全国计算机技术与软件专业技术资格（水平）考试

2004 年上半年 软件设计师 下午试卷

（考试时间 14:00～16:30 共 150 分钟）

请按下述要求正确填写答题纸

1. 在答题纸的指定位置填写你所在的省、自治区、直辖市、计划单列市的名称。
2. 在答题纸的指定位置填写准考证号、出生年月日和姓名。
3. 答题纸上除填写上述内容外只能写解答。
4. 本试卷共 6 道题，试题一至试题四是必答题，试题五至试题六选答 1 题。每题 15 分，满分 75 分。
5. 解答时字迹务必清楚，字迹不清时，将不评分 。
6. 仿照下面例题，将解答写在答题 纸的对应栏内。

例题

2004 年上半年全国计算机技术与软件专业技术资格（水平）考试日期是（1）月（2）日。

因为正确的解答是“5 月 23 日”，故在答题纸的对应栏内写上“5”和“23”（参看下表）。

例题	解答栏
(1)	5
(2)	23

试题一至试题四是必答题

试题一

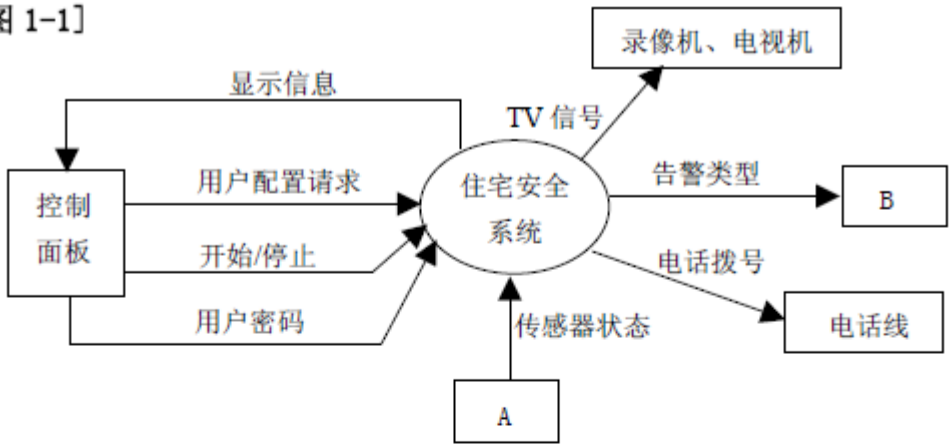
阅读下列说明和数据流图，回答问题 1 至问题 4，将解答填入答题纸的对应栏内。

[说明]

某基于微处理器的住宅安全系统，使用传感器（如红外探头、摄像头等）来检测各种意外情况，如非法进入、火警、水灾等。

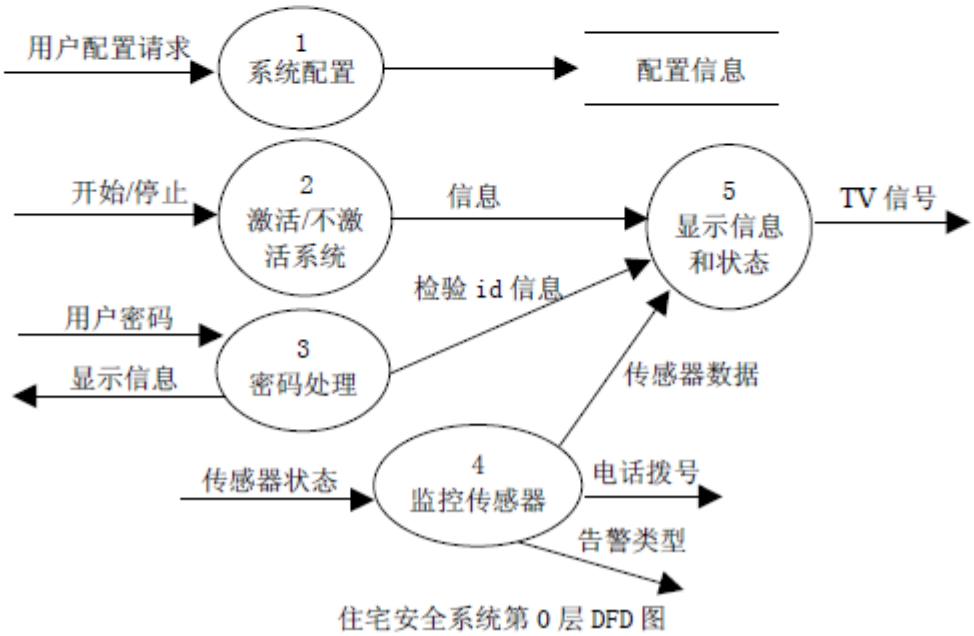
房主可以在安装该系统时配置安全监控设备（如传感器、显示器、报警器等），也可以在系统运行时修改配置，通过录像机和电视机监控与系统连接的所有传感器，并通过控制面板上的键盘与系统进行信息交互。在安装过程中，系统给每个传感器赋予一个编号（即 id）和类型，并设置房主密码以启动和关闭系统，设置传感器事件发生时应自动拨出的电话号码。当系统检测到一个传感器事件时，就激活警报，拨出预置的电话号码，并报告关于位置和检测到的事件的性质等信息。

