

所有试题答案写在答题纸上, 答案写在试卷上无效

## 《数据结构》(共 70 分)

### 一、选择题 (每小题 1.5 分, 共 9 分)

1. 某线性表中最常用的操作是在最后一个元素之后插入一个元素和删除第一个元素, 则采用 ( ) 存储方式最节省运算时间。

- A. 非循环的单链表  
B. 仅有头指针的单循环链表  
C. 非循环的双链表  
D. 仅有尾指针的单循环链表

2. 以下与数据的存储结构无关的术语是 ( )。

- A. 循环队列  
B. 链表  
C. 哈希表  
D. 栈

3. 一个栈的输入序列为 1, 2, 3, ..., n, 若输出序列的第一个元素是 n, 输出第  $i$  ( $1 \leq i \leq n$ ) 个元素是 ( )。

- A. 不确定  
B.  $n-i+1$   
C.  $i$   
D.  $n-i$

4. 已知广义表  $LS = ((a, b), (d, e, f))$ , 运用 head 和 tail 函数取出 LS 中原子 e 的运算是 ( )。

- A. head( tail(LS) )  
B. tail( head(LS) )  
C. head( tail( head( tail(LS) ) ) )  
D. head( tail( tail( head(LS) ) ) )

5. 算术表达式  $a+b*(c+d/e)$  转为后缀表达式后为 ( )。

- A.  $ab+cd+e/*$   
B.  $abcde/+*+$   
C.  $abcde/*++$   
D.  $abcde*/++$

6. B+树应用在 ( ) 文件系统中。

- A. ISAM  
B. VSAM

### 二、问答题 (共 36 分)

1. (4 分) 设指针 p 指向双向链表中的一个结点, 请写出在 p 所指结点之后插入由 s 所指向的结点的操作序列。

2. (4 分) 设有关键字 10, 20, 30, 40 和 50, 依照不同的输入顺序, 共可能组成多少棵不同的二叉排序树。请说明推导理由。

3. (6 分) 假设二维数组 A 的维界为  $[-2..7, 3..6]$ , 当它在内存中按行存放和按列存放时, 分别写出数组元素  $A[i, j]$  的地址计算公式 (设每个元素占两个存储单元)。

4. (8 分) 设有一棵空的 3 阶 B-树, 依次插入关键字 32, 18, 10, 40, 60, 58, 47, 50, 29, 22, 要求:

(1) 画出该 3 阶 B-树;

(2) 画出在该 3 阶 B-树中删除关键字 32 后的树的形态。

5. (10 分) 已知二叉树的先序遍历序列为 ABCELMNDFGHK, 中序遍历序列为 CBLMNEAFDHKG。

要求:

- (1). 画出该二叉树;
- (2). 画出该二叉树对应的后序线索二叉树;
- (3). 画出与该二叉树相对应的森林。

6. (4 分) 在含有  $n(n>0)$  个关键字的小根堆 (堆顶元素最小) 中, 关键字最大的记录可能存储在什么位置上? 说明理由。

### 三、算法设计 (共 25 分)

1. (9 分) 编写一个算法, 输出二叉树中距给定节点最近的叶子子孙 (可以是给定节点的孩子)。

注: 二叉树用二叉链表示。

2. (16 分) 编写克鲁斯卡尔算法求无向连通网的最小生成树, 并分析你所编写的算法的时间和空间复杂度。

注: 采用邻接表保存无向连通网。

### 《操作系统》(共 40 分)

#### 一、简答题 (每小题 6 分, 共 24 分)

1. 请简述“微内核”的含义及特点。
2. 请描述用于死锁避免的银行家算法的主要思想并列出其需要使用的数据结构; 请简单分析这种算法能解决实际问题中的死锁吗?
3. 请给出实现页面淘汰的最近最少使用算法 LRU 的基本原理和软件实现方法。
4. 在设计批处理作业调度算法时, 一般需要考虑哪些因素? 可以采用哪些衡量标准来评价设计出的作业调度算法?

