2015 年硕士学位研究生入学考试试题参考答案

考试科目：872 数据结构与操作系统

一、数据结构部分

1、判断

（1）错误，出栈操作要遵循先进后出的原则，第一个是 100，故出栈顺序一定，第 50 个输出元素应该是 51。

（2）正确。

（3）错误，i 到 j 可能不存在路径。

（4）正确，装填因子 a 的定义知道 a=n/m 其中 n 为关键字个数，m 为表长，a 越小，说明表长越长，冲突就越小。

（5）错误，最小生成树不唯一，所有的最小生成树的各个路径的权值之和最小且唯一。

2、（1）堆排序的基本思想：排序过程最关键的是将关键字序列调整为堆。以大顶堆为例：

① 先将初始关键字 A[1..n]建成一个大根堆，此堆为初始的无序区

② 再将关键字最大的记录 A[1](即堆顶)和无序区的最后一个记录 A[n]交换，由此得到新的无序区 A[1..n-1]和有序区 A[n]，且满足 A [1..n-1]≤A [n]

③ 重新将当前无序区 A [1..n-1]递归调整为堆。

（2）对快速排序而言，堆排序在最坏的情况下时间复杂度也是 $O(n\log_2 n)$,空间复杂度为 O(1),而快速排序空间复杂度为 $O(\log_2 n)$；对归并排序而言，堆排序空间复杂度为 O(1),而归并排序空间复杂度为 O(n).

（3）采用堆排序好，根据大顶堆排序的基本思想，只需要选出 10 个最大的数依次调整为堆，而不必等把全部数排序好后选出，堆排序最快。

3、 思路: 一个由字符元素够长的二叉树以完全二叉树的数组结构进行存储，假设存储在数组 A（其大小为 max，足够大）中。其中的结点若没有左孩子或右孩子，则在数组中的元素的值设为' '，即为空字符（这里可以假设为其他字符）。二叉树结点 A[i]的左孩子为 A[2i]，右孩子为 A[2i+1]。采用递归方式创建该二叉树的链接存储表示。具体的算法如下：

二叉树链表的结构体定义：

```
typedef struct    BiTree
{     char data;
      struct BiTree *lchild;
      struct BiTree *rchild;
 } BiTreeNode ;

void CreatBiTree(BiTreeNode  *&t, char A[], int i)
{   if  (i>=max  || A[i]==' '){   //若 i 无效或 A[i]为无效结点
          t=NULL;
    }else
    {
        t=( BiTreeNode   *)malloc(sizeof(BiTreeNode));
```

```
            t->data=A[i-1];
            if(A[2*i]!= ' ') {
               CreatBiTree (t->lchild,A,2*i);  //递归构造*t 的左子树
            }
            if(A[2*i +1]!= ' ') {
               CreatBiTree (t->rchild,A,2*i+1); //递归构造*t 的右子树
            }
        }
    }
```

4、思路：双向链表有序，使用两个指针：头指针 pNode 和尾指针 qNode，分别从前往后和从后往前遍历双向链表，判断两个指针指向的结点的值之和是否等于 x，若等于 x，则输出，若小于 x，则 pNode 结点向后移动一位，若大于 x，则 qNode 结点向前移动一位。具体算法如下：

```
typedef struct dLinkNode
{
    int data;
    struct dLinkNode* pre;
    struct dLinkNode* next;
}dLinkNode;
```

```
void find(dLinkNode* pNode, int x){
    dLinkNode* qNode = null;
    dLinkNode* temp = pNode;
    if(null == pNode){
        return ;
    }
     // qNode 指向双向链表的尾结点
    while(temp != null){
       qNode = temp;
       temp = temp->next;
    }
    while(pNode->data <= qNode->data){
        int sum = pNode->data    +    qNode->data;
        if(sum < x){
           pNode = pNode->next;
        }else if(sum > x){
           qNode = qNode ->pre;
        }else{
           print("x=%d + %d",pNode->data,qNode->data);
        }
    }
}
```

5、思路：两个有序数组的中间值：如果两个数组的长度之和为奇数，则中间值为两个数组

合并后的中间位置的那个数，若两个数组的长度之和为偶数，则中间值为两个数组合并后的位置为（长度之和 - 1）/2 的那个数。

方法一：两个数组合并为一个有序递增的数组，然后找到中间值（可以只合并前半部分）时间复杂度：O(m+n)，空间复杂度 O((m+n) /2)

方法二：利用计数器 cout，遍历两个有序数组，同时用变量记下 cout 位置时的值，直到 cout 等于 k（k 为要寻找的位置）值时，此时变量的值即是所求值。时间复杂度：O(m+n)，空间复杂度 O(1)

