(A)中级软件设计师下午试题模拟65

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]

某供销系统接受顾客的订货单，当库存中某配件的数量小于订购量或库存量低于一定数量时，向供应商发出采货单；当某配件的库存量大于或等于订购量时，或者收到供应商的送货单时并更新了库存后，向顾客发出提货单。该系统还可随时向总经理提供销售和库存情况表。

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的子图，B.是加工2的子图。

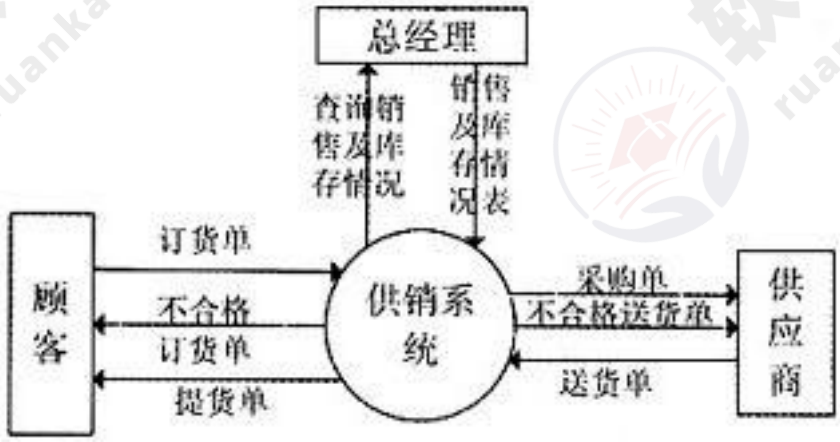


图1

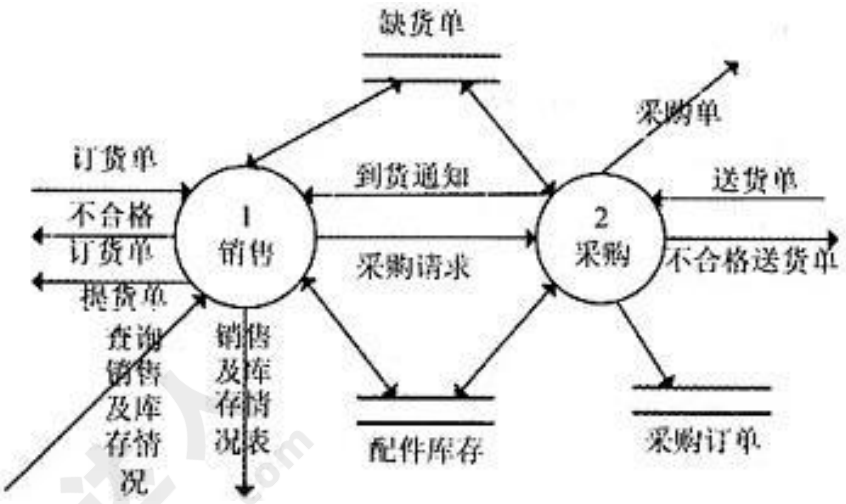


图2

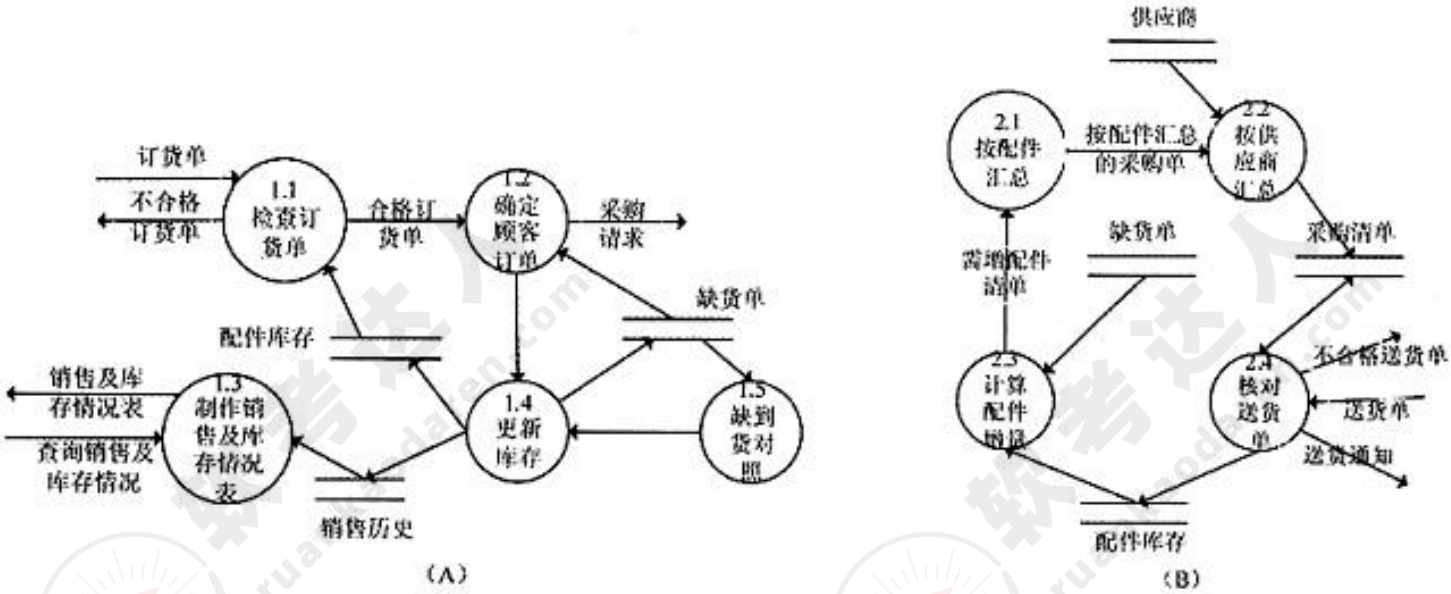


图3

[数据字典]

1数据流条目

订货单=配件号+配件名+规格+数量+顾客名+地址

提货单=订货单+金额

采货单=配件号+配件名+规格+数量+供应商名+地址

送货单=配件号+配件名+规格+数量+金额

2文件说明

文件名：配件库存

