中 山 大 学

二〇一二年攻读硕士学位研究生入学考试试题

科目代码：**909**

科目名称：专业基础（数据结构）

考试时间： 1 月 8 日 下 午

考 生 须 知

全部答案一律写在答题纸上，答在试题纸上的不计分！请用蓝、黑色墨水笔或圆珠笔作答。答题要写清题号，不必抄题。

一、单项选择题 (每题 2 分，共 40 分)

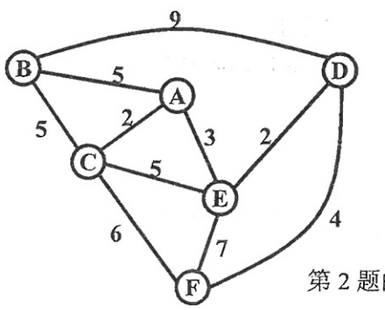
1. 算法复杂度通常是表达算法在最坏情况下所需要的计算量。假设算法 A_1 和 A_2 都可解决规模为 n 的问题 P，且时间复杂度都为 $O(n^2)$。算法 A_1 和 A_2 的时间复杂度之差可能为（ ）
   (A). $O(n^2)$ (B). $O(n)$
   (C). $O(1)$ (D). (A)~(C)都有可能
2. 在数据结构中，按存储结构可把数据结构分为（ ）
   (A). 静态结构和动态结构 (B). 线性结构和非线性结构
   (C). 顺序结构和链式结构 (D). 内部结构和外部结构
3. 在数据结构中，用计算关键字来确定其存储位置的数据结构是（ ）
   (A). Hash 表 (B). 二叉搜索树
   (C). 链式结构 (D). 顺序结构
4. 对链式存储操作的正确描述是（ ）
   (A). 查找操作简单 (B). 遍历操作简单
   (C). 插入和删除操作简单 (D). 定位后的插入和删除操作简单
5. 在下列关于“串”的陈述中，不正确的说明是（ ）
   (A). 串可以用顺序存储 (B). 串是由字母和数字构成
   (C). 串可以用链式存储(分块存储) (D). 在 C 语言中，串的最后隐含一个字符‘\0’
6. 假设用静态数组 entry[SSize]来存储堆栈信息，栈顶下标 Top 的初值为-1。栈满的条件是（ ）
   (A). Top==0 (B). Top==SSize-1 (C). Top==SSize (D). Top==99
7. 关于队列的不正确描述是（ ）
   (A). FILO (B). FIFO
   (C). 可以获取队列头元素中的信息 (D). 不可修改队列头元素中的信息
8. 假设循环队列的长度为 QSize。当队列未满时，向队列中添加一个数据后，其队尾下标 Rear 的变化为（ ）
   (A). Rear= Rear + 1 (B). Rear= Rear++ % QSize0
   (C). Rear= (Rear + 1) % QSize (D). Rear = Rear % Qsize + 1

9. 假设 Head 是指向“不带头结点的单向循环链”指针，判断该链表为空的条件是（ ）
   (A). Head->next == NULL (B). Head->next == Head
   (C). Head.next == NULL (D). Head == NULL
10. 设 A[n][n]为一个对称矩阵，数组下标从[0][0]开始。为了节省存储，将其上三角部分按行存放在一维数组 B[0..m-1]，m=n(n+1)/2，对上三角部分中任一元素 $A_{ij}(i \le j)$，它在一维数组 B 的下标 k 值是（ ）
    (A). (2n-i+1)i/2+(j-i) (B). (2n-i)i/2+(j-i) (C). (2n-i)i/2+(j-i+1) (D). (n-i+1)i/2+(j-i)
11. 假设用二叉链来表达二叉树。若二叉树中有 n 个结点，那么，它共有（ ）个空指针。
    (A). n+2 (B). n+1 (C). n (D). n-1
12. 若一棵二叉树的先序和中序序列分别是 abdefc 和 dbefac，则其后序序列是（ ）
    (A). dfecba (B). defbca (C). dfebca (D). abcdef
13. 用一维数组来存储满二叉树，若数组下标从 0 开始，则元素下标为 k(k>0)的父结点下标是（）
    (A). 2k+1 (B). 2k+2 (C). $\lfloor k/2 \rfloor$ (D). $\lceil k/2 \rceil$
14. 在树高为 $O(h)$，且有 n 个结点的二叉搜索树中搜索关键字。其搜索效率为（ ）。
    (A). $O(1)$ (B). $O(h)$ (C). $O(n)$ (D). $O(nh)$
15. 对 n 个结点和 e 条边的无向图，用邻接矩阵存储它所用的内存空间为（ ）
    (A). $O(en)$ (B). $O(e^2)$ (C). $O(n^2)$ (D). $O(en^2)$
16. 用邻接矩阵存储有 n 个顶点和 e 条边的有向图，则确定某个顶点出度的时间复杂度是（ ）
    (A). $O(n)$ (B). $O(e)$ (C). $O(n+e)$ (D). $O(ne)$
17. 下列排序算法中，时间复杂度为 $O(n\log n)$的排序方法是（ ）
    (A). 选择排序 (B). 桶(基数)排序 (C). 插入排序 (D). 堆排序
18. 基于比较的排序算法对 n 个数进行排序的比较次数至少需要（ ）
    (A). $O(n^2)$ (B). $O(n\log n)$ (C). $O(n)$ (D). $O(\log n)$
19. 在用桶(基数)排序算法对待排数据按“十六进制数”进行排序时，需要桶的个数是（ ）
    (A). 8 (B). 10 (C). 16 (D). 20
20. 在下列算法中，求图中一个结点到其它结点的最短路径算法是（ ）
    (A). Dijkstra 算法 (B). KMP 算法 (C). Kruskal 算法 (D). DFS 算法

