C++实现电梯调度系统

一、背景

随着经济的不断发展，越来越多的摩天大楼拔地而起，而电梯作为高层建筑物种的运送人员货物的设备也越来越被广泛使用。电梯的运行是电梯与大楼的各个楼层之间的使用者进行交互的一个过程，对于电梯的模拟，就是对这一交互过程的一个模拟。以本校七教办公楼为例，有着12层楼，配有两部电梯，在每一层楼中还有着可以呼叫电梯的上下两个按钮。本次课程设计使用C++面向对象的程序设计语言来实现对电梯运行的一个模拟，而面向对象的程序设计方法是将事物抽象为对象，对象有着自己的行为和属性，并通过对消息的反映来完成一定的任务。

二、需求

功能需求：用面向对象技术来实现单部电梯或者多部电梯的模拟运行软件。

电梯调度策略：**顺便服务策略。**

这种策略在运行控制中所规定的安全前提下，一次将一个方向上的所有呼叫和目标全部完成。然后掉转运行方向完成另外一个方向上的所有呼叫和目标。

任务说明：

要求根据下面的功能说明描述实现模拟电梯控制软件

1. 电梯配置

（1） 共有 1 个电梯

（2） 共有 maxfloor 层楼层，这里 maxfloor 暂时取做 9。

（3） 中间层每层有上下两个按钮，最下层只有上行按钮，最上层只有上行按钮。每层都有相应的指示灯，灯亮表示该按钮已经被按下，如果该层的上行或者下行请求已经被响应，则指示灯灭

（4） 电梯内共有 maxfloor 个目标按钮，表示有乘客在该层下电梯。有指示灯指示按钮是否被按下。乘客按按钮导致按钮指示灯亮，如果电已经在该层停靠则该按钮指示灯灭

（5） 另有一启动按钮（GO）。当电梯停在某一楼层后，接受到 GO信息就继续运行。如果得不到 GO 信息，等待一段时间也自动继续运行。

（6） 电梯内设有方向指示灯表示当前电梯运行方向。

2. 电梯的运行控制

（1） 电梯的初始状态是电梯位于第一层处，所有按钮都没有按下。

（2） 乘客可以在任意时刻按任何一个目标钮和呼叫钮。呼叫和目标

对应的楼层可能不是电梯当前运行方向可达的楼层。

（3） 如果电梯正在向 I 层驶来，并且位于 I 层与相邻层（向上运行时是 I–1 层或者向下运行时是 I+1 层）之间，则因为安全考虑不响应此时出现的 I 层目标或者请求。如果电梯正好经过了 I 楼层，运行在 I 楼层和下一楼层之间，则为了直接响应此时出现的 I 层目标或者请求，必须至少到达运行方向上的下一楼层然后才能掉头到达 I 楼层（假设掉头无须其额外时间），如果 I 楼层不是刚刚经过的楼层则可以在任意位置掉头，此时

掉头后经过的第一个楼层不可停。

（4） 电梯系统依照某种预先定义好的策略对随机出现的呼叫和目标进行分析和响应。

（5） 乘客数量等外界因素（可能导致停靠时间的长短变化）不予考虑。假设电梯正常运行一层的时间是 5S，停靠目标楼层、上下乘客和电梯继续运行的时间是 5S。

（6） 当电梯停靠某层时，该层的乘客如果错误的按目标或呼叫按钮

都不予响应。

（7） 电梯停要某一层后，苦无目标和呼叫，则电梯处于无方向状态，方向指示灯全灭，否则电梯内某个方向的指示灯亮，表示电梯将向该方向

运行。等接到“GO”信号后电梯立即继续运行。若无 GO 信号，则电梯在等了上下乘客和电梯继续运行时间后也将继续运行。

（8） 当一个目标(呼叫)已经被服务后，应将对应的指示灯熄灭。

3. 电梯运行的控制策略

顺便服务策略：

顺便服务是一种最常见的简单策略。这种策略在运行控制中所规定的安全前提下，一次将一个方向上的所有呼叫和目标全部完成。然后掉转运行方向完成另外一个方向上的所有呼叫和目标。可以采用设定目标楼层的办法来实现这个策略，即电梯向一个目标楼层运行，但这个楼层可以修改。具体策略如下：

修改目标楼层的策略：

a．如果电梯运行方向向上，那么如果新到一个介于当前电梯所处楼层和目标楼层之间，又可以安全到达的向上呼叫或者目标，将目标楼层修改为这个新的楼层。

b．如果电梯运行方向向下，那么如果新到一个介于当前电梯所处楼层和目标楼层之间，又可以安全到达的向下呼叫或者目标，将目标楼层修改为这个新的楼层。

确定新的目标楼层：

如果电梯向上运行，当它到达某个目标楼层后，则依照以下顺序确定下一个目标楼层：

a．如果比当前层高的楼层有向上呼叫或者目标，那么以最低的高于当前楼层的有向上呼叫或者目标的楼层为目标。

b．如果无法确定目标楼层，那么以最高的向下呼叫或者目标所在楼层为电梯当前目标楼层。

c．如果无法确定目标楼层，那么以最低的向上呼叫所在楼层为电梯当前的目标楼层。

d．如果仍然不能确定目标楼层（此时实际上没有任何呼叫和目标），那么电梯无目标，运行暂停。

如果电梯向下运行，依照以下顺序确定下一目标楼层：

a．如果比当前层低的楼层有向下呼叫或者目标，那么以最高的低于当前楼层的有向下呼叫或者目标的楼层为目标。

b．如果无法确定目标楼层，那么以最低的向上呼叫或者目标所在楼层为电梯当前目标楼层。

c．如果无法确定目标楼层，那么以最高的向下呼叫楼层为目标楼层。

d．如果仍然不能确定目标楼层（此时实际上没有任何呼叫和目标），那么电梯无目标，运行暂停。

三、模拟电梯的功能

根据实际功能分析，模拟电梯具有复杂的状态和交互作用，众多的事件将引起状态的复杂变化。电梯状态如下：电梯向上运行状态、电梯向下运行状态、电梯停止状态。 定义为int run_status1向上，2向下,0停止。按钮有分成两种一种是电梯外的请求电梯赶来的按钮，另外一种则是电梯内发出所要到达楼层的按钮。