[数据流图 1-1]



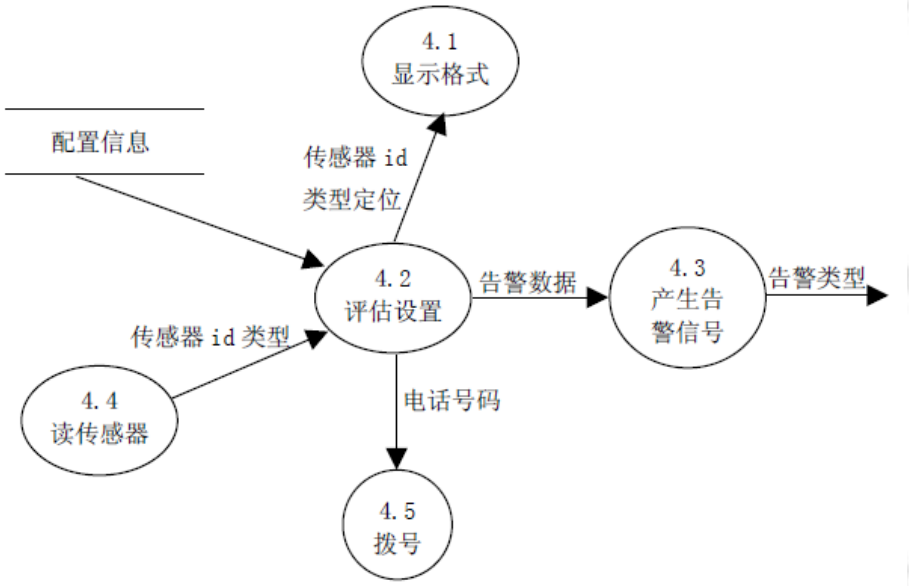
住宅安全系统顶层图

[数据流图 1-2]



住宅安全系统第 0 层 DFD 图

[数据流图 1-3]



加工 4 的细化图

[问题 1]

数据流图 1-1（住宅安全系统顶层图）中的 A 和 B 分别是什么？

[问题 2]

数据流图 1-2（住宅安全系统第 0 层 DFD 图）中的数据存储“配置信息”会影响图中的哪些加工？

[问题 3]

将数据流图 1-3（加工 4 的细化图）中的数据流补充完整，并指明加工名称、数据流的方向（输入/输出）和数据流名称。

[问题 4]

请说明逻辑数据流图（Logical Data Flow Diagram）和物理数据流图（Physical Data Flow Diagram）之间的主要差别。

试题二

阅读下列说明和算法，回答问题 1 和问题 2，将解答填入答题纸的对应栏内。

[说明]

算法 2-1 是用来检查文本文件中的圆括号是否匹配。若文件中存在圆括号没有对应的左括号或者右括号，则给出相应的提示信息，如下所示：

文件	提示信息
(1+2)	
abc)	缺少对应左括号：第 2 行，第 4 列
((def)gx))	缺少对应左括号：第 3 行，第 10 列
(((h)	
ij)(k	
(lml)	缺少对应右括号：第 5 行，第 4 列； 第 4 行，第 1 列

