中级软件设计师下午试题模拟64

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]

考务处理系统具有如下功能：

1对考生送来的报名单进行检查。

2对合格的报名单编好准考证号后将准考证送给考生，并将汇总后的考生名单送给阅卷。

3对阅卷站送来的成绩清单进行检查，并根据考试中心制订的合格标准审定合格者。

4制作考生通知单送给考生。

5进行成绩分类统计(按地区、年龄、文化程度、职业、考试级别等分类)和试题难度分析，产生统计分析表。

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。

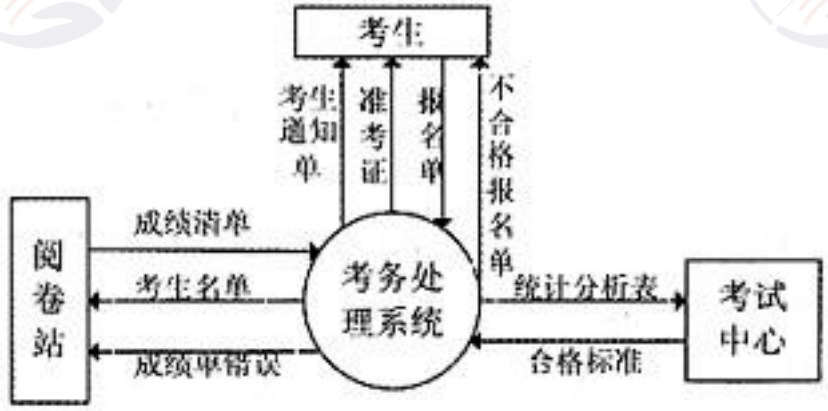


图1

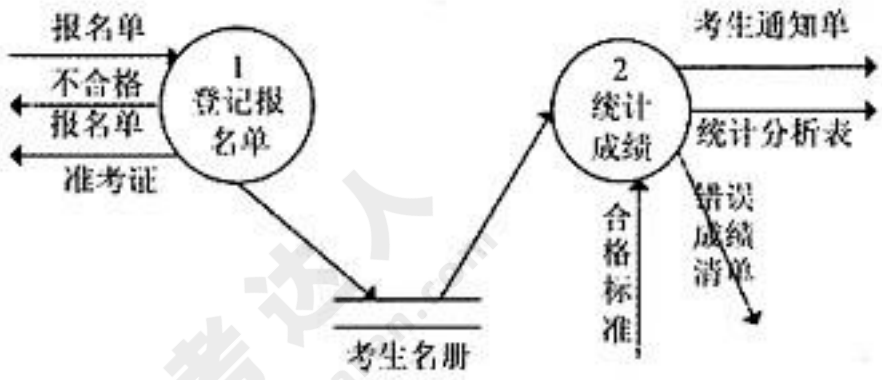


图2

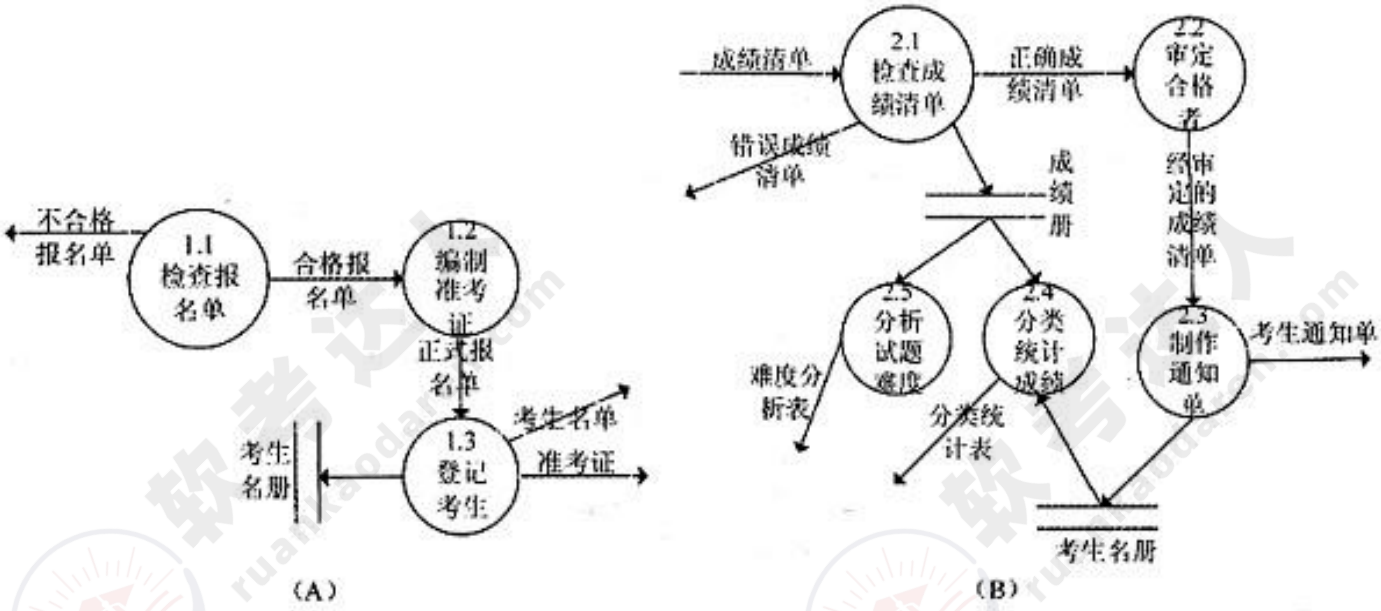


图3

[数据字典]