基于以上的分析，把电梯的状态做出如下说明。

（1）电梯初始时刻在一层最初始的运行方向只可能是向上走。

（2）电梯启动后不管现在所在是第几层，电梯所要做的就是根据自己目前的运行方向往上查询是否有电梯外楼层所发出的请求，而对于向下运行方向的请求先不去理会，保存起来。

（3）电梯的请求有两种，电梯外所在楼层发出的要电梯赶来的请求和电梯内发出的所要到达楼层的请求，这也正好与电梯系统的两种按钮所对应。假设电梯正在往4楼向上运行，此时电梯要查询4楼以上的楼层是否还有人要上楼或者下楼，若没有人要上楼或下楼，则要查询4楼以下是否有人要上下楼，如果有则应该改变电梯运行方向，向下运行去接4楼下发出请求的那一个乘客。

（4）对于这两种请求，可以使用两个数组来储存。使用数组储存的原因是因为，大楼和数组有着相识的特点，都是连续的，这样就可以使用数组的下标来表示大楼的楼层。

A.对于第一种请求电梯外所在楼层所发出的请求 定义为int eout[10]，最开始初始化为0。若eout[i]=1表示第i层有乘客发出了一个向上的请求，若eout[i]=2表示第i层有乘客发出了一个向下的请求，若eout[i]=3表示第i层既有乘客发出一个向下运行的请求，又有乘客发出了一个向上运行的请求。这里我们也可以联系实际，在电梯还没有到发出请求的楼层接乘客时，是没有办法知道到底有多少个乘客要上电梯的，只有乘客进入电梯之后才知道。

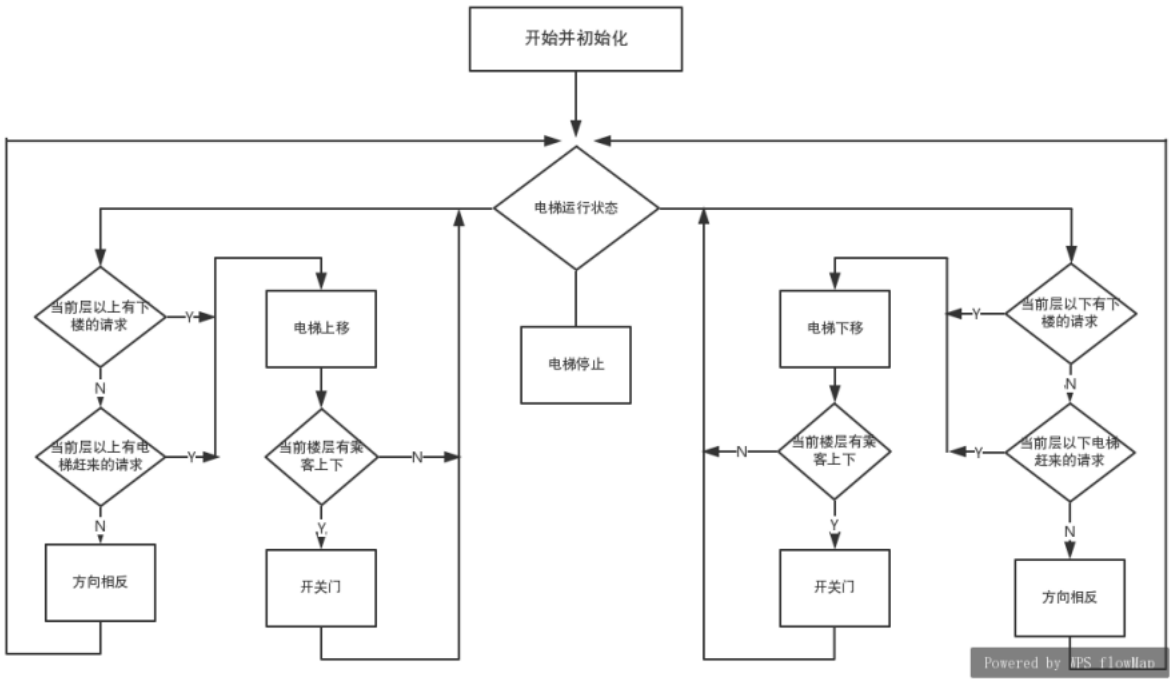
B.对于第二种请求电梯内发出的所要到达楼层的请求， 定义为int floor[10]，最开始初始化为0。若floor[i]=1表示到达第i层时，有乘客到达了目的楼层，需要下电梯。

（5）对应的内部电梯按钮，按下后就会调用相对应的set_floor函数把对应的数组元素修改为1。对于上下按钮，按下后调用对应的set_out函数，若是向上把对应的数组元素修改为1，若是向下把对应的数组元素修改为2，若是既有向上又有向下把对应的数组元素修改为3。

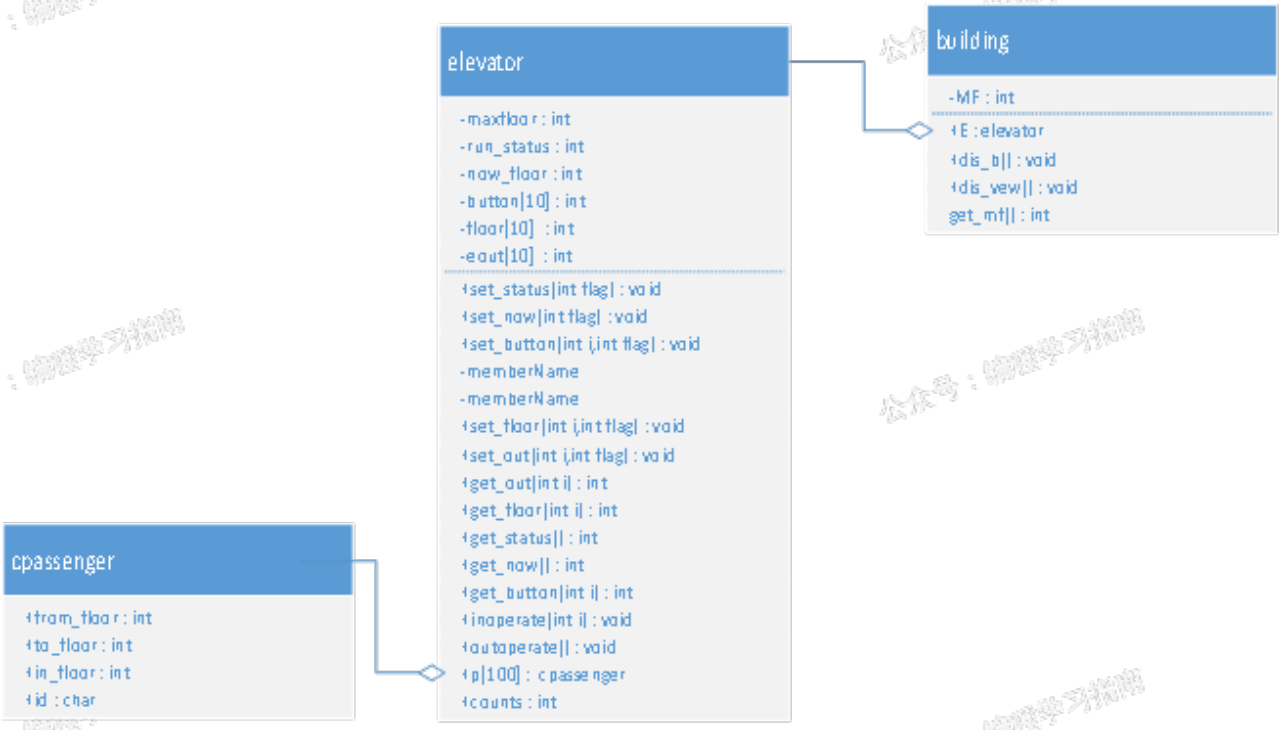
（6）按照两种请求的不同，将两种按钮依照电梯内输入和电梯外输入封装为两个函数 ： 电梯内输入void inoperate(int i)，电梯外输入void outoperate()。