二、解答题 (每题 10 分，共 50 分)

1. 已知一个无向图的顶点集为{ a, b, c, d, e, f }，其邻接矩阵如下所示(0-无边,1-有边)。

$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \\ f \end{array}\begin{array}{c} \begin{array}{cccccc} a & b & c & d & e & f \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \end{array}$$



第 2 题的用图

(1). 画出该图的图形；

(2). 根据邻接矩阵从顶点 *a* 出发进行广度优先遍历(同一个结点的邻接结点按结点序号大小为序)，画出相应的广度优先遍历树。

2．简单描述生成图最小生成树 Prim 算法的基本思想，并按步骤从结点 E 开始列出上图（见上页）最小生成树的求解过程。

3．简单叙述合并排序算法(MergeSort)的基本思想，并按其排序步骤列出下列数据的排序过程。
待排序的数值序列：40　15　33　67　34　78　93

4．已知有下列 13 个元素的散列表：

0	1	2	3	4	5	6	7	8	9	10	11	12
63		43			20			50	37	25		

其散列函数为 $h(key) = (key + 11) \% m(m=13)$，处理冲突的方法为平方探测再散列法，探查序列为：$h_i = (h(key) + d_i) \% m$，　$d_i = 1,-1^2, 2^2, -2^2, \ldots, i^2, -i^2, \ldots$

问：在表中对关键字 10 和 63 进行查找时，所需进行的比较次数为多少？依次写出每次计算公式和值。

5．假设在通信中，字符 *a, b, c, d, e, f, g* 出现的频率如下：

a: 7%　*b*: 25%　*c*:9%　*d*: 15%　*e*: 13%　*f*: 11%　*g*: 20%

(1) 根据 Huffman 算法(赫夫曼算法)画出其赫夫曼树；

(2) 给出每个字母所对应的赫夫曼编码，规定：结点左分支边上标 1，右分支边上标 0；

(3) 计算其加权路径的长度 WPL。

三、阅读理解题，按空白编号填写相应的 C 语言语句，以实现函数功能。(每空 2 分，每题 10 分，共 30 分)

1．假设用链式结构来实现堆栈，用给定的有关数据结构和变量定义完成相关操作的编写。

```
typedef struct _StackNode {
   int   Data;
   struct _StackNode *next;
} StackNode;
StackNode Stack;          /* 定义一个堆栈变量 Stack */
```

(1) 初始化堆栈 ST

```
void InitStack(StackNode *ST)
{
   ST->next = NULL;
}
```

(2) 入栈操作 Push(ST, data)：若无法申请到空间，返回 0，否则，把数据 data 压入到堆栈 ST，并返回 1


```
int Push(StackNode *ST, int data)
{
   StackNode *Pt;
   Pt = (StackNode *) calloc(1, sizeof(StackNode)); /* 申请一个堆栈结点空间 */
   if (Pt == NULL) return 0;
   Pt->Data = data;
   ____(1)____;
   ____(2)____;
   return 1;
}
```