报名单=地区+序号+姓名+性别+年龄+文化程度+职业+考试级别+通信地址

正式报名单=报名单+准考证号

准考证=地区+序号+姓名+准考证号+考试级别

考生名单={准考证号+考试级别}

统计分析表=分类统计表+难度分析表

考生通知单=考试级别+准考证号+姓名+合格标志+通信地址

1、根据题意，指出0层数据流图(图2)中缺失的数据流的名称，并指出该数据流的起点和终点。

2、根据题意，指出加工1子图(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，指出加工2子图(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。加工2子图(图3)中有一条数据流是错误的，请指出这条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]

图1是某医院组织的结构图。该医院分为多个病区，每个病区有一个唯一的编号，一个病区包括多个病房，多名医生；每位医生有一个唯一的编号，负责管辖其主治病人的所有病房；病人住院后给以一个唯一的编号，根据"患何病科"住在相应病区的某个病房里，有且仅有一位医生担任主治医生，除主治医生外其他医生不对其负责。

现假定病区名称有"内科"和"外科"，"内科"病区又细分为多个病区，以编号区分，名称都为"内科"；"外科"病区亦然。图2是经分析得到的E-R图。

图1

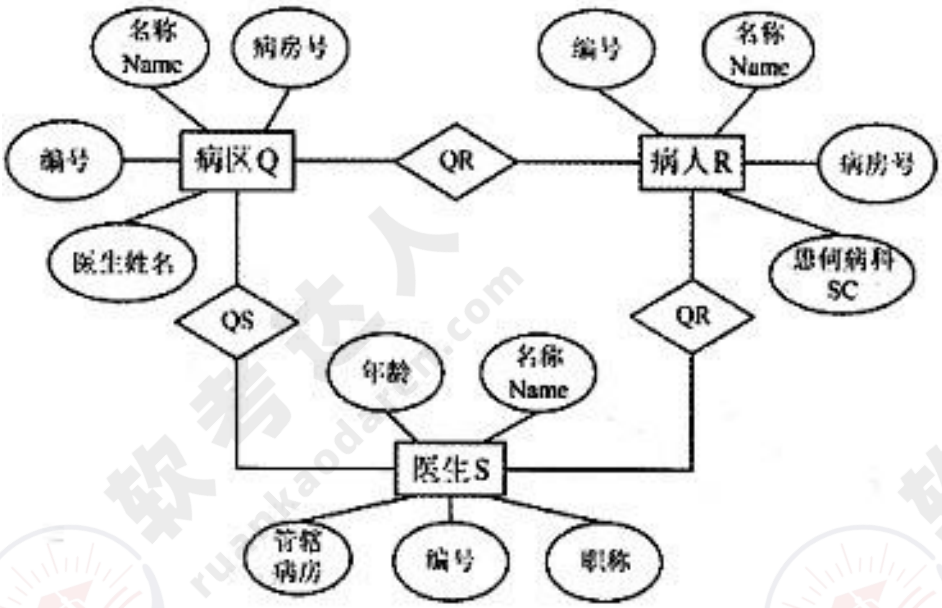


图2

4、实体间的联系有"一对一"、"一对多"和"多对多"，指出图2中各联系分别属于哪一种。

5、选出正确的关系代数表达式。

查询所有"外科"病区和"内科"病区的所有医生姓名：

A. $\sigma_{Name="外科"\vee Name="内科"}((\Pi_4 Q))$　　B. $\sigma_{Name="外科"\wedge Name="内科"}(\Pi_4(Q))$

C. $\Pi_4(\sigma_{Name="外科"\vee Name="内科"}(Q))$　　D. $\Pi_4(\sigma_{Name="外科"\wedge Name="内科"}(Q))$

6、查询内科病区患胃病的病人的姓名。

A. $\sigma_{Name="外科"\vee SC="胃病"}(\Pi_2(R))$　　B. $\sigma_{Name="内科"\wedge SC="胃病"}(\Pi_2(R))$

C. $\Pi_2(\sigma_{Name="外科"\vee SC="胃病"}(R))$　　D. $\Pi_2(\sigma_{Name="内科"\wedge SC="胃病"}(R))$

7、层次模型不能直接表示多对多联系，为什么?可采用哪些方法进行多对多联系的表示。

试题三

阅读下列说明和图，回答问题1至问题2。

[说明]