在算法 2-1 中，stack 为一整数栈。算法中各函数的说明如下表所示：

函数名	函数功能
push(int i)	将整数 i 压入栈 stack 中。
pop()	stack 的栈顶元素出栈。
empty()	判断 stack 栈是否为空，若为空，函数返回 1，否则函数返回 0。
nextch()	读取文本文件中的下一个字符，并返回该字符的 ASCII 值，将字符所在的行号以及字符在行中的位置分别存储到变量 row 和 col 中，若遇到文件结束符，则将变量 EOF 置为 true;
kind(char ch)	判断字符ch是左括号还是右括号，若是左括号，函数返回1，若是右括号，函数返回2，若两者都不是，函数返回0。

[算法 2-1]

```
将栈 stack 置空，置 EOF 为 false
ch ← nextch();
while(not EOF)
    k ← kind(ch);
    if(k == (1) )
    push( (2) ); push( (3) );
    elseif( k == (4) )
    if(not empty())
    pop(); pop();
    else
    显示错误信息（缺少对应左括号或右括号）;
    显示行号 row; 显示列号 col;
    endif
    endif
    ch ← nextch();
```

```
endwhile
if (not empty())
    显示错误信息（缺少对应左括号或右括号）;
    while(not empty())
    row ← pop(); col ← pop();
    显示行号 row; 显示列号 col;
    endwhile
endif
```

为了识别更多种类的括号，对算法 2-1 加以改进后得到算法 2-2。算法 2-2 能够识别圆括号，方括号和花括号(不同类型的括号不能互相匹配)。改进后，函数 kind(char ch)的参数及其对应的返回值如下表所示：

ch	(	)	{	}	[	]	其它
返回值	1	2	3	4	5	6	0

[算法 2-2]

```
将栈 stack 置空，置 EOF 为 false
ch ← nextch();
while(not EOF)
    k ← kind(ch);
    if(k > 0)
    if( 判断条件 1 )
    push( (5) );push( (6) );push( (7) );
    elseif( 判断条件 2 and 判断条件 3 )
    pop(); pop(); pop();
    else
    显示错误信息（缺少对应左括号或右括号）;
    显示行号 row; 显示列号 col;
    endif
    endif
    ch ← nextch();
```

```
endwhile
if(not empty())
    显示错误信息（缺少对应左括号或右括号）;
    while(not empty())
    pop(); row ← pop(); col ← pop();
    显示行号 row; 显示列号 col;
    endwhile
endif
```

[问题 1]

请将[算法 2-1]和[算法 2-2]中（1）至（7）处补充完整。

[问题 2]

请从下面的选项中选择相应的判断逻辑填补[算法 2-2]中的“判断条件 1”至“判断条件 3”。注意，若“判断条件 2”的逻辑判断结果为假，就无需对“判断条件 3”进行判断。

(a)字符是括号 (b)字符是左括号 (c)字符是右括号 (d)栈空 (e)栈不空

(f)栈顶元素表示的是与当前字符匹配的左括号

(g)栈顶元素表示的是与当前字符匹配的右括号

试题三

阅读下列说明以及图 3-1 和图 3-2，回答问题 1、问题 2 和问题 3，将解答填入答题纸的对应栏内。

[说明]