四、电梯系统流程图

使用流程图来描述电梯控制系统的主程序流程：



UML类图：



五、相关类的实现

电梯的运行是一个电梯与大楼各个楼层的使用者之间交互的过程，在写类定义的过程中，要注意函数及变量的命名要符合其含义或功能。根据自顶向下的程序设计原则，然后可以根据规划编写类的定义。

（1）本程序第一个主要的类名为elevator的电梯类。其实现如下：

```
class elevator
{
public:
    elevator(int mf=9);//构造函数，默认9层
    ~elevator();
    void set_status(int flag);//设置电梯当前的运行状态
    void set_now(int flag);//设置当前电梯所在的楼层
    void set_button(int i,int flag);//设置电梯内按钮的明灭
    void set_floor(int i,int flag);//1有，0没有
    void set_out(int i,int flag);//1向上走，2向下走.3既有向上又有向下
    int get_floor(int i);//获得第i层是否有乘客到达的消息
    int get_out(int i);//获得第i层的请求消息
    int get_status();//获得电梯当前的状态
    int get_now();//获得电梯当前的位置
    int get_button(int i);
    void inoperate(int i);//电梯内部输入
    void outoperate();
    cpassenger p[100];//要乘坐电梯的乘客
    int counts;//乘坐该电梯乘客数
private:
    int maxfloor;//共有maxfloor层
    int run_status;//电梯运行状态，1向上，2向下,0停止
    int now_floor;//电梯当前所在楼层
    int button[10];//楼层按钮，1按下，0未按
    int floor[10];//消息队列,静态数据成员
    int eout[10];//标记来自电梯外所在的楼层
};
```

电梯中乘客类 cpassenger，我们需要知道乘客从哪一楼层上的电梯，要在哪一楼层下电梯，同时还要知道现在是否还在电梯上。

```
class cpassenger
{
public:
    int from_floor;//所在楼层
    int to_floor;//要去楼层
    int in_floor;//是否在电梯中,1在，0不在
    char id;//编号
};
```

（2）最后需要一个building类，电梯类要安装在building中，同时building还要有模拟的电梯内外部的指示灯和电梯时刻运行状态。

```
class building
{
public:
    building(int mf=9);
    ~building();
    elevator E;
    void dis_b();//打印模拟的电梯时刻运行
    void dis_vew();//打印电梯内部以及电梯内部的按钮
    int get_mf();
private:
    int MF;//9层楼
};
```

六、详细设计

写好类定义，接下来就要给每个函数写函数定义，这就涉及到了电梯模型的具体算法。设计的电梯模型是这样的。

（1）**电梯类的构造函数**

```
elevator::elevator(int mf)
{
    int i;
    maxfloor=mf;
    run_status=0;//初始向上运行
    now_floor=1;//电梯初始状态在第一层
    counts=1;
    for(i=1; i<=mf; i++)//消息数组的初始化
    {
        button[i]=0;
        floor[i]=0;
        eout[i]=0;
    }
}
```

电梯默认有9层，电梯起始方向是向上的run_status=0，起始处在1层。通过一个for循环将1到9层的请求数组队列和电梯内的要到达的数组队列都清为0。

（2）**上面说到的两个电梯内部和电梯外部按按钮的两个函数。**

```
void elevator::inoperate(int i)
{
    int fl;
    cout<<”请进到电梯内乘客按下所要到达的楼层！”<<endl;
    cout<<”多个楼层中间用空格隔开，结束输入请按0：”;
    while(1)
    {
        cin>>fl;
        if(fl==0)
        {
            break;
        }
        p[counts].from_floor=i;//乘客来自的楼层
        p[counts].to_floor=fl;//要去的楼层
        p[counts].in_floor=1;//在电梯中
        p[counts].id=’A’+counts-1;//给乘客编号
        counts++;
        elevator::set_floor(fl,1);//要到达的楼层
        set_button(fl,1);//设置电梯内的显示按钮
    }
}
```

乘客进入电梯内按下按钮的同时，乘客类 cpassenger就会捕获乘客的相关信息，乘客来自的楼层、乘客要去的楼层、乘客是否还在电梯中、乘客获得的编号，同时floor数组标记要到达的楼层，电梯内相应的指示灯亮起来。

```
void elevator::outoperate()//在电梯外部发送请求时，还没有乘客上电梯
{
```

```
    int fl;
    cout<<”请电梯外部乘客按下所在楼层的上楼信号！”<<endl;
    cout<<”多个楼层用空格隔开，结束输入请按0：”;
    while(1)
    {
        cin>>fl;
        if(fl==0)
        {
            break;
        }
        if(elevator::get_out(fl)==2)//若已经有了向下的信号
        {
            elevator::set_out(fl,3);
        }
        else
        {
            elevator::set_out(fl,1);//1向上走
        }
    }
    cout<<”请电梯外部乘客按下所在楼层的下楼信号！”<<endl;
    cout<<”多个楼层用空格隔开，结束输入请按0：”;
    while(1)
    {
        cin>>fl;
        if(fl==0)
        {
            break;
        }
        if(elevator::get_out(fl)==1)//若已经有了向上的信号
        {
            elevator::set_out(fl,3);
        }
        else
        {
            elevator::set_out(fl,2);//2向下走
        }
    }
}
```