移动电话是传统固定式电话的延伸，通过无线电网络可以与千里之外的朋友沟通而不受电话线的束缚。现在的移动电话功能更全面，除了作为电话使用外，还可以发送短信，可以管理电话簿，可以下载铃声、图案。

手机由键盘、显示屏以及移动通信设备组成，移动通信设备负责发送和接收信号，与基站进行连线。打电话的流程如下：

8用户拨电话号码，每按下一个数字键显示屏上显示相应数字；

9按OK键进行连线，显示屏上显示"连线中…"，请连接基站，基站通过移动电话网络连接到对方手机，若有误则返回相关信息；

10接通后，显示屏显示"连线成功"；

11打电话结束后，按Cancel送出断线信号，通知移动电话基站断线，基站切断连接，显示屏显示"断线成功"。

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，图1是该系统的用例图，图2是该系统的类图，图3描述了打电话(包括断开)的序列图。

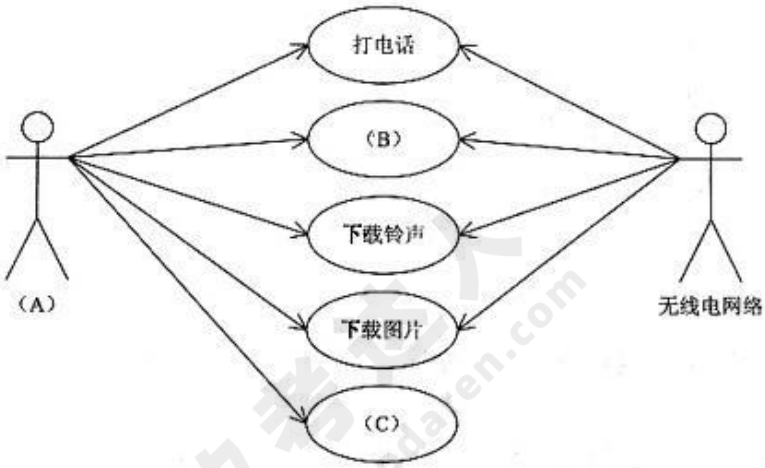


图1

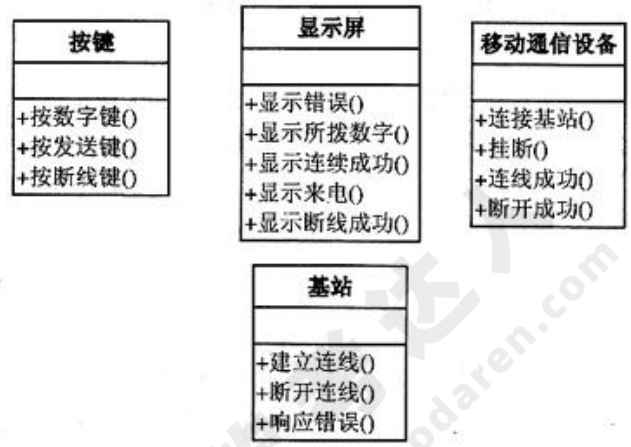


图2

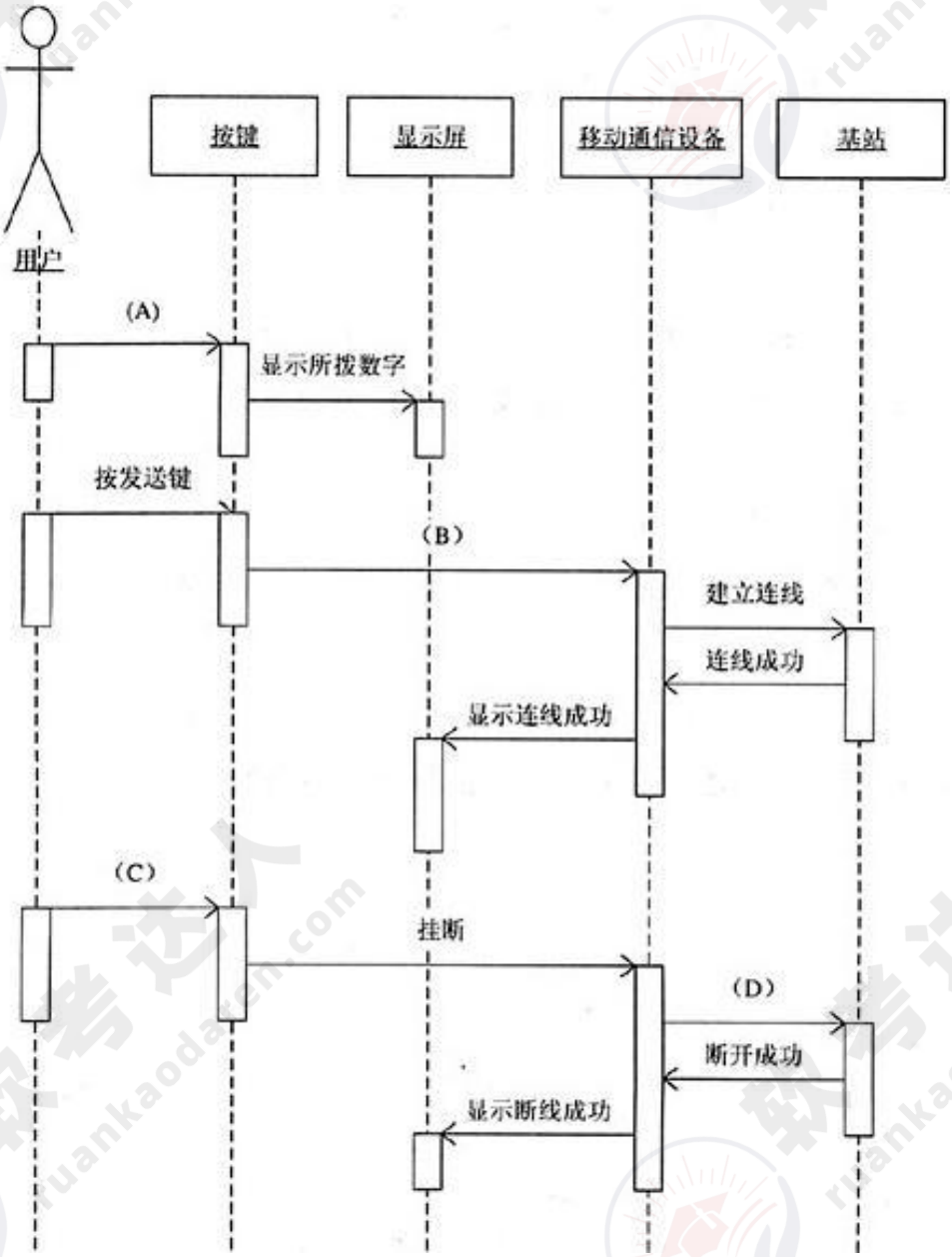


图3

8、根据题意，用题中及类图中提供的术语指出图1中的参与者A及用例B、C各是什么。

9、根据题意，用题中及类图中提供的术语指出图3所示的打电话序列图中的消息A.～D.。

试题四

阅读下列说明和图表，回答问题1到问题3。

[说明]

在多道程序系统中，各个程序之间是并发执行的，共享系统资源。CPU需要在各个运行的程序之间来回地切换，这样的话，要想描述这些多道的并发活动过程就变得很困难。为此，操作系统设计者提出了进程的概念。

