(A)中级软件设计师下午试题模拟66

试题一

阅读下列说明和数据流图，回答问题1至问题3。

[说明]

下面给出的是某房产管理系统的一套分层数据流图。其功能描述如下：

1系统随时根据住房送来的入住单更新住户基本信息文件；

2每月初系统根据物业管理委员会提供的月附加费(例如清洁费、保安费、大楼管理费等)表和房租调整表，计算每家住户的月租费(包括月附加费)，向住户发出交费通知单。住户交费时，系统输入交费凭证，核对后输出收据给住户；

3系统定期向物业管理委员会提供住房分配表和交费情况表；

4住户因分户或换房，在更新住户基本信息文件的同时，系统应立即对这些住户做月租费计算，以了结分户或换房前的房租。

以下是经分析得到的数据流图及部分数据字典，有些地方有待填充，假定顶层数据流图是正确的。图1是顶层数据流图，图2是第0层数据流图，图3是第1层数据流图，其中A是加工1的细化图，B是加工2的细化图。假定题中提供的顶层图是正确的，请回答下列问题。

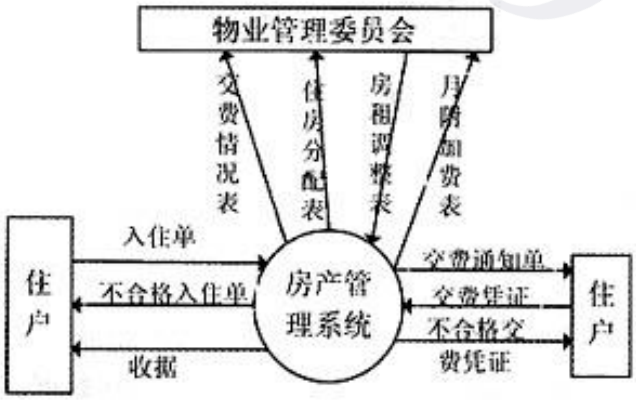


图1

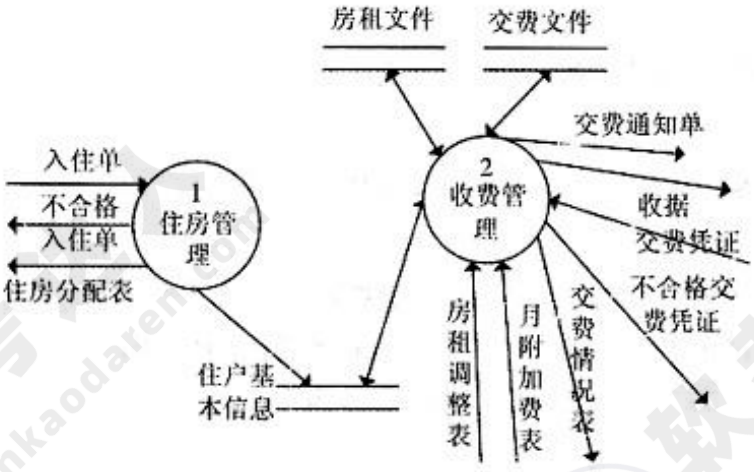