组成：{配件号+配件名+规格+数量+允许的最低库存量}

1、根据题意，图2中哪个文件可不必画出。

2、根据题意，指出图3A.中缺失的数据流的名称，并指出该数据流的起点和终点。

3、根据题意，指出图3B.中缺失的数据流的名称，并指出该数据流的起点和终点。

试题二

阅读下列说明和E-R图，回答问题1至问题3。

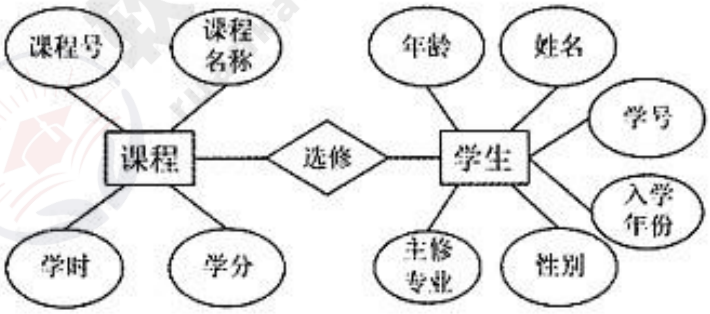
[说明]

某学校的教学系统描述如下：

学生信息包括：学号(SNo)、姓名(Sname)、性别(Sex)、年龄(Age)、入学年份(Year)、主修专业(Major)，其中学号是入学时唯一编定的。

课程信息包括：课程号(CNo)、课程名称(CName)、学时(Period)、学分(Credit)，其中课程号是唯一编定的。

一个学生可选多门课，每个学生选每门课有一个成绩。图是经分析得到的E-R图。



4、 设基本表：Student(SNo, SName, Sex, Age, Year, Major), Course(CNo, Cname, Period, Credit), Grade(SNo, CNo, Grade)通过如下SQL语句建立，请在SQLN句空缺处填