电梯外部乘客的请求则要分为上楼elevator::set_out(fl,1)和下楼elevator::set_out(fl,2)，如果该楼层同时有乘客上楼和下楼的请求，那么我们就要设置elevator::set_out(fl,3)。

七、例举其中的算法

判断电梯是否需要转变方向的算法：

这里使用电梯运行状态从向上运行转变向下运行的一段程序进行说明。电梯要转变运行状态向下运行，那么一定是当前电梯内没有乘客要到达上方的楼层，上方也没有呼叫电梯的请求，但是下方要有呼叫电梯的请求，否则电梯就会停止了。

```
flag1=0;//1电梯运行向上
flag2=0;//1电梯运行向下
for(i=cur; i<=mf; i++)//当前电梯所在楼层以上是否有乘客要到达或者是否有请求
{
```

```
    if(A.E.get_floor(i)==1||A.E.get_out(i)!=0)//上方有请求
    {
        flag1=1;
        break;
    }
}
for(j=1; j<=cur; j++)//当前电梯所在楼层以下是否有乘客要到达或者是否有请求
{
    if(A.E.get_floor(j)==1||A.E.get_out(j)!=0)//下方有请求
    {
        flag2=1;
        break;
    }
}
cnt=0;//计数器
for(i=1; i<=mf; i++)//整栋大楼中是否还有请求或者还有人没有送达
{
    if(A.E.get_floor(i)==0&&A.E.get_out(i)==0)
    {
        cnt++;
    }
}
if(cnt==mf)//电梯里没有人了，大楼中也没有请求
{
    A.E.set_status(0);
    break;
}
if(flag1==1)
{
    A.E.set_status(1);//电梯向上
}
else if(flag2==1)//电梯向下
{
    A.E.set_status(2);//电梯向下
    break;
}
```

上述算法通过三个循环来遍历整座大楼的各个楼层的请求和电梯内乘客要去的楼层，时间复杂度为O(n)。至于电梯从向下运行转变为向上运行的过程则与本算法相同。

八、存在的问题与不足及对策

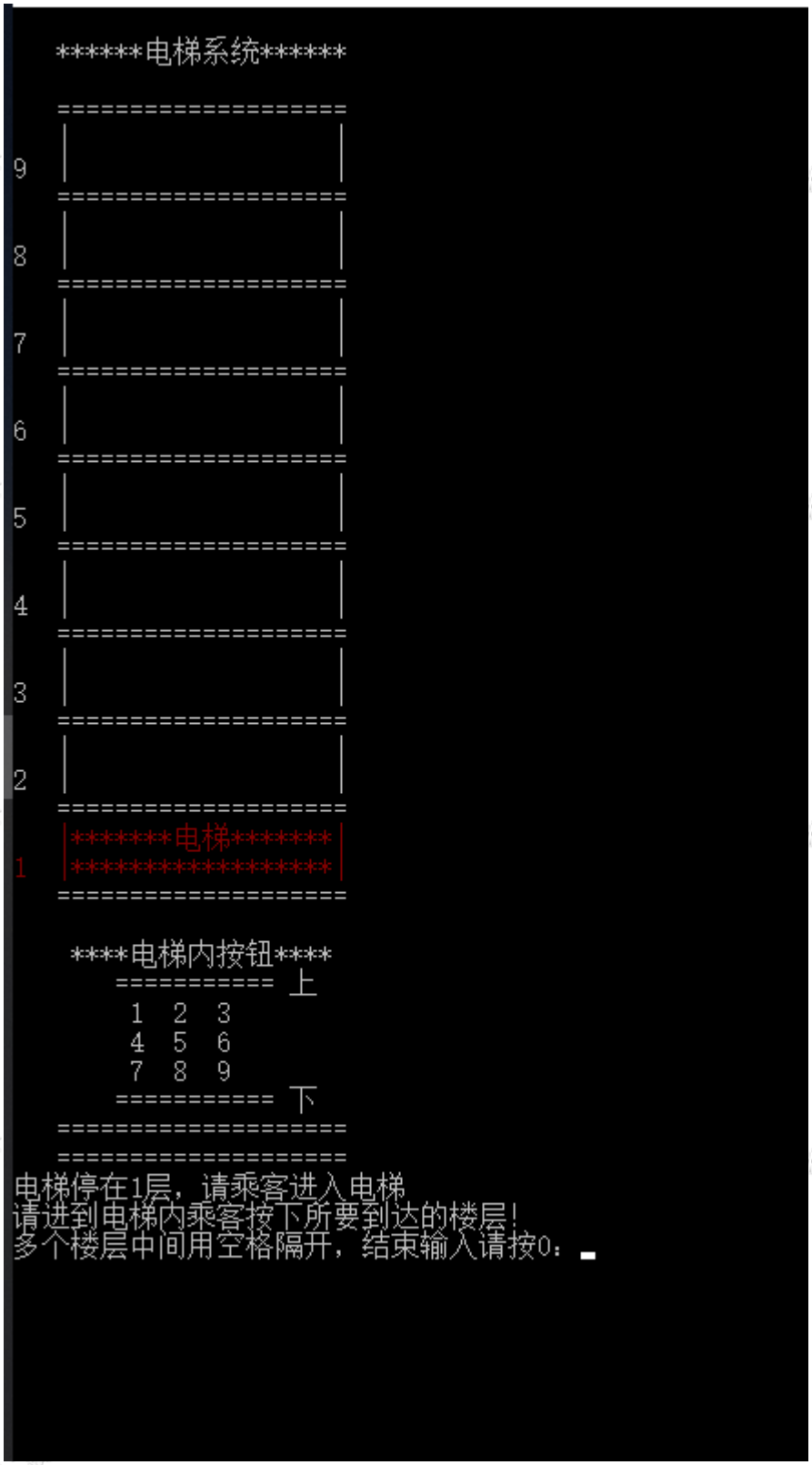
1.由于本程序采用的是顺便服务的电梯调度策略，所以可能会存在着部分用户等候电梯时间过长，刚好错过电梯之后需要等候好久的问题，而这主要是由电梯的调度策略决定的，不能很好解决。联系生活实际，我们平常电梯系统就是这样实现的，但对于一般的大楼会设置多台电梯，对于本程序如果采用多台电梯，那么主控制程序就需要进行很多的判断，会造成程序结构复杂的情况，一个很好的解决方式是选择多线程，通过多线程技术，实现多台电梯同步运行同时修改接受消息队列数组中的信息，电梯内部各有各自要到达楼层的请求，互补影响。