图2

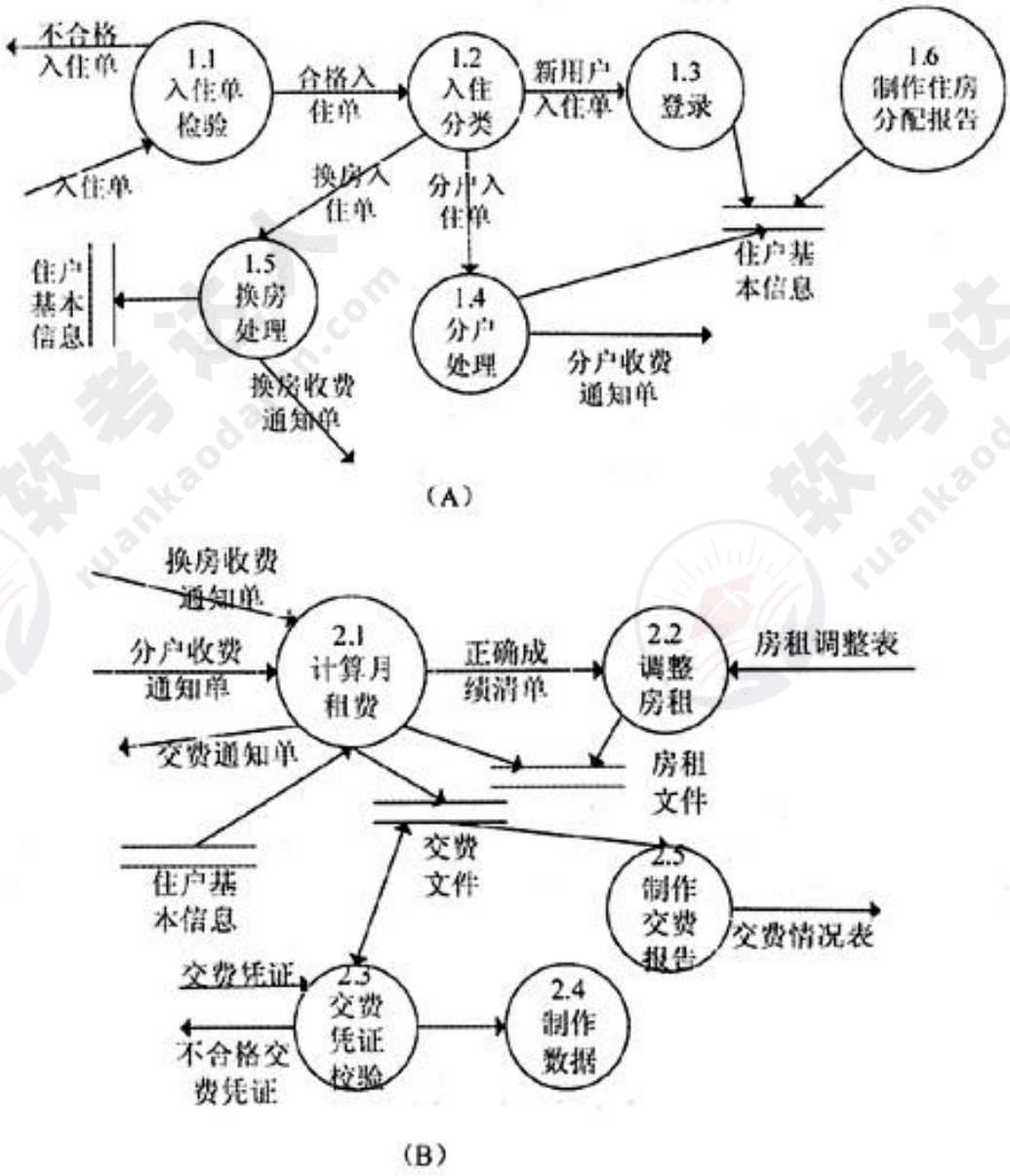


图3

1、指出图2中的哪些文件可不必画出。

2、指出在哪些图中遗漏了哪些数据流。回答时请用如下形式之一：

1)××图中遗漏了××加工(或文件)流向××加工(或文件)的××数据流；

2)××图中加工××遗漏了输入(或输出)数据流××。

3、指出图3的B中加工2.3能检查出哪些不合格交费凭证。

试题二

阅读下列说明和E-R图，回答问题1至问题3。

[说明]

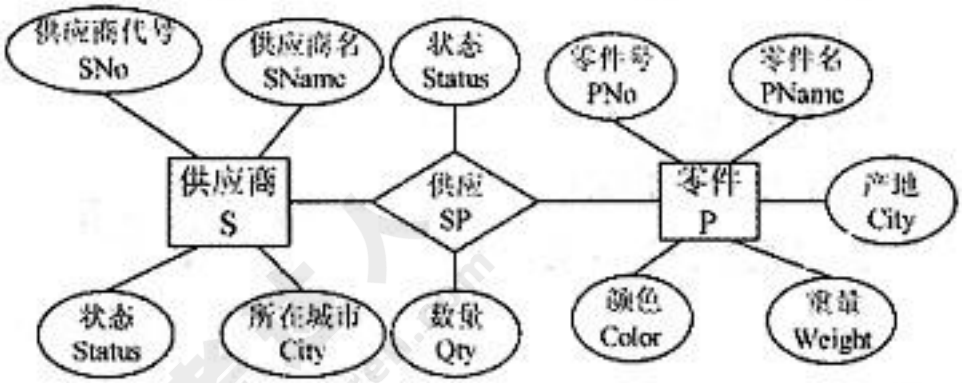
建立一个供应商零件数据库，数据库要满足如下要求：

4供应商代码不能为空，且是值唯一的，供应商的名也是唯一的。

5零件号不能为空，且值是唯一的，零件号不能为空。

6一个供应商可以供应多个零件，而一个零件可以由多个供应商供应。

图是该系统的E-R图。



4、根据E-R图中给出的词汇，按照"有关模式名(属性，属性，...)"的格式，将此E-R图转换为3个关系模式，指出每个关系模式中的主码和外码，其中模式名根据需要取实体名或联系名。