#### 二、分析题 (每小题 8 分, 共 16 分)

1. 在设计操作系统的进程调度算法时, 需要综合考虑优先级别、等待时间等因素; 实际的操作系统并不是单纯使用某一种进程调度算法, 而是综合多种算法并针对实际系统的需要进行针对性的优化设计。请结合 Windows 的线程调度算法, 分析: 一个好的调度算法在 (1) 优先级的动态调整策略, 和 (2) 进程/线程一次执行的时间单元的分配额度等两个方面, 应该需要怎样的优化设计?
2. 某操作系统采用可变分区分配方案管理内存空间; 用户区主存 512kB, 空闲区由空闲分区表管理。分配时采用从低地址部分开始的方案, 并假设初始时全为空。对于下述申请次序: req(300KB), req(100KB), release(300KB), req(150KB), req(30KB), req(40KB), req(60KB), 请问 (需要写出主要过程):
  - (1) 若采用首次最佳适应 (FF), 空闲区中有哪些空块 (大小, 起始地址)?
  - (2) 若采用最佳适应 (BF) 呢?
  - (3) 若申请序列后再加上 req(90KB), 那么使用 (1) (2) 两种不同策略得到的结果如何?

### 《编译原理》(共 40 分)

1. (10 分) 描述由正规式  $b^*(abb^*)^*(a|\epsilon)$  定义的语言, 并画出接受该语言的最简 DFA。



2. (10 分) 证明文法  $E \rightarrow E + id \mid id$  是 SLR(1) 文法。

3. (10 分) 下面是表达式和赋值语句的文法，其中 **and** 的类型是  $bool \times bool \rightarrow bool$ ，**+** 的类型是  $int \times int \rightarrow int$ ，**=** 的类型是  $int \times int \rightarrow bool$ ，**:=** 要求 **id** 和 **E** 的类型都是  $int$  或者都是  $bool$ 。为该文法写一个语法制导定义或翻译方案，它完成类型检查。

$S \rightarrow id := E$

$E \rightarrow E \text{ and } E \mid E + E \mid E = E \mid id$

4. (5 分) 对于下面 C 语言文件 s.c

```
f1(int x)
```

```
{
```

```
    long x;
```

```
    x = 1;
```

```
}
```

```
f2(int x)
```

```
{
```

```
    {
```

```
        long x;
```

```
        x = 1;
```

```
    }
```

```
}
```

某编译器编译时报错如下：

s.c: In function 'f1':

s.c:3: warning: declaration of 'x' shadows a parameter

请回答，对函数 f2 为什么没有类似的警告错误。

5. (5 分) 下面 C 语言程序经非优化编译后，若运行时输入 2，则结果是

area=12.566360, addr=-1073743076

经优化编译后，若运行时输入 2，则结果是

area=12.566360, addr=-1073743068

请解释为什么输出结果有区别。

```
main()
```

```
{
```

```
    float s, pi, r;
```

```
    pi=3.14159;
```

```
    scanf("%f", &r);
```

```
    printf("area=%f,    addr=%d\n", s=pi*r*r, &r);
```

```
}
```