2.本程序由于时间仓促以及作者本人水平受限，未能配置上图形化的操作界面，只是借用了编译器的命令台，影响了使用的观感和体验。

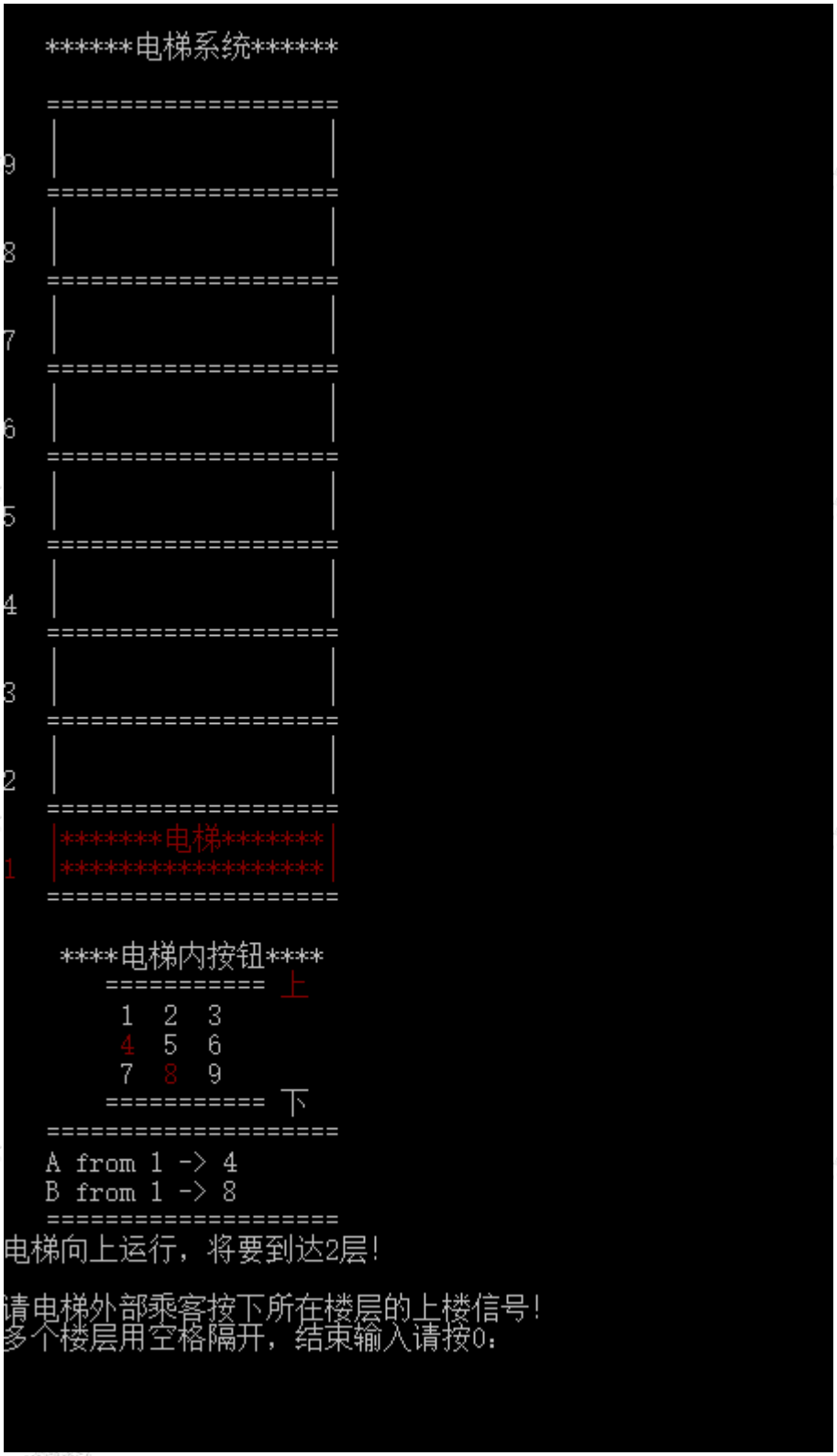
九、程序使用说明

该楼有9层，电梯一开始停在1层。该电梯系统主要有两个界面构成，一个是显示当前电梯所在楼层的信息，另一个界面显示电梯内部的信息。电梯当前在那一层，那一层就会用红色显示。如果电梯外部的乘客想要进电梯会在所在层发出请求，将按照向上或向下来分类打印请求。