进程是具有独立功能的程序关于某个数据集合上的一次动态执行过程，是系统进行资源分配和调度的独立单位。

10、进程在生命周期结束前处于且仅处于三种基本状态之一。运行态(Running)：进程占有CPU，并在CPU上运行。就绪态(Ready)：一个进程已经具备运行条件，但由于无CPU暂时不能运行的状态(当调度给其CPU时，立即可以运行)。等待态(Blocked)：指进程因等待某种事件的发生而暂时不能运行的状态，即使CPU空闲，该进程也不可运行。指出如下进程状态转换图(图1)中"状态1"～"状态3"分别是什么状态。

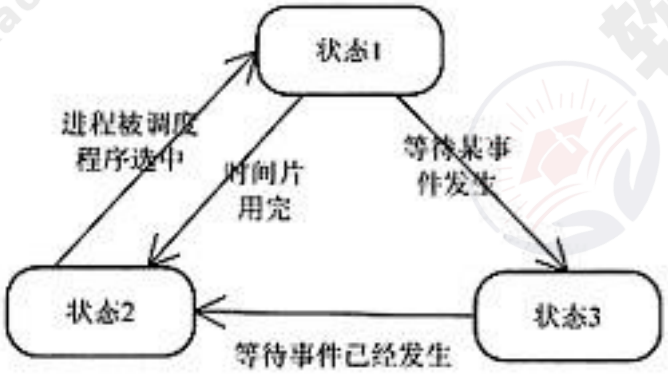


图1

11、如果单CPU系统中有N个进程，运行的进程最多几个，最少几个；就绪进程最多几个，最少几个；等待进程最多几个，最少几个？

12、进程调度算法解决以何种次序对各就绪进程进行处理机的分配以及按何种时间比例让进程占用处理机。

常见的的调度算法有：先进先出FIFO(按照进程进入就绪队列的的先后次序选择)、时间片轮转RR(进程轮流运行一个时间片)、最高优先级HPF(分配给具有最高优先级的就绪进程)。

在实际系统中，调度模式往往是几种调度算法的结合。某系统按优先级别设置若干个就绪队列，对级别较高的队列分配较小的时间片S_i(i=1, 2, …, n)，即有$S_1<S_2<…<S_n$。除第n级队列是按RR法调度之外，其他各级队列均按FIFO调度。系统总是先调度级别较高的队列中的进程，仅当该队列为空时才去调度下一级队列中的进程。当执行进程用完其时间片时便被剥夺并进入下一级就绪队列。当等待进程被唤醒时，它进入其优先级相应的就绪队列，若其优先级高于执行进程，便抢占CPU执行进程。