5、创建S表时，SNo使用CHAR(5)并且唯一，SName使用CHlAR(30)，Status使用CHAR(8)，City使用CHAR(20)。请在下列用于创建表S的SQL语句空缺处填入正确的内容。

```
CREATE TABLE S(SNo CHAR(5),
SName CHAR(30),
Status CHAR(8),
City CHAR(20),
_______;
```

6、假定SP表存储供应情况，如下的SQL语句是用于查询"产地为'Beijing'、零件号为'P101'的零件的所供应的总数(包括所有供应商)"的不完整语句，请在空缺处填入正确的内容。

```
SELECT SUM(Qty)
FROM SP
WHERE PN0="p101"
_______ PNo _______
(SELECT PNo
FROM _______
WHERE City = "Beijing")
_______ PNo;
```

试题三

阅读以下说明和程序流程图，将应填入画横线处的字句写在对应栏内。

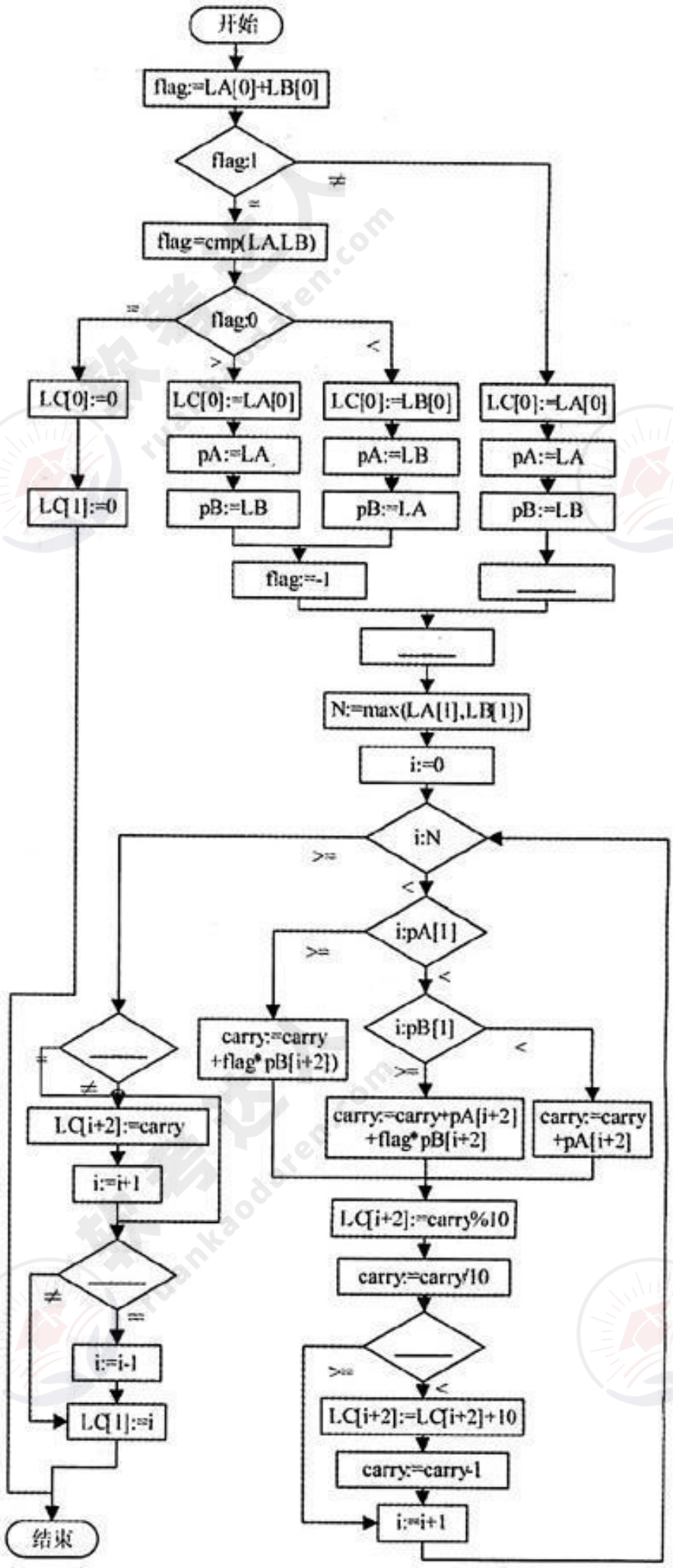
7、[说明]

