和树 B 一样（代码摘自 CSDN 博客，用 Java 写的）（8 分）

答：算法思想：

1 先采用遍历找出树一中是否含有树二的头结点

2 如果不含有，返回 false，如果含有使用 isSubTree 判断树二是否是树一的子树

联想

```
public class Solution {
    private class TreeNode {
        int val = 0;
        TreeNode left = null;
        TreeNode right = null;

        public TreeNode(int val) {
            this.val = val;

        }

    }
    //HasSubtree 主要是判断树 root1 中是否含有 root2 节点
    public boolean HasSubtree(TreeNode root1,TreeNode root2) {
        if(root2==null) return false;
        if(root1==null && root2!=null) return false;
        boolean flag = false ;
        if(root1.val==root2.val){
            flag = isSubTree(root1,root2);
```

```
                }
                return flag || HasSubtree(root1.left, root2) || HasSubtree(root1.right,
root2) ;
            }

            private boolean isSubTree(TreeNode root1, TreeNode root2) {
                if(root2==null) return true;
                if(root1==null && root2!=null) return false;
                if(root1.val==root2.val){
                    return isSubTree(root1.left, root2.left) &&
isSubTree(root1.right, root2.right);
                }
                return false;
            }
        }
```