## 《数据结构》(共 70 分)

### 一、选择题 (每小题 1.5 分, 共 9 分)

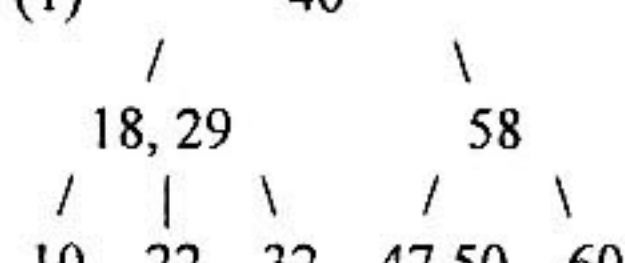
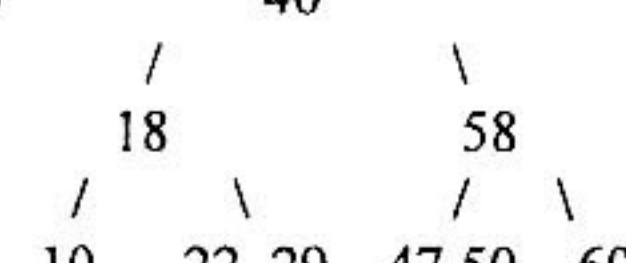
1. D; 2. D; 3. B; 4. C; 5. B; 6. B

### 二、问答题 (共 36 分)

1. (4 分) 略

2. (4 分) 中序遍历有序  $\rightarrow$  二叉树的计数问题:  $b_5 = 42$

3. (6 分) 按行存放:  $2*((i+2)*4+(j-3))$ ; 按列存放:  $2*((j-3)*10+(i+2))$

4. (8 分) (1)  (2) 

5. (10 分) 略

6. (4 分)  $[ \lfloor n/2 \rfloor + 1, n ]$

### 三、算法设计 (共 25 分)

1. (9 分) 算法思路: 从给定节点开始对二叉树进行层次遍历, 遇到的第一个叶子即为所求。

2. (16 分) 算法思路:

1) 对各边的权值排序。

2) 连通分量的合并, 重点: 如何表示各个连通分量集合? ---- 可以用 MFSet(用双亲表示法表示的树来表示一个集合)

3) 时间、空间复杂度的分析: 排序的代价、检查边所关联的两个顶点是否在同一集合中、集合合并的代价。

## 《操作系统》(共 40 分)

### 一、简答题 (每小题 6 分, 共 24 分)

1. 微内核是指尽可能少地提供服务的 OS 内核; 在这种内核下, 大量的操作系统服务可以从用户级服务器上获得。微内核具有很好的灵活性; 一般而言, 微内核仅提供如下 4 类服务: (1) 进程通信服务、(2) 某些内存管理功能、(3) 少量的低层进程管理功能和调度、(4) 低层输入输出功能。



2. 银行家算法是解决死锁的一种策略。在实施资源分配之前，该算法先计算该次分配后的系统状态是否安全，即是否存在一种顺序，使得所有进程都能执行结束；如安全才分配，否则拒绝分配。

此算法需要用到的数据结构包括：(1) 可利用资源向量 Available (2) 最大需求矩阵 Max (3) 分配矩阵 Allocation (4) 需求矩阵 Need。

银行家算法有很好的理论意义，但是在实际系统中却很难使用。这是因为：算法所假设的条件如进程预知申请资源最大数目、系统中进程数目固定等在现实环境中很难成立。

3. LRU 算法即最不经常使用页淘汰算法，其原理为：当需要淘汰某一页时，首先淘汰最后一次访问时间距离当前时间最长的一页。

在软件实现时，只需在页表中给每一页增加一个访问计数器（软件计数器）即可实现；当该页被访问时，访问计数器加 1；发生缺页中断时，淘汰计数最小的页，并将所有计数器置 0。当然其实现代价很高。

4. 设计调度算法时应考虑的因素有：调度算法应与系统设计目标保持一致、注意系统资源均衡使用、保证提交的作业在截止时间内完成、设法缩短作业平均周转时间等。

为评价作业调度算法的性能，经常采用作业平均周转时间、平均带权周转时间等标准。

## 二、分析题（每小题 8 分，共 16 分）

1. Windows 的线程调度算法的调度单位是线程而不是进程，采用严格的抢先式动态优先级调度，依据优先级和分配时间配额来调度。

在优先级控制策略方面，设置了 32 个级别。在下列 5 种情况下，Windows 会提升线程的当前优先级，但永远不会提升实时优先级范围内(16 至 31)的线程优先级：