(3) 出栈操作 Pop(ST, data)：若堆栈 ST 为空，返回 0，否则，在弹出栈顶之前，把栈顶数据存入参数 data 之中，并返回 1

```
int Pop(StackNode *ST, int *data)
{
   StackNode *Pt;
   if (ST->next == NULL ) return 0;
   *data = ____(3)____;
   ____(4)____;
   ____(5)____;
   free(Pt) ;
   return 1;
}
```

2．假设用不带头结点的单向链表存储一元多项式(按“指数”从大到小的顺序)。其链表结点的结构定义如下：

```
typedef struct _PNode {
   int   Coef;     /* 系数 */
   int   Expn;     /* 指数(规定：指数≥0) */
   struct _PNode *next;
} PNode;
```

函数 Derivative(PNode *P1)：求参数 P1 所指向多项式的导数(P_1')，且返回导数多项式的首地址。

```
PNode *Derivative(PNode *P1)
{
   PNode *Pt1, *Pt2, *P2, *Tail;
   P2 = NULL;
   for (Pt1 = P1; Pt1 != NULL; Pt1 = Pt1->next) {
      if (____(1)____) {
         /* 申请一个多项式结点空间，且不考虑申请内存空间失败的情况 */
```

```
            Pt2 = (PNode *) calloc(1, sizeof(PNode));
            Pt2->Coef = _____(2)_____;
            Pt2->Expn = _____(3)_____;
            if (_____(4)_____) P2 = Pt2;
            else Tail->next = Pt2;
            _____(5)_____;
        }
    }
    return P2;
}
```

3．假设二叉树 $T=<T_L, root, T_R>$中叶结点数的定义如下:

$$Leaf(T)=\begin{cases}0 & T\text{ 是空树}\\ 1 & \text{根结点 }root\text{ 是叶结点}\\ Leaf(T_L)+Leaf(T_R) & \text{其他}\end{cases}$$

已知二叉树的结点定义如下：

```
typedef struct _TreeNode {
    int     Key;
    struct _TreeNode *LChild, *RChild;
} TreeNode;
```

(1) 函数 Leaf(root)是求以结点 root 为根的二叉树中的叶结点数。

```
int Leaf(TreeNode *root)
{
    if ( ___(1)___ ) return 0;
    if ( ___(2)___ ) return 1;
    return ( ____(3)____ );
}
```

(2) 函数 FreeTree(root)是释放以结点 root 为根的二叉树所占的内存空间。

```
void FreeTree(TreeNode *root)
{
    if (root != NULL) {
        FreeTree(root->LChild);
        _____(4)_____;
        _____(5)_____;
    }
}
```

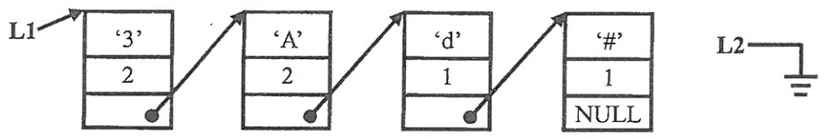

四、算法设计题 (每题 15 分，共 30 分)

用 C 语言或类 C 语言实现下面函数的功能。

1．假设用链表依次存储字符串中各字符及其出现的次数(不区分字母的大小写)。链表的结点定义如下：

```
typedef struct _Node {
    char    Char;       /* 字符 */
    int     Num;        /* 字符 Char 的出现次数 */
    struct _Node *next;
} Node;
```

例如：S1="3Ada3#"，S2=""，字符串 S1 和 S2 统计后所对应的链表 L1 和 L2 如下图所示。



(1) Node *Statistics(char *Str)，其功能是生成字符串 Str 中所有字符及其出现次数的统计链表，并返回该链表的头指针(不考虑申请内存空间失败的情况) (9 分)

(2) double Probability(Node *List, char Char)，其功能是返回字符 Char 在链表中出现的概率 (6 分)

例如有下列语句：

```
Node *L1 = Statistics("3Ada3#");
double p1 = Probability(L1, 'a');     // p1 的值为 0.333333
double p2 = Probability(L1, '1');     // p2 的值为 0
```

2．已知二叉搜索树(二叉排序树)的结点定义如下：

```
typedef struct _TreeNode {
    int     Key;
    struct _TreeNode *LChild, *RChild;   /* 左子树的关键字比根小，右子树的关键字比根大*/
} TreeNode;
```

编写函数 Find(TreeNode *root, int key)，其功能是在以结点 root 为根的二叉搜索树中找"比参数 key 小的最大值"。若找不到，则返回 NULL，否则，返回该结点地址。