假定用一个整型数组表示一个长整数，数组的每个元素存储长整数的一位数字，则实际的长整数m表示为：

$m=a[k]\times10^{k-2}+a[k-1]\times10^{k-3}+...+a[3]\times10+a[2]$

其中a[1]保存该长整数的位数，a[0]保存该长整数的符号：0表示正数、1表示负数。注：数组下标从0开始。

流程图用于计算长整数的加(减)法。运算时先决定符号，再进行绝对值运算。对于绝对值相减情况，总是绝对值较大的减去绝对值较小的，以避免出现不够减情况。注，此处不考虑溢出情况，即数组足够大。这样在程序中引进两个指针pA和pB，分别指向绝对值较大者和较小者。而对绝对值相加情况，让pA指向LA，pB指向LB，不区分绝对值大小。pA±pB可用通式pA+flag*pB来计算，flag为+1时即对应pA+pB，flag为-1时即对应pA-pB。需特别注意的是，对于相减，不够减时要进行借位，而当最高位借位后正好为0时，结果的总位数应减1；对于加法，有最高进位时，结果的总位数应加1。



流程图中涉及的函数说明如下：

(1)cmp(int *LA，int *LB.函数，用于比较长整数LA与LB的绝对值大小，若LA绝对值大于LB绝对值则返回正值，LA绝对值小于LB绝对值返回负值，相等则返回0。

(2)max(int A，int B.函数，用于返回整数A与B中较大数。

另外，对流程图中的写法进行约定：(1)“:=”表示赋值，如“flag:=LA[0]+LB[0]”表示将“LA[0]+LB[0]”的结果赋给flag，相当于C中的赋值语句：“flag=LA[0]+LB[0];”；(2)“:”表示比较运算，如“flag:1”表示flag与1比较。

试题四

阅读下列函数说明和*C*代码，将应填入画横线处的字句写在对应栏内。

8、[说明]

Huffman树又称最优二叉树，是一类带权路径长度最短的树，在编码中应用比较广泛。

构造最优二叉树的Huffman算法如下：

①根据给定的n各权值$\{w_1,w_2,\ldots,w_n\}$构成n棵二叉树的集合$F=\{T_1,T_2,\ldots,T_n\}$，其中每棵树T_i中只有一个带权为w_i的根节点，其左右子树均空。

②在F中选取两棵根节点的权值较小的树作为左右子树，构造一棵新的二叉树，置新构造二叉树的根节点的权值为其左右子树根节点的权值之和。

③从F中删除这两棵树，同时将新得到的二叉树加入到F中。

重复②③，直到F中只剩一棵树为止。

函数中使用的预定义符号如下：

```
#define INT_MAX 10000
#define ENCODING LENGTH 1000
typedef enum(none，left_chiid，right chiid} Which;
/*标记是左孩子还是右孩子*/
typedef char Elemtype;
typedef struct TNode{//Huffman树节点
Elemtype letter;
int weight;    //权值
int parent;   //父节点
Which sigh;
char *code;    //节点对应编码
}HTNode,*HuffmanTree;
int n;
char coding[50];//储存代码
[函数]
void Select(HuffmanTree HT，int end，int *s1，int *s2)
/*在0～END之间，找出最小和次小的两个节点序号，返回s1、s2*/
{
int i;
int min1= INT_MAX;
int min2 = INT_MAX;
for (i = 0; i <= end; i++) {/*找最小的节点序号*/
if((_____) && (HT[i].weight < minl1)) {
*s1  =  i;
min1 = HT[i].weight;
}
}
for(i = 0; i <= end; i++) {*找次小节点的序号*/
if((HT[i].parent == 0) && (_____)
```