入正确的内容。

```
CREATE TABLE Student (SNo CHAR(6) NOT NULL,
SName CHAR(20),
Sex CHAR(1),
Age INTEGER,
Year CHAR(4),
Major CHAR(20),
_____;
CREATE TABLE Course (CNo CHAR(6) NOT NULL,
CName CHAR(20),
Period INTEGER,
Credit INTEGER,
_____;
CREATE TABLE Grade  (SNo CHAR(6)  NOT NULL,
CNo CHAR(6) NOT NULL
Grade REAL,
_____,
_____,
_____;
```

5、 若另有表Teach(CName, TName)存储教师任课情况，Tname表示教师名。用SQL创建一个含有学号、姓名、课程名、成绩、任课教师名的"主修专业为计算机CS"的学生成绩视图，并要求进行修改、插入操作时保证该视图只有计算机系的学生。请在SQL语句空缺处填入正确的内容。

```
CREATE VIEW SG _____
SELECT Student.SNo, SName, Grade, Course.CName, TName
FROM Student, Grade, Teach,
WHERE _____
AND _____
AND Major = 'CS',
_____;
```

6、 如下的SQL语句是用于查询"每个学生的选修课程数、总成绩、平均成绩"的不完整语句，请在空缺处填入正确的内容。

```
SELECT Student.SNo, _____, SUM(Grade), AVG(Grade)
FROM Student, Grade
WHERE Student.SNo = Grade.SNo,
GROUP BY _____;
```

试题三

阅读下列说明和图，回答问题1至问题2。

[说明]

银行的自动柜员机(ATM)的功能描述如下：

7金融卡与信用卡识别：包含伪卡识别以及密码验证；

8主菜单项：这是一台ATM最主要的人机界面，提供各项功能给客户，具体有：提款、转帐、更改密码以及存款；

9结束操作：客户执行完"菜单项"的功能后，可以选择"打印单据"或"不打印单据"，选好后就结束此次交易。

注意，ATM除了能处理本行的银行卡外，其他银行的银行卡也应该能处理，通过"金融中心"与其他银行主机进行数据交换。另外，为了方便，ATM还提供快捷提款，并提供代交费功能(代交费是以转帐的方式处理的)。