方法三：由于两个数组都递增有序，可以用二分查找的思想寻找中间值的位置。时间复杂度：O(log(m+n))，空间复杂度 O(1)

具体算法实现如下：（方法二比较容易想到，这里用 java 实现）

```
public static int findMidValue(int[] a, int[] b) {
    int k = (a.length + b.length - 1) / 2;
    int i = 0, j = 0;
    int midValue = 0; // 保存 cout 位置的值
    while (i < a.length && j < b.length && k >= 0) {
        midValue = (a[i] < b[j]) ? a[i++] : b[j++];
        k--;
    }
    while (i < a.length && k >= 0) {
        midValue = a[i++];
        k--;
    }
    while (j < b.length && k >= 0) {
        midValue = b[j++];
        k--;
    }
    return midValue;
}
```

二、操作系统部分

6、判断

（1）错误，不是所有的进程都常驻内存的，很多情况下只是正在执行的进程在内存中。

（2）正确，发生死锁的进程至少有两个进程。

（3）错误， SCAN 算法寻道性能较好，可避免“饥饿”现象。

（4）错误，内存中的进程数增加，不一定能增加 cpu 的利用率。

（5）错误，引入 TLB 只是减少查找对应页表项的时间，不能减少每次访问内存的时间。

7、（1）1GB = 1024MB = 1024*1024KB = 1024*1024*1024 bit = 2^{30} bit

故主存地址位数：30

（2）主存中的页框，即是主存中的物理块；物理块的大小与页面的大小相等，故页框的大小为 4KB，4KB = 2^{12} bit， 2^{30} bit / 4kB = 2^{18}，故页框的个数为 2^{18}

（3）页面大小为 4KB = 2^{12} bit，主存地址为 30 位，所以逻辑地址的页内偏移应该用 18 位来表示。

（4）页号 3 对应的页框为 134，故物理地址为：134*4KB + 1 = 548864 + 1 = 548865

（5）页框个数为 2^{18}，故位示图大小为：$(2^{18})^2 = 2^{36}$

8、五个进程先后到达，根据优先级来确定先执行某个进程，优先数随着时间的增加是动态变化的，所有的进程执行完总的时间是 11s，可分析每 1s 后各个进程的优先数变化及执行时间，到达时间。

（1）第 1s 时，P1 先到达，则 P1 先执行，1s 后，各个进程的参数状态如下：

进程	还需执行时间	达到时间	优先级 p	等待时间
P1	2	0	8	0
P2	2	0	3	1
P3	3	1	5	1
P4	1	2	1	1
P5	2	3	9	1

第 2s 时，P1,P2 都到达了，但 P2 的优先级高于 P1，故 P2 先执行，2s 后，各个进程的参数状态如下：

进程	还需执行时间	达到时间	优先级 p	等待时间
P1	2	0	7	1
P2	1	0	5	1
P3	3	0	3	2
P4	1	1	-1	2
P5	2	2	7	2

第 3s 时，P3 也到达了，P3 的优先级最高，故 P3 先执行；3s 后，各个进程的参数状态如下：

进程	还需执行时间	达到时间	优先级 p	等待时间
P1	2	0	5	2
P2	1	0	2	2
P3	2	0	7	2
P4	1	0	-4	3
P5	2	1	4	3

第 4s 时，P4 也到达了，P4 的优先级最高，故 P4 先执行；4s 后，各个进程的参数状态如下：

进程	还需执行时间	达到时间	优先级 p	等待时间
P1	2	0	3	3
P2	1	0	0	3
P3	2	0	5	3
P4	0	0	2	3
P5	2	0	1	4

第 5s 时，P5 也到达了，P4 执行完了，此时只有 P1，P2，P3，P5，但 P2 的优先级最高，故 P2 先执行；5s 后，各个进程的参数状态如下：

进程	还需执行时间	达到时间	优先级 p	等待时间
P1	2	0	-1	4
P2	0	0	6	3
P3	2	0	1	4

P4	0	0	2	3
P5	2	0	-4	5

第 6s 时，P4 执行完了，P2 执行完了，此时只有 P1，P3，P5，但 P5 的优先级最高，故 P5 先执行；6s 后，各个进程的参数状态如下：

进程	还需执行时间	达到时间	优先级 p	等待时间
P1	2	0	-5	5
P2	0	0	6	3
P3	2	0	-3	5
P4	0	0	2	3
P5	1	0	6	5

第 7s 时，P4 执行完了，P2 执行完了，此时只有 P1，P3，P5，但 P1 的优先级最高，故 P1 先执行；7s 后，各个进程的参数状态如下：

进程	还需执行时间	达到时间	优先级 p	等待时间
P1	1	0	5	5
P2	0	0	6	3
P3	2	0	-9	6
P4	0	0	2	3
P5	1	0	0	6