某电话公司决定开发一个管理所有客户信息的交互式网络系统。系统的功能如下：

1.浏览客户信息：任何使用 Internet 的网络用户都可以浏览电话公司所有的客户信息(包括姓名、住址、电话号码等)。

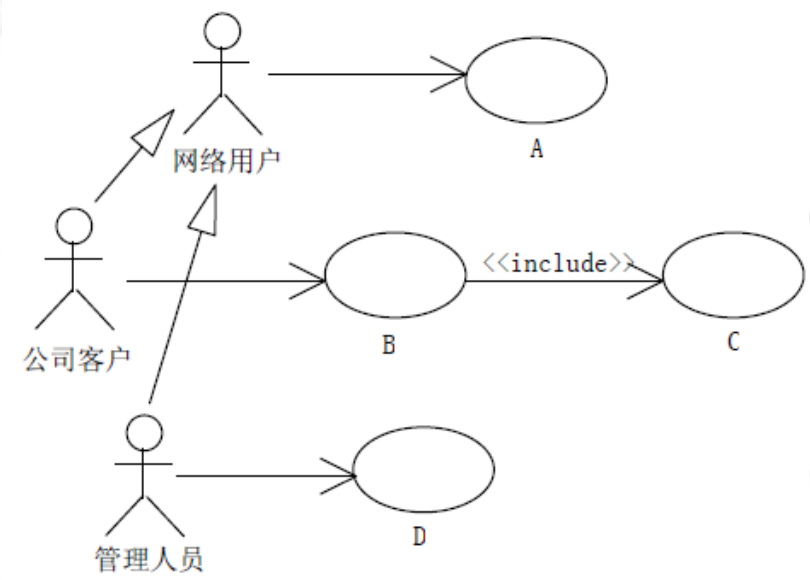
2.登录：电话公司授予每个客户一个帐号。拥有授权帐号的客户，可以使用系统提供的页面设置个人密码，并使用该帐号和密码向系统注册。

3.修改个人信息：客户向系统注册后，可以发送电子邮件或者使用系统提供的页面，对个人信息进行修改。

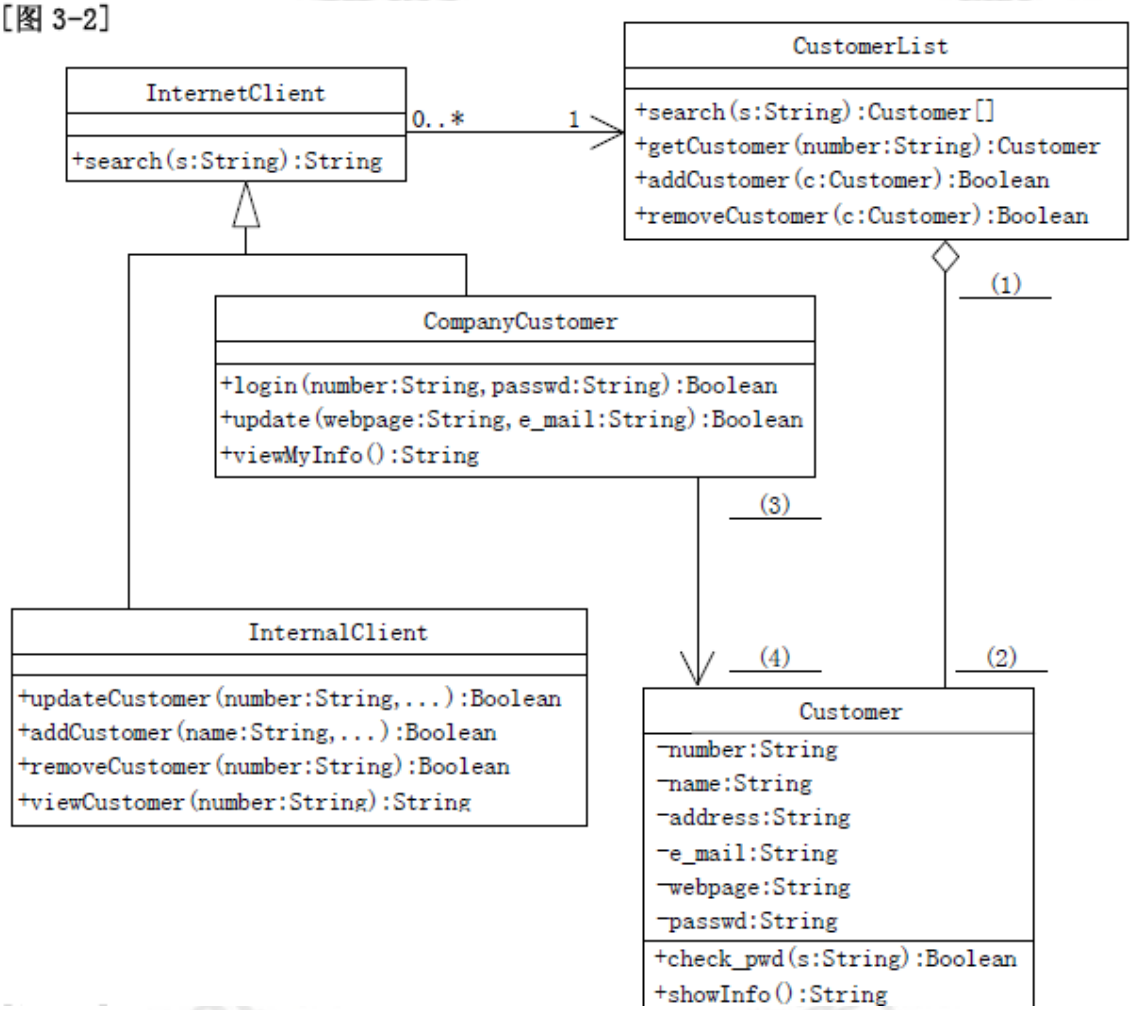
4.删除客户信息：只有公司的管理人员才能删除不再接受公司服务的客户的信息。系统采用面向对象方法进行开发，在开发过程中认定出的类如下表所示：