该系统采用面向对象方法开发，系统中的类以及类之间的关系用UML类图表示。

7、图1是该系统的用例图，根据题意，用题中所述术语指出图1中参与者A、B分别是什么，用例C、D分别是什么。

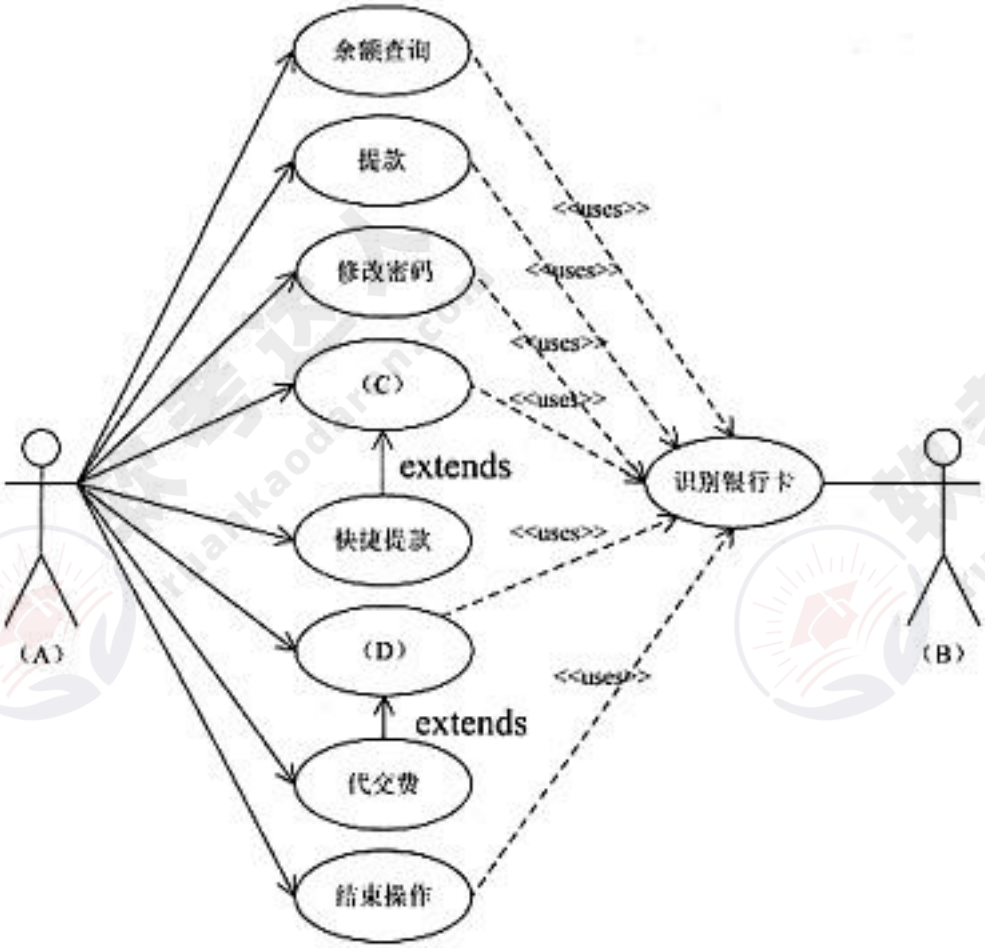


图1

8、ATM机有如下状态：空闲、银行卡验证、业务选择等待、取款金额输入、密码修改、出钞、单据打印。ATM机一般处于空闲状态，当有客户插入银行卡，则进行银行卡验证，若银行卡无效则结束服务，否则进入业务选择等待。业务有取款、修改密码等，也可以选择退出结束服务，ATM返回空闲状态。选择取款业务后，等待取款金额输入，确认后判断余额是否足够，若余额不足，则给出提示信息，并进入业务选择等待；若余额充足，则出钞，若客户需要打印单据则进入单据打印状态，否则返回业务选择等待。选择任意一个业务后，可以取消返回业务选择等待。图2描述了ATM状态的转变情况。

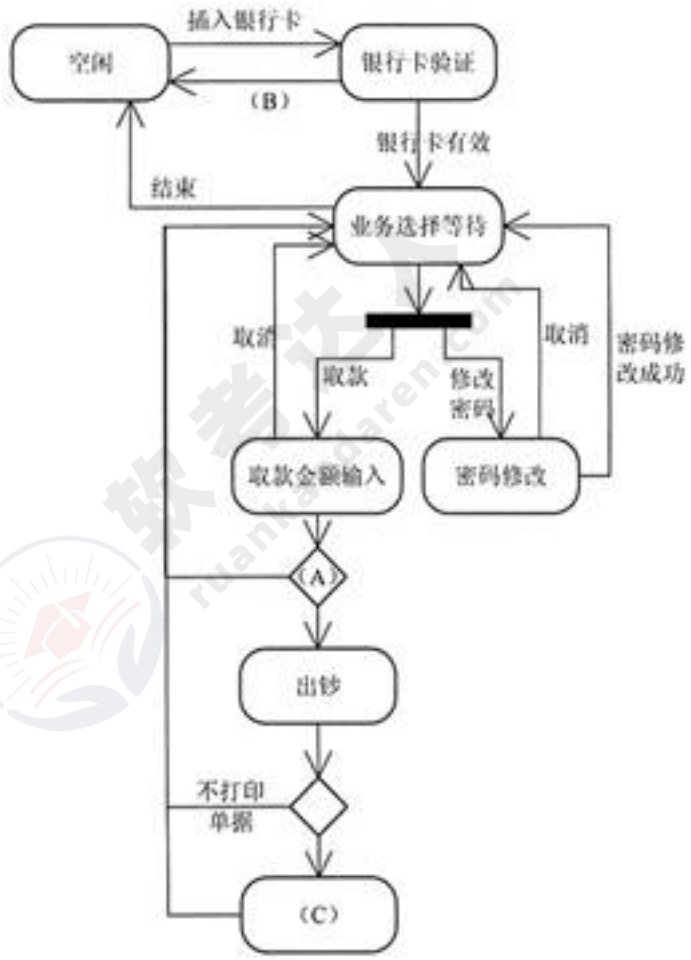


图2

请指出判定A、转换B及状态C分别是什么。

试题四

阅读下列说明和图，回答问题1到问题3。

[说明]