第 8s 时，P4 执行完了，P2 执行完了，此时只有 P1，P3，P5，但 P3 的优先级最高，故 P3 先执行；8s 后，各个进程的参数状态如下：

进程	还需执行时间	达到时间	优先级 p	等待时间
P1	1	0	-1	6
P2	0	0	6	3
P3	1	0	3	6
P4	0	0	2	3
P5	1	0	-7	7

第 9s 时，P4 执行完了，P2 执行完了，此时只有 P1，P3，P5，但 P5 的优先级最高，故 P5 先执行；9s 后，各个进程的参数状态如下：

进程	还需执行时间	达到时间	优先级 p	等待时间
P1	1	0	-8	7
P2	0	0	6	3
P3	1	0	-4	7
P4	0	0	2	3
P5	0	0	7	7

第 10s 时，P4，P2，P5 执行完了，此时只有 P1，P3，但 P1 的优先级最高，故 P1 先执行；10s 后，各个进程的参数状态如下：

进程	还需执行时间	达到时间	优先级 p	等待时间
P1	0	0	6	7
P2	0	0	6	3
P3	1	0	-12	8
P4	0	0	2	3
P5	0	0	7	7

第 11s 时，P4，P2，P5，P1 执行完了，此时只有 P3，故 P3 先执行；11s 后，各个进程

的参数状态如下：

进程	还需执行时间	达到时间	优先级 p	等待时间
P1	0	0	6	7
P2	0	0	6	3
P3	0	0	4	8
P4	0	0	2	3
P5	0	0	7	7

故 5 个进程执行的顺序图：P1-P2-P3-P4-P2-P5-P1-P3-P5-P1-P3

(2) 周转时间= 完成时间 - 提交时间，

响应时间是指进程首次运行的时间点，两者时间如下表：

进程	周转时间	响应时间	等待时间
P1	10	0	7
P2	5	1	3
P3	11	2	8
P4	4	3	3
P5	9	5	7

9、据题意，定义相应的信号量和各个进程描述如下：

Semaphore mutex = 1；//信号量 mutex 用于实现对缓冲区的互斥操作，其初值为 1；

Semaphore full = 0；//信号量 full 用于表示缓冲区中是否取消息，其初值为 0；

Semaphore empty = 1；//信号量 empty 用于表示 B 是否发送消息，其初值为 1；

Semaphore $s_1 = 0$； //信号量 s_1 用来控制 A_1 是否可以取出消息，其初值为 0；

Semaphore $s_2 = 0$； //信号量 s_2 用来控制 A_2 是否可以取出消息，其初值为 0；

……

Semaphore $s_n = 0$； //信号量 s_n 用来控制 A_n 是否可以取出消息，其初值为 0；

```
Procedure B {
      while(true){
         P(empty)
         P(mutex);
          向缓冲区发送消息;
         V(mutex);
         V(full)
         V(s1);
  }
}

Procedure A1{
    while(true){
         P(s1);
         P(full)
         P(mutex)
          从缓冲区中取消息;
         V(mutex)
         P(full)
```

```
        V(s2);
    }
}

Procedure A2{
    while(true){
        P(s2);
        P(full)
        p(mutex);
         从缓冲区中取消息；
        V(mutex)
        V(full)
        V(s3);
    }
}
……

Procedure An{
    while(true){
        P(sn);
        P(full)
        P(mutex);
         从缓冲区中取出消息；
        V(mutex);
        V(empty)
    }
}
```

10、

根据题意，可以估算出每条记录的长度，其中姓名占 4*2＝8 byte，年龄占 2 byte，住址占 128*2＝256 byte，身份证占 18 byte，性别占 2 byte，故一条记录总的占 286＋4(空格所占)＝290 byte；文件总的字节数大约为： 290000000 byte ＝290000kb ＝ 290MB

设计的方案：

a、文件的逻辑结构：由于对此文件的操作主要是根据姓名进行记录的查询，因此可以根据姓名的长度对文件进行分目录存储，即姓名相同长度的分在同一个目录，最多有 63 个目录，每个目录中的文件长度差不多，因此可以将这个文件的逻辑文件信息连续存放，即采用顺序文件的方式。

b、文件的物理结构：由于该文件的大小已知，故可以采用连续分配的方式，把逻辑文件中的记录顺序地存储到相邻的物理盘块中；这样查找速度快，且没有增加其他额外空间。

（1）需要 290 000 个磁盘块

（2）平均需要访问 290000/2＝145000 个磁盘块