现有五个进程P1、P2、P3、P4、P5，它们同时依次进入就绪队列，它们所需的CPU时间和优先级如表所示。注意，优先数越大优先级越低。

进程	CPU时间	优先数
P1	10	3
P2	1	1
P3	2	3
P4	1	4
P5	5	2

在该系统中，假定不同级别的时间片为$S^i=2^{i-1}$(i为优先数)，请给出五个进程的CPU占用序列，并注明每次占用所用的时间。

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。

13、[说明]

所谓货郎担问题，是指给定一个无向图，并已知各边的权，在这样的图中，要找一个闭合回路，使回路经过图中的每一个点，而且回路各边的权之和最小。

应用贪婪法求解该问题，程序先计算由各点构成的所有边的长度(作为边的权值)，按长度大小对各边进行排序后，按贪婪准则从排序后的各边中选择组成回路的边，贪婪准则使得边的选择按各边长度从小到大选择。

函数中使用的预定义符号如下：

```
#define M 100
typedef struct{/为两端点p1、p2之间的距离, p1、p2所组成边的长度*/
float x;
int p1, p2;
}tdr;
typedef struct{/*p1、p2为和端点相联系的两个端点, n为端点的度*/
int n, p1, p2;
}tr;
typedef struct {/*给出两点坐标*/
float x, y;
}tpd;
typedef int t1[M];
int n = 10;
[函数]
float distance(tpd a, tpd b);/*计算端点a、b之间的距离*/
void sortArr(tdr a[M], int m);
/*将已经计算好的距离关系表按距离大小从小到大排序形成排序表, m为边的条数*/
int isCircuit(tr r[M], int i, int j);
/*判断边(i, J)选入端点关系表r[M]后，是否形成回路，若形成回路返回0*/
void selected(tr r[M], int i, int j);/*边(i, J)选入端点关系表r*/
void course(tr r[M], t1 1 [W]);/*从端点关系表r中得出回路轨迹表*/
void exchange(tdr a[M], int m, int b);
/*调整表排序表, b表示是否可调, 即是否有长度相同的边存在*/
void travling(tpd pd[M], int n, float dist, t1 locus[M])
/*dist记录总路程*/
{
tdr dr[M];/*距离关系表*/
tr r[M];/*端点关系表*/
int i, j, k, h, m;/*h表示选入端点关系表中的边数*/
int b;/*标识是否有长度相等的边*/
k = 0;
/*计算距离关系表中各边的长度*/
for(i = 1; i < n; i++) {
for(j = i + 1; j <= n; j++) {
k++;
dr[k].x = _____;
dr[k].p1 = i;
dr[k].p2 = j;
}
}
m = k;
sortArr (dr, m);/*按距离大小从小到大排序形成排序表*/
do {
b = i;
dist = 0;
k = h = 0;
```

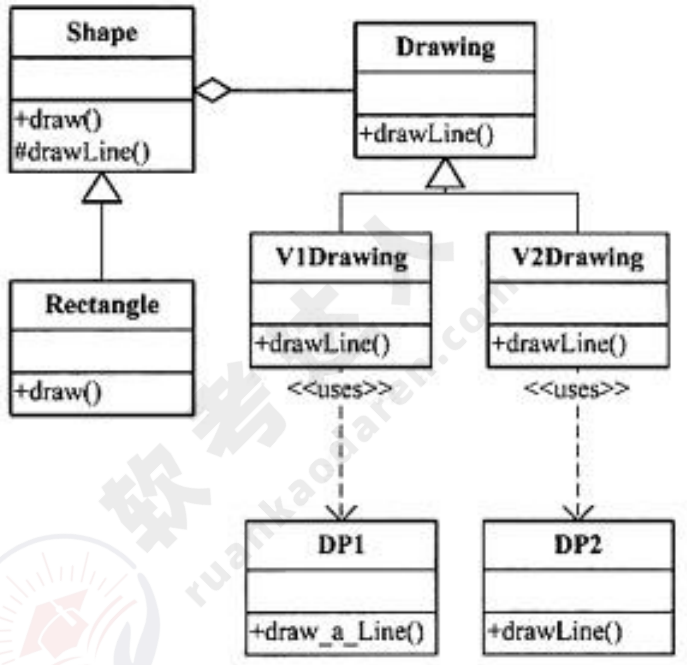
```
do{
k++;
i = dr[k].p1;
j = dr[k].p2;
if((r[i].n <= 1) && (r[j].n <= 1)) {/*度数不能大于2*/
if(_______) {
/*若边(i, j )加入r后形成回路，则不能加入*/
_______;
h++;
dist += dr[k].x;
}else if(_______) {
selected(r, i, j);
/*最后一边选入r成回路，完成输出结果*/
h++;
dist += dr[k].x;
}
}
}while((k != n) && (h != n));
if((h == n)) {/*最后一边选入构成回路，完成输出结果*/}
course(r, locus);
}else{/*找不到解，调整dr，交换表中边长相同的边在表中的顺序，并将b置0*/
_______;
}
}while(!b);
}
```