操作系统中，死锁(Deadlock)是指多个进程在运行的过程中因争夺资源而造成的一种僵局。当进程处于这种僵持状态时，若无外力作用，它们都将无法再向前推进。

面对死锁问题有两个解决方案：预防死锁和避免死锁。

预防死锁是一种较简单和直观的事先预防方法。该方法是通过设置某些限制条件，去破坏产生死锁的四个必要条件中的一个或多个，以此来预防死锁的发生。预防死锁由于较易实现，已被广泛应用，但由于所施加的限制条件往往太严格，可能会导致系统资源利用率和系统吞吐量的降低。

避免死锁同样是属于事先预防的策略，但它无须事先采取各种限制措施去破坏产生死锁的四个必要条件，而是在资源分配过程中，用某种方法去防止系统进入不安全状态，从而避免发生死锁。

银行家算法(Banker's algorithm)是Dijkstra于1965年提出的一个经典的避免死锁的算法。形象地描述银行发放贷款不能使有限可用资金匮乏而导致整个银行无法运转的思路，也就是说每次请求贷款，银行要考虑他能否凭着贷款完成项目，并还清贷款使银行运转正常。令Request(i)是进程P(i)请求向量，如果Request(i)[j]=k则进程P(i)希望请求j类资源k个。具体算法步骤如下：

9如果Request(i)>Need(i)则出错(请求量超过申报的最大量)，否则转到10；

10如果Request(i)>Available则P(i)等待，否则转12；

12系统对P(i)所请求的资源实施试探分配，并更改数据结构中的数值；

```
14      Available = Available - Request(i):
          Allocation(i) = Allocation(i) + Request(i);
           Need(i) = Need(i) - Request(i);
```

15执行安全性算法，如果是安全的，则承认试分配，否则废除试分配，让进程P(i)继续等待。

所谓系统是安全的，是指系统中的所有进程能够按照某一种次序分配资源，并且依次运行完成，这种进程序{P1, P2, …, Pn}就是安全序列。如果存在这样一个安全序列，则系统是安全的；如果系统不存在这样一个安全序列，则系统是不安全的。

9、简述产生死锁的四个必要条件。

10、设系统中有三种类型的资源(A, B,C.和五个进程(P0, P1, P2, P3, P4)，某时刻的资源分配状态如表所示。给出该时刻存在的一个安全序列。

	Allocation				Max				Available		
	A	B	C		A	B	C				
P0	0	1	0		7	5	3				
P1	3	0	2		3	2	2		A	B	C
P2	3	0	2		9	0	2		2	3	0
p3	2	1	1		2	2	2				
P4	0	0	2		4	3	3				

11、若系统中有同类资源16个，有4个进程共享该资源。已知P1、P2、P3、P4所需总资源分别是8、5、9、6。各进程请求资源次序为(序号，进程，申请量)：(1, P1, 6)、(2, P2, 4)、(3, P3, 5)、(4, P4,1)、(5, P1, 1)、(6, P2, 1)。若用银行家算法为它们分配资源，分析每一步请求以后，各个进程还需的资源数以及系统所剩资源数，并指出系统是否安全。注，当某序号的申请导致系统不安全时，跳过该请求(拒绝该请求)继续往下判断，相当于将该进程阻塞。

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。

12、[说明]

为网球比赛的选手安排比赛日程。设有n(n=2^m)位选手参加网球循环赛，循环赛共进行n-1天，每位选手要与其他n-1位选手赛一场，且每位选于每天赛一场，不轮空。

设n位选手被顺序编号为1, 2, …, n，比赛的日程表是一个n行n-1列的表，第i行j列的内容是第i号选手第j天的比赛对手。用分治法设计日程表，就是从其中一半选手(2^{m-1}位)的比赛日程导出全体2^m选手的比赛日程。从众所周知的只有两位选手的比赛日程出发，反复这个过程，直至为n位选手安排好比赛日程为止。