- I/O 操作完成
- 信号量或事件等待结束
- 前台进程中的线程完成一个等待操作
- 由于窗口活动而唤醒图形用户接口线程
- 线程处于就绪状态超过一定时间，但没能进入运行状态(处理机饥饿)

在时间配额控制策略方面，也根据目前线程的状态（如运行完、重新运行）给予一个新的值。

2. (1) 两个空块：起始地址 280KB，大小 20kB；起始地址 400KB，大小 112kB

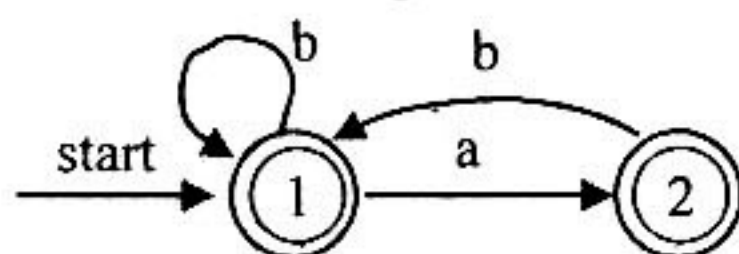
(2) 两个空块：起始地址 210KB，大小 90kB；起始地址 470KB，大小 42kB

(3) 对 (1)，第 2 个空块可以满足分配，第 2 个空块起始地址变为 490 KB；对 (2)，第 1 个空块可以满足分配，并且全部分配出去。

## 《编译原理》（共 40 分）

1. (10 分)

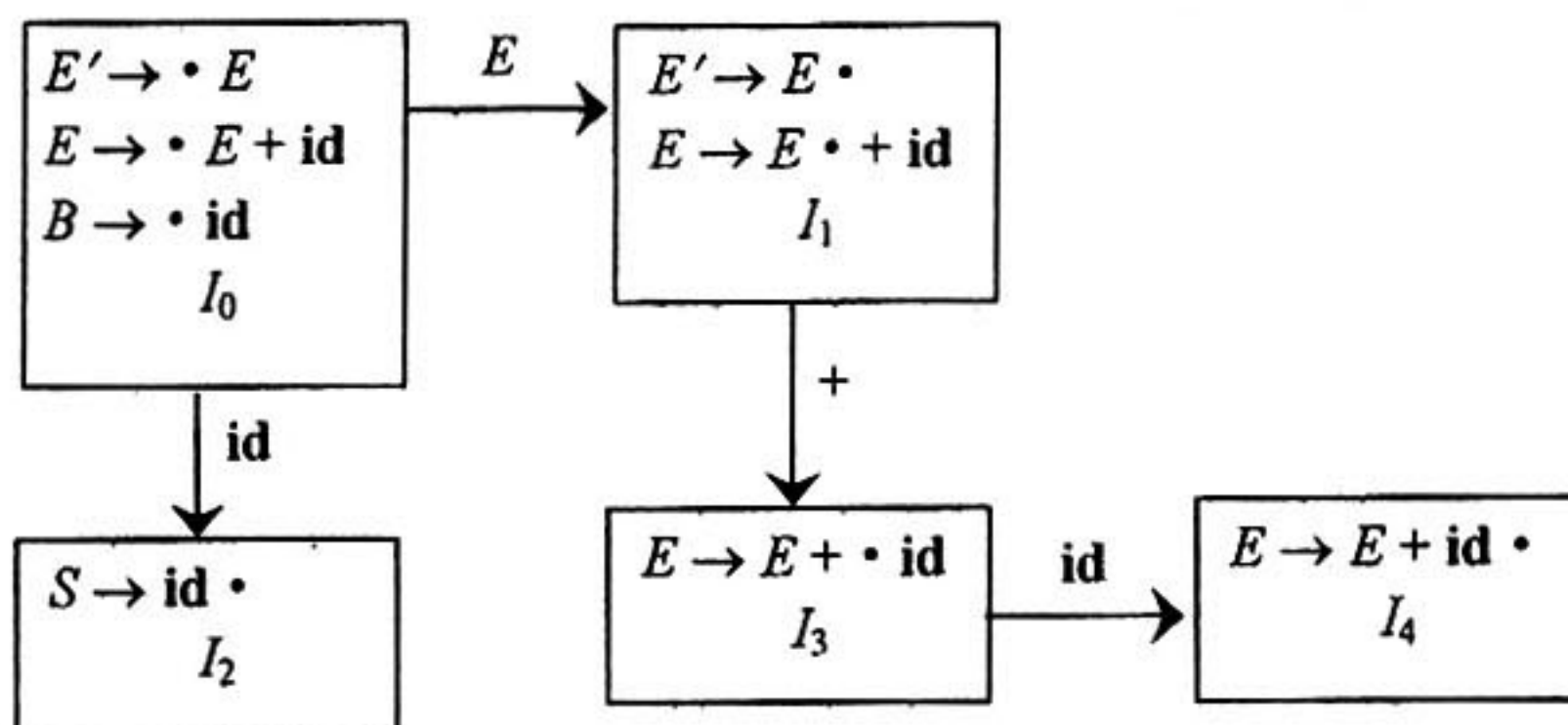
由正规式  $b^*(abb^*)^*(a| \epsilon)$  定义的语言是字母表  $\{a, b\}$  上不含子串  $aa$  的所有串的集合。最简 DFA 如下：