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。

14、[说明]

现要编写一个画矩形的程序，目前有两个画图程序：DP1和DP2，DP1用函数draw_a_line(x1, y1, x2, y2)画一条直线，DP2则用drawline(x1, x2, y1, y2)画一条直线。当实例化矩形时，确定使用DP1还是DP2。为了适应变化，包括"不同类型的形状"和"不同类型的画图程序"，将抽象部分与实现部分分离，使它们可以独立地变化。这里，"抽象部分"对应"形状"，"实现部分"对应"画图"，与一般的接口(抽象方法)与具体实现不同。这种应用称为Bridge(桥接)模式。图显示了各个类间的关系。



这样，系统始终只处理3个对象：Shape对象、Drawing对象、DP1或DP2对象。以下是C++语言实现，能够正确编译通过。

[C++代码]

```
class DP1{
public:
static void draw_a_line(double x1, double y1,double x2, double y2) {
//省略具体实现
}
};
class DP2{
public:
static void drawline(double x1, double x2, double y1, double y2) {
//省略具体实现
}
};
class Drawing{
public:
______ void drawLine(double x1,double y1,double x2,double y2)=0
};
class ViDrawing : public Drawing{
public:
void drawLine(double x1, double y1, double x2, double y2) {
DP1::draw_a_line(x1, y1, x2, y2);
}
};
class V2Drawing : public Drawing{
public:
void drawLine(double x1, double y1, double x2, double y2) {
______;
}
};
class Shape{
private:
______ _dp;
public:
Shape(Drawing *dp);
virtual void draw()  = 0;
```

```
    void drawLine(double x1, double y1, double x2, double y2);
    };
    Shape::Shape(Drawing *dp)
    {
    _dp : dp;
    }
    void Shape::drawLine(double x1, double y1, double x2, double y2)
    {    //画一条直线
    _______;
    }
    class Rectangle : public Shape{
    private:
    double _x1, _y1, _x2, _y2;
    public:
    Rectangle(Drawing *dp, double x1, double y1,
    double x2, double y2);
    void draw();
    };
    Rectangle::Rectangle(Drawing *dp, double x1, double y1, double x2, double
y2)
    : _______
    {
    _x1 = x1; _y1 = y1; _x2 = x2; _y2 = y2
    }
    void Rectangle::draw()
    {
    //省略具体实现
    }
```

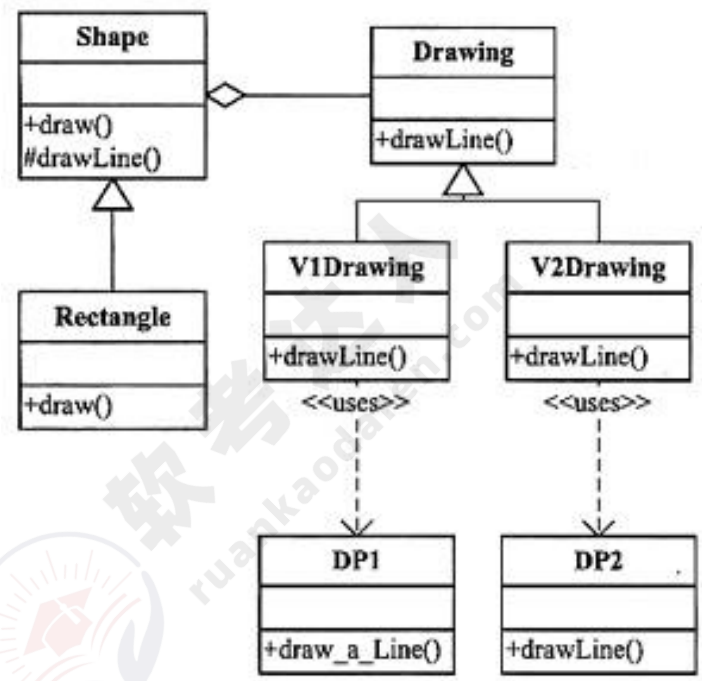
试题七

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。

15、[说明]

现要编写一个画矩形的程序，目前有两个画图程序：DP1和DP2，DP1用函数draw_a_line(x1, y1, x2, y2)画一条直线，DP2则用drawline(x1, x2, y1, y2)画一条直线。当实例化矩形时，确定使用DP1还是DP2。

为了适应变化，包括"不同类型的形状"和"不同类型的画图程序"，将抽象部分与实现部分分离，使它们可以独立地变化。这里，"抽象部分"对应"形状"，"实现部分"对应"画图"，与一般的接口(抽象方法)与具体实现不同。这种应用称为Bridge(桥接)模式。图显示了各个类间的关系。