如两位选手比赛日程表如下所示：

	1
1	2
2	1

如四位选手比赛日程表如下所示：

	1	2	3
1	2	3	4
2	1	4	3
3	4	1	2
4	3	2	1

函数中使用的预定义符号如下：

```
#define M 64
int a[M+1][M];
```

[函数]

```
void main() {
int twom1, twom, i, j, m, k;
printf("指定n(=2的k次幂)位选手, 请输入k: \n");
scanf("*d", &k);
```

```
/*预设两位选手的比赛日程*/
a[l] [1] = 2;
a[2] [1] = 1;
m = 1;
twoml = 1;
while(_______) {
m++;
twoml += twoml;
twom = twoml * 2;/*为2^m位选手安排比赛日程*/
/*填日程表的左下角*/
for(i = twoml + i; _______; i++) {
for(j = i; j <= twoml - i; j++) {
a[i] [j] = a[i - twoml] [j] + twoml;
}
}
/*填日程表的右上角*/
a[1] [twoml] = _______;/*填日程表右上角的第1列*/
for(i = 2; i <= twoml; i++) {
a[i] [twoml] = a[i - 1] [twoml] + i;
}
/*填日程表右上角的其他列，参照前一列填当前列+/
for(j = twoml + 1; j < twom; j++) {
for(i = i; i < twoml; i++) {
a[i] [j] = _______;
}
a[twoml] [j] = a[l] [j - 1];
}
/*填日程表的右下角*/
for(j = twoml; j < twom; j++) {
for(i = i; i <= twoml; i++) {
a[_______] [j] = i;
}
}
/*输出日程表*/
for(i = i; i <= twom; i++) {
for(j = i; j < twom; j++) {
printf("%4d", a[i] [j]);
}
printf("\n");
}
printf("\n");
}
}
```

试题六

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。

13、[说明]

很多时候，希望某些类只有一个或有限的几个实例，典型解决方案是所谓单身(singleton)模式。但在多线程情况下，singleton模式有可能出现问题，需要进行同步检查。如果对"检查singleton对象是否已经创建"进行同步，则存在严重的瓶颈，所有的线程都必须等待检查对象是否存在。解决方式是一种称为Double-Checked-Locking模式，其意图是将非必须的锁定优化掉，

同步检查最多只发生一次，因此不会成为瓶颈。以下是C++语言实现，能够正确编译通过。

[C++代码]

```
class USTax{
______:
USTax( ) { }; //构造函数
public:
static USTax* getInstance();
private:
static USTax* instance;
};
______ = NULL;
USTax* USTax::getInstance() {
if (instance == NULL) {
//进行某种同步
cout<<"实例暂时不存在"<<endl;
if(______) {
cout<<"实例不存在，创建实例... "<<endl;
instance = ______;
cout <<"实例创建成功 "<<endl;
}
else{
cout <<"实例已被创建了"<<endl;
}
}
else{
cout<<"实例已经存在" <<endl;
}
return ______;
}
```

试题七

阅读以下函数说明和*Java*代码，将应填入画横线处的字句写在对应栏内。

14、[说明]

很多时候，希望某些类只有一个或有限的几个实例，典型解决方案是所谓单身(Singleton)模式。但在多线程情况下，singleton模式有可能出现问题，需要进行同步检查。如果对"检查Singleton对象是否已经创建"进行同步，则存在严重的瓶颈，所有的线程都必须待检查对象是否存在。解决方式是一种称为Double-Checked-Locking模式，其意图是将非必须的锁定优化掉，同步检查最多只发生一次，因此不会成为瓶颈。以下是Java语言实现，能够正确编译通过。

[Java代码]

```
public class USTax {
private static USTax instance = null;
______ USTax() {}
private ______ static void doSync() {
if(instance == null) {
System.out.println ("实例不存在，创建实例...");
instance = ______;
System.out.print ln ("实例创建成功");
}else{
System.out.println ("实例已被创建了");
}
}
```

```
public static USTax getInstance() {
if(instance == null) {
System.out.println ("实例暂时不存在");
_____; / /同步控制
}else{
System.out.println ("实例已经存在");
}
return _____;
}.
}
```

答案：

试题一

1、采购订单

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“采购订单”只与加工采购有关，故不必画出。

2、起点：库存配件，终点：确定顾客订单

起点：库存配件，终点：制作的销售及库存情况表

提货单，起点：更新库存，终点：顾客

到货通知，起点：采购，终点：缺到货对照

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。

根据题述，图3A是加工1的细化图，加工1在图2中，认真对照其输入输出数据流。发现缺失数据流“提货单”和“到货通知”，进一步确定数据流的起点和终点。“提货单”是输出数据流，起点应为加工“更新库存”，其终点自然是“客户”；“到货通知”是输入数据流，终点应为加工“缺到货对照”，起点应为加工“采购”。