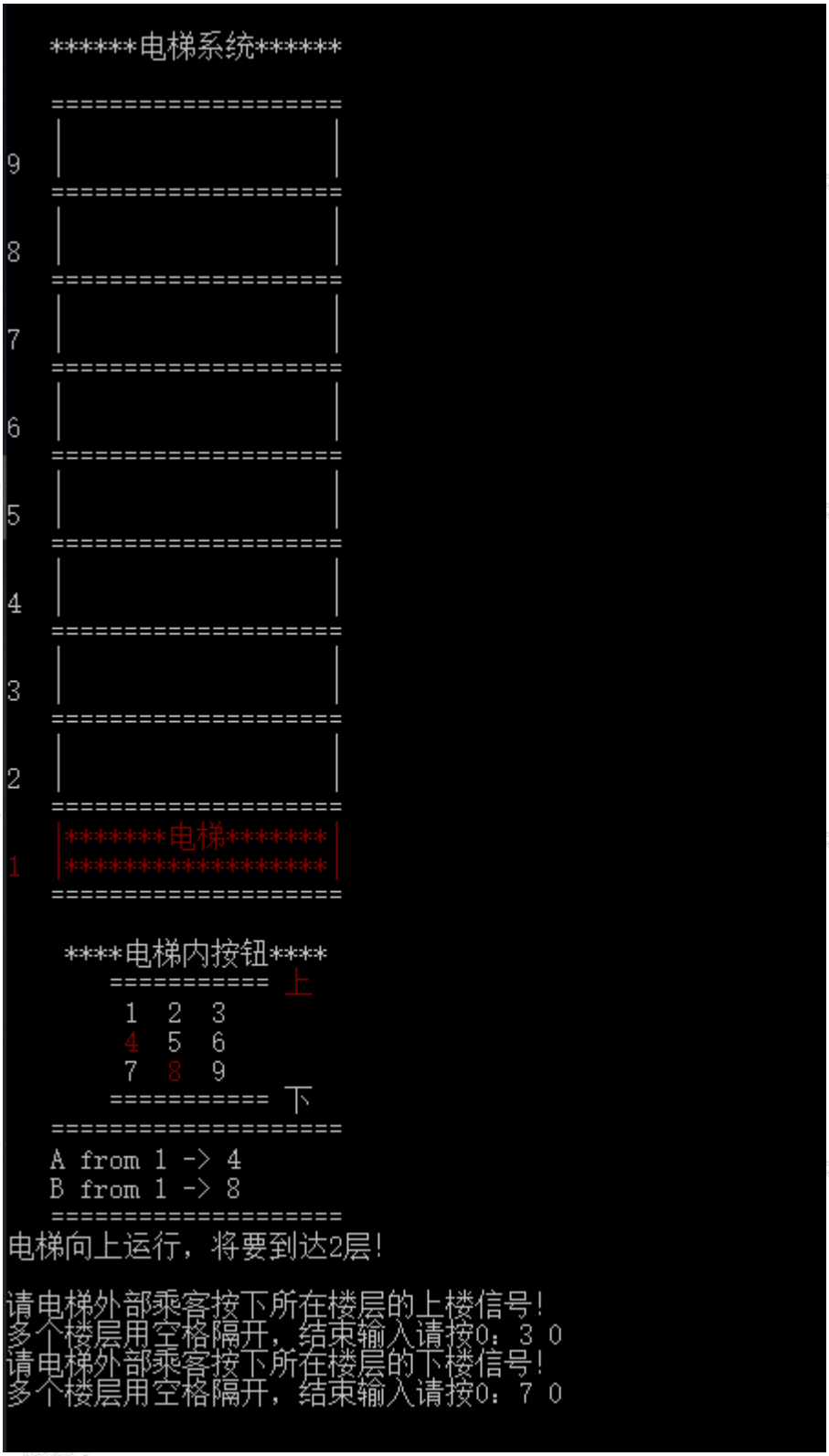
进入电梯后的乘客按下要去的楼层按钮，该按钮就会变为红色，直到该乘客到达要去的楼层后，按钮才会再次变为无色。同时显示电梯内部的界面也会显示当前电梯的运行状态，若是向上运行，‘上’为红色；向下运行，‘下’为红色；若电梯停止，都不为红色。为了显示电梯内的成员信息，该程序会按照上电梯的顺序为每一个乘客按字母编号，同时打印该乘客的信息，如“ A from 1 –> 3”表示A号乘客从1层上电梯要去3层。



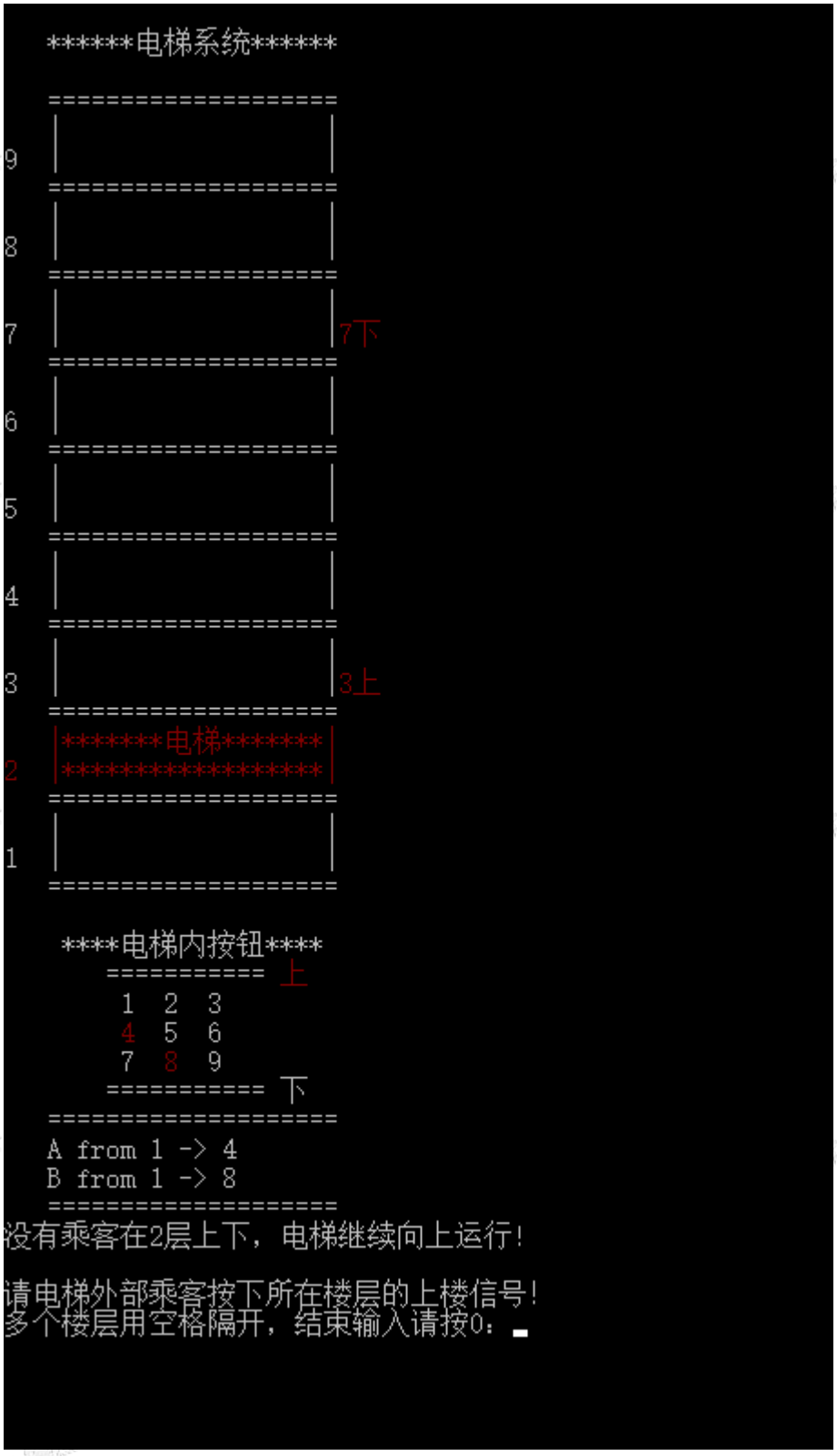
（1）在1楼上来两位乘客A、B分别要去4和8楼，输入4 8 0（0表示输入结束）。