这样，系统始终只处理3个对象：Shape对象、Drawing对象、DP1或DP2对象。以下是JAVA语言实现，能够正确编译通过。

[Java代码]

```
//DP1. java文件
public class DP1 {
static public void draw_a_line(double x1, double y1,
double x2, double y2) {
//省略具体实现
}
}
//DP2. java文件
public class DP2 {

static public void drawline(double x1, double y1,
double x2, double y2) {
//省略具体实现
}
//mrawing.java文件
_______ public class Drawing {
abstract public void drawLine(double x1, double y1, double x2, double y2);
}
//VlDrawing.java文件
public class VlDrawing extends Drawing {
public void drawLine(double x1, double y1, double x2, double y2) {
DP1.draw_a_line(x1, y1, x2, y2);
}
}
//V2Drawing.java文件
public class V2Drawing extends Drawing {
public void drawLine(double x1, double y1,
double x2, double y2) {//画一条直线
_______;
}
}
//Shape.java文件
```

```
abstract public class Shape {
abstract public void draw();
private _______ _dp;
Shape(Drawing dp) {
_dp = dp;
}
protected void drawLine(double x1, double y1,
double x2, double y2) {
_______;
}
}
//mectangle.java文件
public class Rectangle extends Shape {
private double _x1,_x2, _y1, _y2;
public Rectangle(Drawing dp,
double x1, double y1,
double x2, double y2) {
_______;
_x1 = x1; _x2 = x2;
_y1 = y1; _y2 = y2;
}
public void draw() {
//省略具体实现
}
}
```

答案：

试题一

1、成绩清单，起点：阅卷站，终点：统计成绩

对于分层数据流图，一定要注意平衡原则，即父图与子图数据流一致。

仔细与项层数据流图比对，可发现缺失了数据流“成绩清单”，其起点应为“阅卷站”，终点为加工2“统计成绩”。

2、报名单，起点：考生，终点：检查报名单

方法同问题1。可得加工1子图(图3A)中缺失了数据流“报名单”，其起点自然是“考生”，终点应为加工1.1“检查报名单”。

3、合格标准，起点：考试中心，终点：审定合格者

起点：制作通知单，终点：考生名册

方法同问题1。可得加工2子图(图3B)中缺失了数据流“合格标准”，其起点为“考试中心”，终点为加工2.2“审定合格者”。

错误数据流多半是表现在文件读出/写入上。加工2.3“制作通知单”是从“考生名册”处获得考生信息，而不是写入，因此该数据流是错误的。

试题二

4、QR：一对多，QS：一对多，RS：一对多

关系模型中实体间的联系有：一对一、一对多和多对多。一对一是指一个实体只与另一个实体相联系，一对多是指一个实体与多个实体相联系，多对多是指多个实体与多个实体间的联系。

在这个系统中，一个病人只在一个病区，一个病区有多个病人，因此联系QR是一对多联系；一个病人只有一个主治医生，一个医生显然可以医治多名病人，因此联系QS是一对多联系：一名医生只属于一个病区，一个病区有多名医生，故联系QS是一对多联系。

5、C

基本的关系代数包括并、差、广义笛卡儿积、投影、选择，其他运算可以通过基本的关系运算导出。

关系R与S具有相同的关系模式，即R与S的结构相同，关系R与S的并由属于R或属于S的元组构成的集合组成，记做R∪S，其形式定义如下：R∪S={t|t∈R˅t∈S}，式中t为元组变量。

· 并(Union)。关系R与S具有相同的关系模式，即R与S的结构相同，关系R与S的并由属于R或属于S的元组构成的集合组成，记做R∪S，其形式定义如下：R∪S={t|t∈R˅t∈S}，式中t为元组变量。

· 差(Difference)。关系R与S具有相同的关系模式，关系R与S的差由属于R但不属于S的元组构成的集合组成，记做R-S，其形式定义如下：R∪S={t|t∈R^t∉S}。

· 广义笛卡儿积(Extended Cartesian Product)。两个元组分别为n目和m目的关系R和S的笛卡儿积是一个n+m列的元组的集合，元组的前n列是关系R的一个元组，后m列是关系S的一个元组，记做R×S，其形式定义如下：R×S={t|t=<t_n，t_m>^t^n∈R^t^m∈S}。如果R和S中有相同的属性名，那么可在属性名前加关系名作为限定，以示区别。若R有k1个元组，S有k2个元组，则R和S的广义笛卡儿积有k1×k2个元组。

· 投影(Projection)。投影运算是从关系垂直方向进行运算的，在关系R中选择出若干属性列A组成新的关系，记做π_A(R)，其形式定义如下：π_A(R)={f[A]|t∈R}。

· 选择(Selection)。选择运算是从关系的水平方向进行运算的，从关系R中选择满足给定条件的诸元组，记做σ_F(R)，其形式定义如下：σ_F(R)={t|t∈R^F(r)=True}。其中，F中的运算对象是属性名(或列的序号)或常数。运算符是算术比较符(<，≤，>，≥，=，≠)和逻辑运算符(^，˅，¬)。

在此主要涉及投影和选择，根据语义，(1)中"外科"与"内科"应为或关系，且应先选择再投影，因为作投影运算之后，选择操作涉及的列已经不在了，故为C；(2)中"内科"和"胃病"应为与关系，同样应该先选择再投影，故为D。