编号	类名	描述
1	InternetClient	网络用户
2	CustomerList	客户信息表，记录公司所有客户的信息
3	Customer	客户信息，记录单个客户的信息
4	CompanyCustomer	公司客户
5	InternalClient	公司的管理人员

[图 3-1]



[图 3-2]



[问题 1]

在需求分析阶段，采用 UML 的用例图（use case diagram）描述系统功能需求，如图 3-1 所示。请指出图中的 A、B、C 和 D 分别是哪个用例？

[问题 2]

在 UML 中，重复度（Multiplicity）定义了某个类的一个实例可以与另一个类的多少个实例相关联。通常把它写成一个表示取值范围的表达式或者一个具体的值。例如图 3-2 中的类 InternetClient 和 CustomerList，InternetClient 端的“0..*”表示：一个 CustomerList 的实例可以与 0 个或多个 InternetClient 的实例相关联；CustomerList 端的“1”表示：一个 InternetClient 的实例只能与一个 CustomerList 的实例相关。

请指出图 3-2 中(1)到(4)处的重复度分别为多少？

[问题 3]

类通常不会单独存在，因此当对系统建模时，不仅要识别出类，还必须对类之间的相互关系建模。在面向对象建模中，提供了四种关系：依赖（dependency）、概括（generalization）、

关联（association）和聚集（aggregation）。请分别说明这四种关系的含义，并说明关联和聚集之间的主要区别。

试题四

在 COMET 型计算机上可以使用试卷上所附的 CASL 汇编语言，阅读程序说明和 CASL 程序，把应填入 （n） 处的字句写在答卷的对应栏内。

[程序 4 说明]

本程序将 16 位无符号二进制数转换为 5 位十进制数，转换结果用 ASCII 码表示，并从高位至低位依次存放在首地址为 BTASC 的连续 5 个内存单元中。待转换的 16 位无符号二进制数存放在 DATA 内存单元中。

[程序 4]

```
                START
    PROGBC      LD          GR0, DATA
                LEA         GR1, 0
                LEA         GR3, 48
    LOOP1       CPL         GR0, WDT, GR1
                JPZ         LOOP2
                ST          GR3, BTASC, GR1
                LEA         GR1, 1, GR1
                LEA         GR2, -4, GR1
                JNZ         LOOP1
                （1）
LOOP2           LEA         GR2, 48
LOOP3           CPL         GR0, WDT, GR1
                JMI         NEXT
                （2）
                LEA         GR2, 1, GR2
                JMP         LOOP3
NEXT            （3）
                LEA         GR1, 1, GR1
```

```
                LEA         GR2, -4, GR1
                JNZ         LOOP2
LAST            （4）  ;     处理个位数
                （5）
                EXIT
C48             DC          48
WDT             DC          10000
                DC          1000
                DC          100
                DC          10
BTASC           DS          5
DATA            DC          #FA59H
                END
```

从下列的 2 道试题（试题五至试题六）中任选 1 道解答。
如果解答的试题数超过 1 道，则题号小的 1 道解答有效。

试题五

阅读下列函数说明和 C 函数，将应填入 （n） 处的字句写在答题纸的对应栏内。

[函数 5 说明]