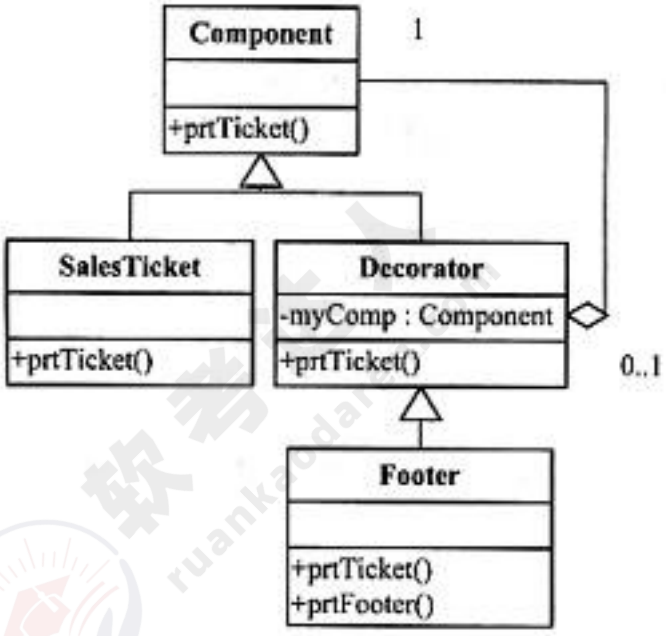
```
&& (min2 > HT[i].weight)) {
*s2 = i;
min2 = HT[i].weight;
}
}
}
void HuffmanTreeCreat(HuffmanTree &HT)/*建立HUFFMAN树*/
{
int i;
int m = 2 * n - 1;
int s1,s2;
for(i = n; i < m; i++) {
Select(_____);
HT[s1].parent = i;
HT[s2].parent = i;
HT[s1].sigh = left_child;
HT[s2].sigh = right_child;
HT[i].weight = _____;
}
}
void HuffmanTreeEncoding(char sen[],HuffmanTree HT)
{    /*将句子进行编码*/
int i = 0;
int j;
while(sen[i] != '\0') {
for(j = 0; j < n; j++) {
if (HT[j].letter == sen[i]) { /*字母匹配则用代码取代*/
strcat(coding, _____);
break;
}
}
i++;
if (sen[i] == 32) i++;
}
printf("\n%s",coding);
}
```

试题五

阅读下列函数说明和C++代码，将应填入画横线处的字句写在对应栏内。

9、[说明]

在销售系统中常常需要打印销售票据，有时需要在一般的票据基础上打印脚注。这样就需要动态地添加一些额外的职责。如下展示了Decorator(修饰)模式。SalesOrder对象使用一个SalesTicket对象打印销售票据，先打印销售票据内容，然后再打印脚注。图显示了各个类间的关系。以下是C++语言实现，能够正确编译通过。



[C++代码]

```
class Component{
public:
______ void prtTicket() = 0;
};
class SalesTicket : public Component{
public:
void prtTicket() {
cout<<"Sales Ticket!"<<endl;
}
};
class Decorator : public Component{
public:
virtual void prtTicket();
Decorator(Component *myC.;
private:
______ myComp;
};
Decorator::Decorator(Component *myC.
{
myComp = myC;
}
void Decorator::prtTicket()
{
myComp->prtTicket();
}
class Footer : public Decorator{
public:
Footer(Component *myC.;
void prtTicket();
void prtFooter();
};
Footer::Footer(Component *myC. : ______{}
void Footer::prtFooter()
{
cout<<"Footer"<<end1;
```