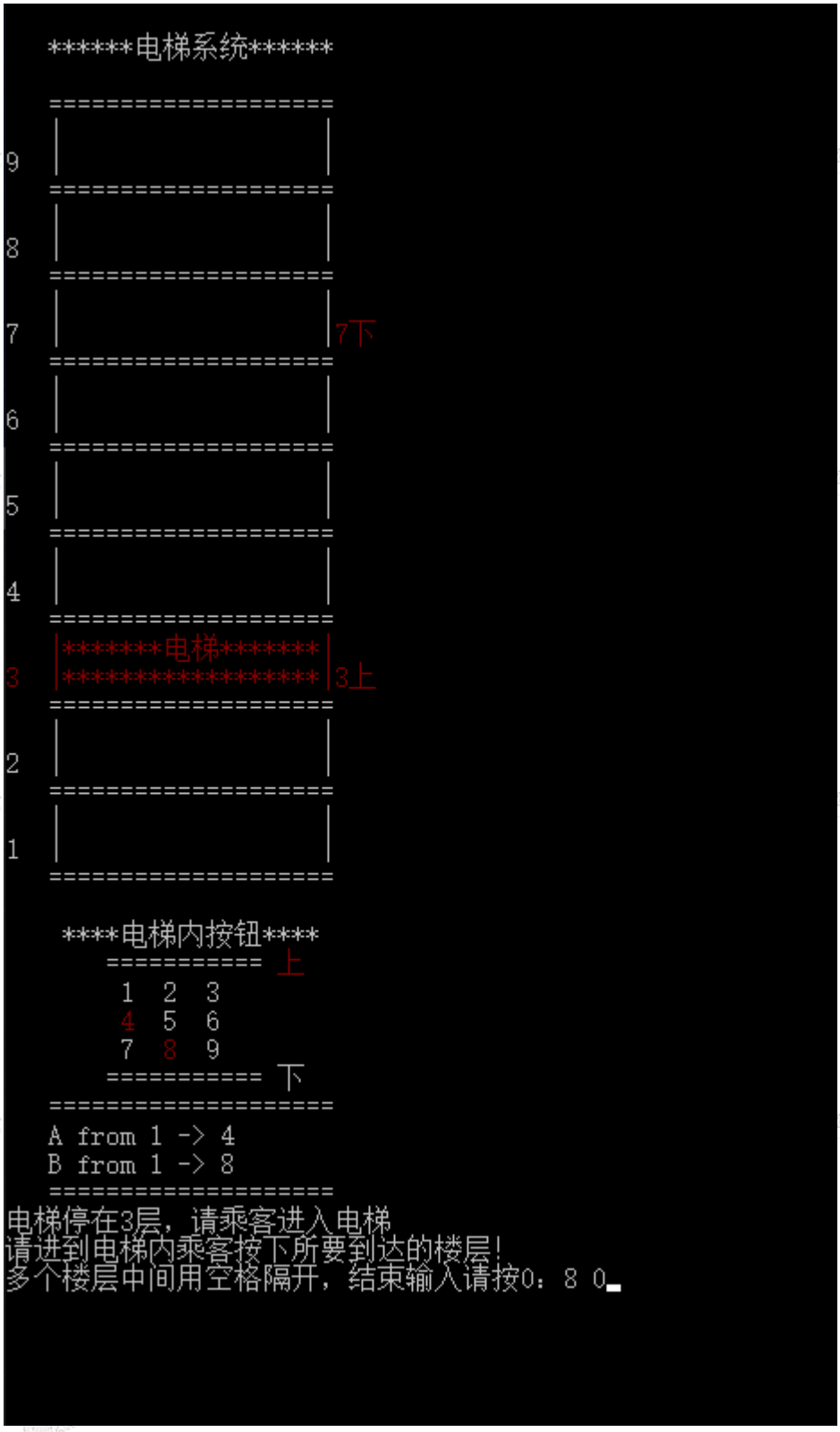
（2）电梯开始了向上运行，电梯内4层和8层的指示灯变红，这是在第3层有人发出一个向上的请求，在第8层有人发出了一个向下的请求。