另外，确定顾客订单时，需要检查库存配件，因此应有文件“配件库存”到加工1.2的数据流。同理，也应有文件“配件库存”到加工1.4的数据流。

3、采购单，起点：按供应商汇总，终点：供应商

采购请求，起点：销售，终点：计算配件增量

同问题2的分析，仔细对照父图与子图的输入输出数据流，并确认与文件相关的数据流。

试题二

```
4、PRIMARY KEY(SNo)
PRIMARY KEY(Cno)
PRIMARY KEY(SNo, CNo)
FOREIGN KEY(SNo)REFERENCES Student(SNo)
FOREIGN KEY(CNo)REFERENCES Course(CNo)
5、AS
Student.SNo=Grade.SNo
Course.CName=Teach.CName
WITH CHECK OPTION
```

创建视图：CREATE CIEW视图名(列表名)

```
AS SELECT查询子句
[WITH CHECK OPTION]6、COUNT(Grade.CNo)
Student.SNo
```

试题三

7、A：“客户” B：“金融中心” C：“提款” D：“转账”

图1给出了系统用例图，用例图(use case diagram)展现了一组用例、参与者(actor)以及它们之间的关系。

易知参与者A是“客户”，参与者B为“金融中心”。

用例“快捷提款”是“提款”的扩展，因此用例C是“提款”；用例“代交费”是“转账”的扩展，因此用例D是“转账”。

8、A：“金额是否足够” B：“银行卡无效” C：“打印单据”

取款时，若金额不足，自然取款失败，因此判定A是判断“金额是否足够”。

当银行卡验证失败，服务结束，ATM机转入“空闲”，故B是“银行卡无效”。

状态C为“打印单据”。

试题四

9、死锁的发生必须具备四个必要条件：

· 互斥条件：进程对所分配到的资源进行排他性使用，即在一段时间内某资源只有一个进程占用；

· 请求和保持条件：进程已经保持了至少一个资源但又提出了新的资源请求，若得不到满足则阻塞该进程，但其保持已获得的资源不释放；

· 不剥夺条件：进程已获得的资源，在未使用完之前，不能被剥夺，只能在使用完时由自己释放；

· 环路等待条件：在发生死锁时，必然存在一个进程-资源的环形链，即进程集合{P1，P2，...，Pn}中的P1等待P2占用的资源，P2等待P3占用的资源，...，Pn等待P0占用的资源。

10、{P1，P3，P0，P4，P2}

11、①(1，P1，6)余资源10。此时P1还需2，P2还需5，P3还需9，P4还需6。系统存在安全序列：{P1，P2，P3，P4}，故系统安全。②(2，P2，4)余资源6。此时P1还需2，P2还需1，P3还需9，P4还需6。系统存在安全序列：{P1，P2，P3，P4}，故系统安全。③(3,P3，5)余资源1。此时P1还需2，P2还需1，P3还需4，P4还需6。系统存在安全序列：{P2,P1，P3，P4}，故系统安全。④(4，P4，1)余资源0。此时P1还需2，P2还需1，P3还需4，P4还需5。系统不存住安全序列，故系统不安全。请求(4，P4，1)是不安全的，排除该请求，继续往后判断。⑤(5，P1，1)余资源0。此时P1还需1，P2还需1，P3还需4，P4还需6。系统不存在安全序列，故系统不安全。请求(5，P1，1)是不安全的，排除该请求，继续往后判断。⑥(6，P2，1)余资源0。此时P1还需2，P2还需0，P3还需4，P4还需6。P2进程资源已得到完全满足，P2完成后，资源释放。系统存在安全序列：{P2,P1，P3，P4}，故系统安全。至此，6个进程均进行了是否分配资源判断。

试题五

12、m＜k

i＜=twom

twoml+1

[i+1][j-1]

[i][j]

试题六

13、private

USTax*USTax::instance

instance==NULL

new USTax

instance

试题七

14、private

synchronized

new USTax()

doSync()

```
jnstance
```