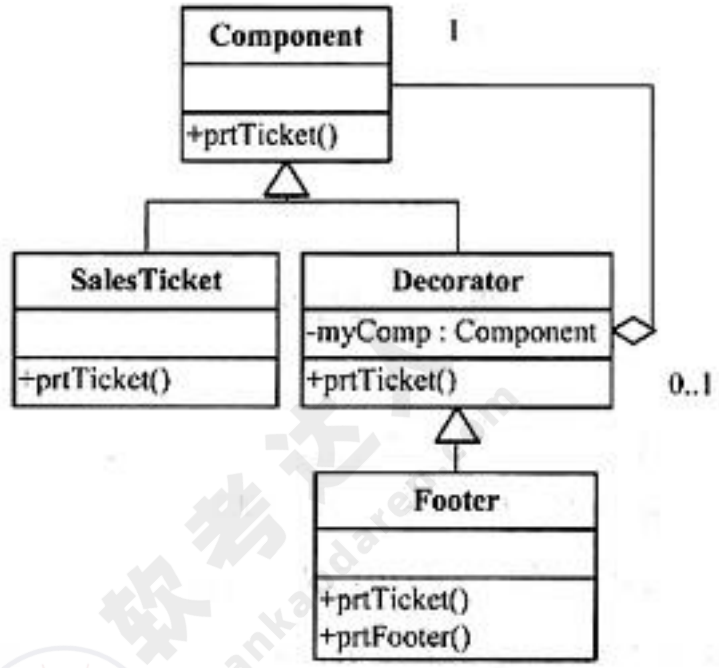
```
}
void Footer::prtTicket ()
{
_______;
prtFooter();
}
class SalesOrder{
public:
void prtTicket();
};
void SalesOrder::prtTicket()
{
Component *myST;
myST = new Footer_______;
myST->prtTicket();
}
```

试题六

阅读以下说明和*Java*代码，将应填入画横线处的字句写在对应栏内。

10、[说明]

在销售系统中常常需要打印销售票据，有时需要在一般的票据基础上打印脚注。这样就需要动态地添加一些额外的职责。如下展示了Decorator(修饰)模式。SalesOrder对象使用一个SalesTicket对象打印销售票据。显示了各个类间的关系。以下是Java语言实现，能够正确编译通过。



[Java代码]

```
//Component.java文件
public _______ class Component {
abstract public void prtTicket();
//SalesTicket.java文件
public class SalesTicket extends Component {
public void prtTicket() {
//Sales ticket printing code here
System.out.println("SalesTicket");
```

```
}
}
//Decorator.java文件
public abstract class Decorator extends Component {
public void prtTicket() {
if(myComp != null)myComp.prtTicket();
}
private ______ myComp;
public Decorator(Component myC. {
myComp = myC;
}
}
//Footer.java文件
public class Footer extends Decorator {
public Footer(Component myC. {
______;
}
public void prtTicket() {
______;
prtFooter();
}
public void prtFooter() {
//place printing footer code here
System.out.println("Footer");
}
}
//SalesOrder.java文件
public class SalesOrder {
void prtTicket() {
Component myST;
myST = new Footer(______);
//Print Ticket with footers as needed
myST.prtTicket();
}
}
```

试题七

阅读以下说明和*C*代码，将应填入画横线处的字句写在对应栏内。

11、[说明]

函数combine(a,b,c)是计算两个整数的组合数。由于计算结果可能超出long整型的可表示范围，故采用数组方式存储，例如：k位长整数m用数组c[]存储结构如下：$m=c[k]\times10^{k-1}+c[k-1]\times10^{k-2}+\ldots+c[2]\times10+c[1]$，利用c[0]存储长整数m的位数，即c[0]=k。数组的每个元素只存储长整数m的一位数字，长整数运算时，产生的中间结果的某位数字可能会大于9，这是就应该调用format将其归整，使数组中的每个元素始终只存储长整型的一位数字。

整数据a和b(a≥b)的组合为：$c(a,b)=C_a^b=\frac{a!}{(a-b)!b!}=\frac{a\times(a-1)\times\cdots\times(a-b+1)}{b!}=\frac{u_1\times u_2\times\cdots\times u_b}{d_1\times d_2\times\cdots\times d_b}$，其中$u_1=a,u_2=a-1,\ldots,u_b=a-b+1,d_1=1,d_2=2,\ldots,d_b=b$。为了计算上述分式，先从$u_1,u_2,\ldots,u_b$中去掉$d_1\times d_2\times\ldots\times d_b$的因子，得到新的$u_1,u_2,\ldots,u_b$，然后再将它们相乘。