函数 DeleteNode(Bitree *r,int e)的功能是：在树根结点指针为 r 的二叉查找（排序）树上删除键值为 e 的结点，若删除成功，则函数返回 0，否则函数返回-1。二叉查找树结点的类型定义为：

```
typedef struct Tnode{
    int data; /*结点的键值*/
    struct Tnode *Lchild,*Rchild; /*指向左、右子树的指针*/
}*Bitree;
```

在二叉查找树上删除一个结点时，要考虑三种情况：

①若待删除的结点 p 是叶子结点，则直接删除该结点；

②若待删除的结点 p 只有一个子结点，则将这个子结点与待删除结点的父结点直接连接，

然后删除结点 p；

③若待删除的结点 p 有两个子结点，则在其左子树上，用中序遍历寻找关键值最大的结点 s，用结点 s 的值代替结点 p 的值，然后删除结点 s，结点 s 必属于上述①、②情况之一。

[函数 5]

```
int DeleteNode(Bitree *r, int e) {
    Bitree p = *r, pp, s, c;
    while ( （1） ){ /*从树根结点出发查找键值为 e 的结点*/
    pp = p;
    if (e < p->data) p = p->Lchild;
    else p = p->Rchild;
    }
    if (!p) return -1; /*查找失败*/
    if (p->Lchild && p->Rchild) { /*处理情况③*/
    s = （2） ; pp = p;
    while ( （3） ) { pp = s; s = s->Rchild; }
    p->data = s->data; p = s;
    }
    /*处理情况①、②*/
    if ( （4） ) c = p->Lchild;
    else c = p->Rchild;
    if (p == *r) *r = c;
    else if ( （5） ) pp->Lchild = c;
    else pp->Rchild = c;
    free(p);
    return 0;
}
```

试题六

阅读下列说明和 C++程序，将应填入 （n） 处的字句写在答题纸的对应栏内。

[程序 6 说明]

C++语言本身不提供对数组下标越界的判断。为了解决这一问题，在程序 6 中定义了相应的类模板，使得对于任意类型的二维数组，可以在访问数组元素的同时，对行下标和列下标进行越界判断，并给出相应的提示信息。

[程序 6]

```
#include <iostream.h>
template <class T> class Array;
template <class T> class ArrayBody {
    friend  (1)  ;
    T* tpBody;
    int iRows, iColumns, iCurrentRow;
    ArrayBody(int iRsz, int iCsz) {
    tpBody =  (2)  ;
    iRows = iRsz; iColumns = iCsz; iCurrentRow = -1;
    }
    public:
    T& operator[](int j) {
    bool row_error, column_error;
    row_error = column_error = false;
    try {
    if (iCurrentRow < 0 || iCurrentRow >= iRows)
    row_error = true;
    if (j < 0 || j >= iColumns)
    column_error = true;
    if (row_error == true || column_error == true)
     (3)  ;
    }
    catch(char) {
    if (row_error == true)
    cerr << "行下标越界[" << iCurrentRow << "] ";
    if (column_error == true)
```

```
    cerr << "列下标越界[" << j << "]";
    cout << "\n";
    }
    return tpBody[iCurrentRow * iColumns + j];
    }
    ~ArrayBody() { delete[] tpBody; }
    };
    template <class T> class Array {
    ArrayBody<T> tBody;
    public:
    ArrayBody<T> & operator[](int i) {
     (4)  ;
    return tBody;
    }
    Array(int iRsz, int iCsz) :  (5)  { }
    };
    void main()
    {
    Array<int> a1(10, 20);
    Array<double> a2(3, 5);
    int b1;
    double b2;
    b1 = a1[-5][10]; // 有越界提示：行下标越界[-5]
    b1 = a1[10][15]; // 有越界提示：行下标越界[10]
    b1 = a1[1][4]; // 没有越界提示
    b2 = a2[2][6]; // 有越界提示：列下标越界[6]
    b2 = a2[10][20]; // 有越界提示：行下标越界[10] 列下标越界[20]
    b2 = a2[1][4]; // 没有越界提示
}
```

（本试题的参考答案请在软考网下载。网址是 http://www.RuanKao.net）