2. (10 分)

先给出接受该文法活前缀的 DFA 如下：





$I_0$  和  $I_3$  都只有移进项目，肯定不会引起冲突； $I_2$  和  $I_4$  都无移进项目并仅含一个归约项目，也肯定不会引起冲突；在  $I_1$  中， $E'$  的后继符号只有  $\$$ ，同第 2 个项目的展望符号 “+” 不一样，因此  $I_1$  也肯定不会引起冲突。由此可以断定该文法是 SLR(1) 的。

### 3. (10 分)

语法制导定义如下。

$S \rightarrow id := E$       {  $S.type := \text{if } (id.type = \text{bool and } E.type = \text{bool}) \text{ or } (id.type = \text{int and } E.type = \text{int}) \text{ then type\_ok else type\_error}$  }  
 $E \rightarrow E_1 \text{ and } E_2$       {  $E.type := \text{if } E_1.type = \text{bool and } E_2.type = \text{bool then bool else type\_error}$  }  
 $E \rightarrow E_1 + E_2$       {  $E.type := \text{if } E_1.type = \text{int and } E_2.type = \text{int then int else type\_error}$  }  
 $E \rightarrow E_1 = E_2$       {  $E.type := \text{if } E_1.type = \text{int and } E_2.type = \text{int then bool else type\_error}$  }  
 $E \rightarrow id$       {  $E.type := \text{lookup}(id.entry)$  }

### 4. (5 分)

对于函数 f1，局部变量 x 声明的作用域是整个函数体，导致在函数体中不可能访问形式参数 x。由于这是一个合法的 C 语言函数，因此编译器给出警告错误。

对于函数 f2，由于局部变量 x 的作用域只是函数体的一部分，不会出现上述问题，因而编译器不报错。

### 5. (5 分)

使用非优化编译时，变量 s, pi, r 在局部数据区都分配 4 个字节的空間。使用优化编译时，由于复写传播， $pi * r * r$  变成  $3.14159 * r * r$ ， $pi = 3.14159$  成为无用赋值而删去，函数中不再有 pi 的引用，因此不必为 pi 分配空间。类似地， $s = 3.14159 * r * r$  也是一个无用赋值（表达式要计算，但赋值是无用的），也不必为 s 分配空间。这样，和非优化情况相比，局部数据区少了 8 个字节，因此 r 的地址向高地址方向移动了 8 个字节。