6、D

7、层次模型采用树型结构表示数据与数据间的联系。在层次模型中，每一个节点表示记录类型(实体)，记录之间的联系用节点之间的连线表示，并且根节点以外的其他节点有且仅有一个双亲节点。层次模型不能直接表示多对多联系，若要表示多对多的联系，可采用如下两种方法：

· 冗余节点法—两个实体的多对多的联系转换为两个一对多的联系，该方法的优点是节点清晰，允许节点改变存储位置，缺点是需要额外的存储空间，有潜在的数据不一致性。

· 虚拟节点分解法—将冗余节点转换为虚拟节点，虚拟节点是一个指引元，指向所代替的节点，该方法的优点是减少对存储空间的浪费，避免数据不一致性，缺点是改变存储位置可能引起虚拟节点中指针的修改。

试题三

8、A："客户"　　B："发短信"　　C："管理电话簿"

图给出了系统用例图，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。

从说明易知参与者A是"用户"。

仔细分析，缺少的用例为"发短信"和"管理电话簿"，而"发短信"与"无线电网络"相关，故用例B为"发短信"，用例C为"管理电话簿"。

9、(A)"按数字键"(B)"连接基站"(C)"按断线键"(D)"断开连接"

根据题意，打电话的流程如下：

①用户拨电话号码，每按下一个数字键，显示屏上显示相应数字；

②按OK键进行连线，显示屏上显示"连线中…"，请求连接基站，基站通过移动电话网络连接到对方手机，若有误则返回相关信息；

③接通后，显示屏显示"连线成功"；

④打电话结束后，按Cancel送出断线信号，通知移动电话基站断线，基站切断连接，显示屏显示"断线成功"。

对比可得，(A)为"按数字键"，(B)为"连接基站"，(C)为"按断线键"，(D)为"断开连接"。

试题四

10、状态1：运行态，状态2：就绪态，状态3：等待态。

根据问题1中对进程3种状态的描述，易于判断状态1：运行态，状态2：就绪态，状态3：等待态。

11、运行进程最多1个，最少0个；就绪进程最多N-1个，最少0个；等待进程最多N个，最少0个。

问题1给出了三种状态的具体表现形式。对于单CPU系统，运行的进程最多只有1个，最少可以是0个(当所有进程都处于阻塞态时)。就绪进程最多只可能有N-1个，因为有就绪进程的话，肯定有运行进程，最少0个。等待进程最多可有N个，最少可为0个(1个运行，N-1个就绪)。

12、P2(1)、P5(2)、P1(4)、P3(2)、P5(3)、P4(1)、P1(6)。括号内数字表示该进程还需的执行时间。

根据题意，开始调度前，各个级别队列为：

- 优先数1：P2(1)，时间片为1单位；
- 优先数2：P5(5)，时间片为2单位；
- 优先数3：P1(10)、P3(2)，时间片为4单位；
- 优先数4：P4(1)，时间片为8单位。

根据调度策略"系统总是先调度级别较高的队列中的进程，仅当该队列为空时才去调度下一级队列中的进程；当执行进程用完其时间片时便被剥夺并进入下一级就绪队列"，系统先训度P2进程，执行1单位时间，时间片到，P2亦执行完毕，各个级别队列为：

- 优先数1：时间片为1单位；
- 优先数2：P5(5)，时间片为2单位；
- 优先数3：P1(10)、P3(2)，时间片为4单位；
- 优先数4：P4(1)，时间片为8单位。

系统调度P5进程，执行2单位时间，进程P5还需3单位时间，进入优先数3队列，各个级别队列为：

- 优先数1：时间片为1单位；
- 优先数2：时间片为2单位；
- 优先数3：P1(10)、P3(2)、P5(3)，时间片为4单位；
- 优先数4：P4(1)，时间片为8单位。

系统调度P1进程，执行4单位时间，进程P1还需6单位时间，进入优先数4队列；继续调度P3进程，执行2单位时间，进程P3执行完毕；训度进程P5，执行3单位时间，执行完毕，各个级别队列为：

- 优先数1：时间片为1单位；
- 优先数2：时间片为2单位；
- 优先数3：时间片为4单位；
- 优先数4：P4(1)、P1(6)，时间片为8单位。

系统调度P4进程，执行1单位时间，进程P4执行完毕；继续调度P1进程，执行6单位时间，进程P1执行完毕。

至此，可得五个进程的CPU占用序列以及其占用时间。P2(1)、P5(2)、P1(4)、P3(2)、P5(3)、P4(1)、P1(6)。

试题五

13、distance(pd[i],pd[j])

!isCircuit(r,i,j)

selected(r,i,j)

h==n-1

exchange(dr,m,b)

试题六

14、virtual

DP2::drawline(x1,x2,y1,y2)

```
Drawing
_dp->drawLine(x1,y1,x2,y2)
Shape(dp)
```

试题七

```
15、abstract
DP2.drawline(x1,x2,y1,y2)
Drawing
_dp.drawLine(x1,y1,x2,y2)
super(dp)
```