[函数]

```
#define MAXN 100
int gcd(int a, int b)//求两个整数a和b的最大公因子
```

```
{
if(a < b) {
int c = a; a = b; b = c;
}
for(int i = b; i >= 2; i--) {
if(_____)return i;
}
return 1;
}
void format(int *a)//将长整型数纰归整
{
int i;
for(i =1; i > a[0] || a[i] >= 10; i++) {
if(i >= a[0]) _____;
a[i+1] += a[i]/10;
a[i] = a[i]%10;
}
if(i > a[0]) _____;
}
void combine(int a, int b, int *c)
{
int i, j, k, x;
int d[MAXN], u[MAXN];
k = 0;
for(i =a; i >= a-b+1; i--)u[++k] = i;
u[0] = b;
for(i = 1; i <= b; i++)d[i] = i;
for(i = i; i <= u[0]; i++) { //从u各元素去掉d中整数的因子
for(j = i; j <= b; j++) {
x = gcd(u[i], d[j]);//计算最大公约数
u[i] /= x;
d[j] /= x;
}
}
_____; c[1] = 1;长整数C初始化
for(i = i; i <= u[0]; i++) {//将u中各整数相乘，存于长整数C中
if(u[i] != 1) {
for(j = i; j <= c[0]; j++) {
e[j] =_____;
}
format (c);//将长整数c归整
}
}
}
```

答案：

试题一

1、“房租文件”和“交费文件”

分层数据流图中，只涉及单个加工的文件不必画出，可在子图中再画。依此标准，图2中文件“房租文件”和“交费文件”不必画出。

2、①加工1子图中，遗漏了从住户基本信息文件到加工1.1(入住单校验)的数据流。②加工1子图中，加工1.6(制作住房分配报告)遗漏了输出数据流：住房分配表。③加工2子图中，加工2.1(计算月租费)遗漏了输入数据流：月附加费表。④加工2子图中，加工2.4(制作收据)遗漏了输出数据流：收据。

分层数据流图时刻牢记父图与子图平衡原则。对这种数据流缺失题目，认真对照父图与子图就可得出答案。另外，还要注意与文件的交互，包括错误数据流大多也是出在此。

3、①交费凭证中有非法字符；②交费文件中不存在与之对应的交费凭证。

试题二

4、S(Sno,Sname,Status,City)，主键为SNo。

P(PNo,PName,Color,Weight,City)，主键为PNo。

SP(SNo,PNo,Status,Qty)，主键为(SNo,PNo)。

E-R模型向关系模型的转换应遵循如下原则：

· 每个实体类型转换成一个关系模式。

· 一个1:1的联系(一对一联系)可转换为一个关系模式，或与任意一段的关系模式合并。

· 一个1:n的联系(一对多联系)可转换为一个关系模式，或与n端的关系模式合并。

· 一个n:m的联系(多对多联系)可转换为一个关系模式，两端关系的码及其联系的属性为该关系的属性，而关系的码为两端实体的码的组合。

· 三个或三个以上多对多的联系可转换为一个关系模式，诸关系的码及联系的属性为关系的属性，而关系的码为各实体的码的组合。

· 具有相同码的关系可以合并。

根据题述易于判断供应商的主键为供应商编号SNo，零件的主键为零件编号PNo。

5、PRIMARY KEY SNo

创建表时往往需要声明主键、外键、非空、唯一等完整性约束条件，表S中，SNo是主键，声明主键有两种实现手法：PRIMARY KEY(SNO)，或者NOT NULL、UNIQUE，不同的是NOT NULL是列级约束，必须在列名之后声明，而PRIMARY KEY是表级约束。创建表的完整SOL语句如下：

CREATE TABLE＜表名＞(＜列名＞＜数据类型＞[列级完整性约束条件]

[＜列名＞＜数据类型＞[列级完整性约束条件]]…

[＜表级宽整性约束条件＞])

6、AND

IN

P

GROUPBY

试题三

7、flag:=1

carry:=0

carry:0

LC[i+1]:0

LC[i+2]:0

对这种题目，首先阅读说明，从功能上了解程序的结构，把握整体框架，再仔细对照阅读流程图，且勿先阅读流程图。

仔细阅读完说明，就知道整体框架了：先决定符号，再进行绝对值的加减，其中加减是用flag来标识的。对于加法，要注意进位，特别是最高进位；对于减法，要注意借位，亦即负进位，在此不用考虑不够减情况，但仍要特别注意最高借位，当最高位正好为0时，要把高位所有的0去掉。

试题四

8、HT[i].parent==0
*s1 !=i
HT,i-1,&s1,&s2
HT[s1].weight+HT[s2].weight
HT[j].code

试题五

9、virtual
Component*
Decorator(myC)
Decorator::prtTicket()
new SalesTicket()

试题六

10、abstract
Component
super(myC)
super.prtTicket()
new SalesTicket()

试题七

11、a%i == 0 && b % i== 0
a[i+1] = 0
a[0] = i
c[0] =1
u[i]*c[j]