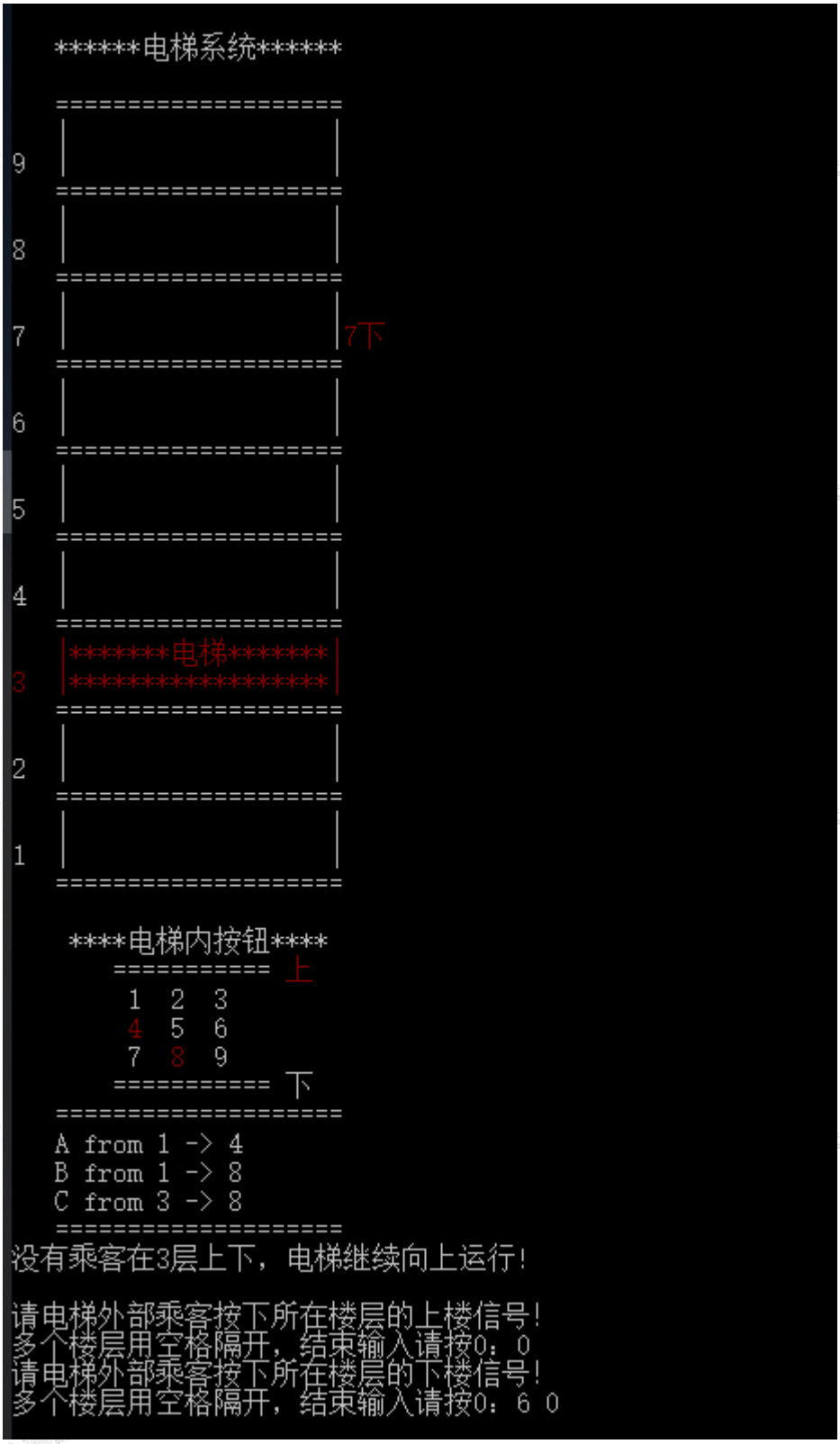
(3) 没有乘客在第二层下电梯，电梯继续向上走。电梯里也没有人发出向上或者向下的请求。



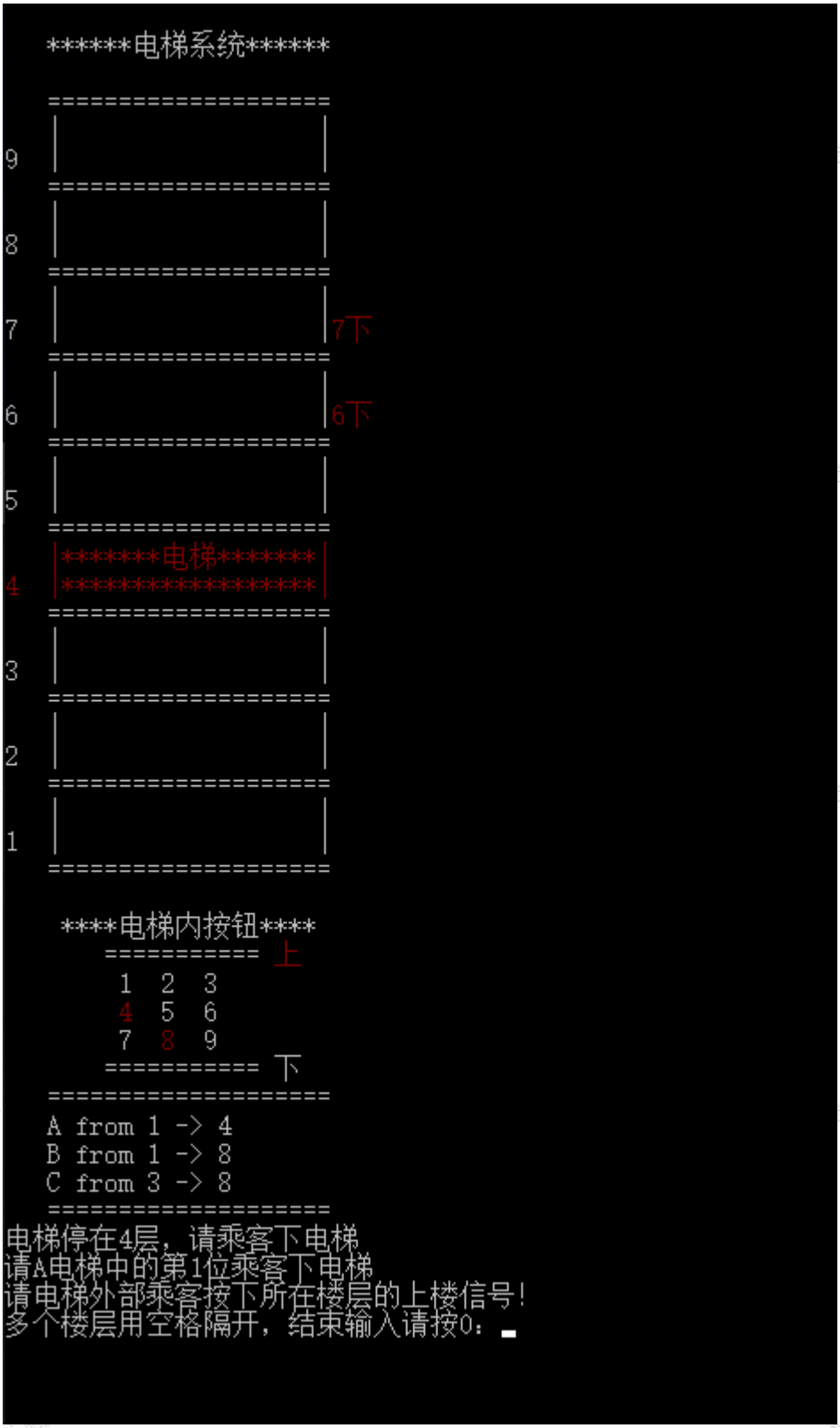
（4）电梯到达第3层，乘客C进入电梯中，按下8层的按钮。



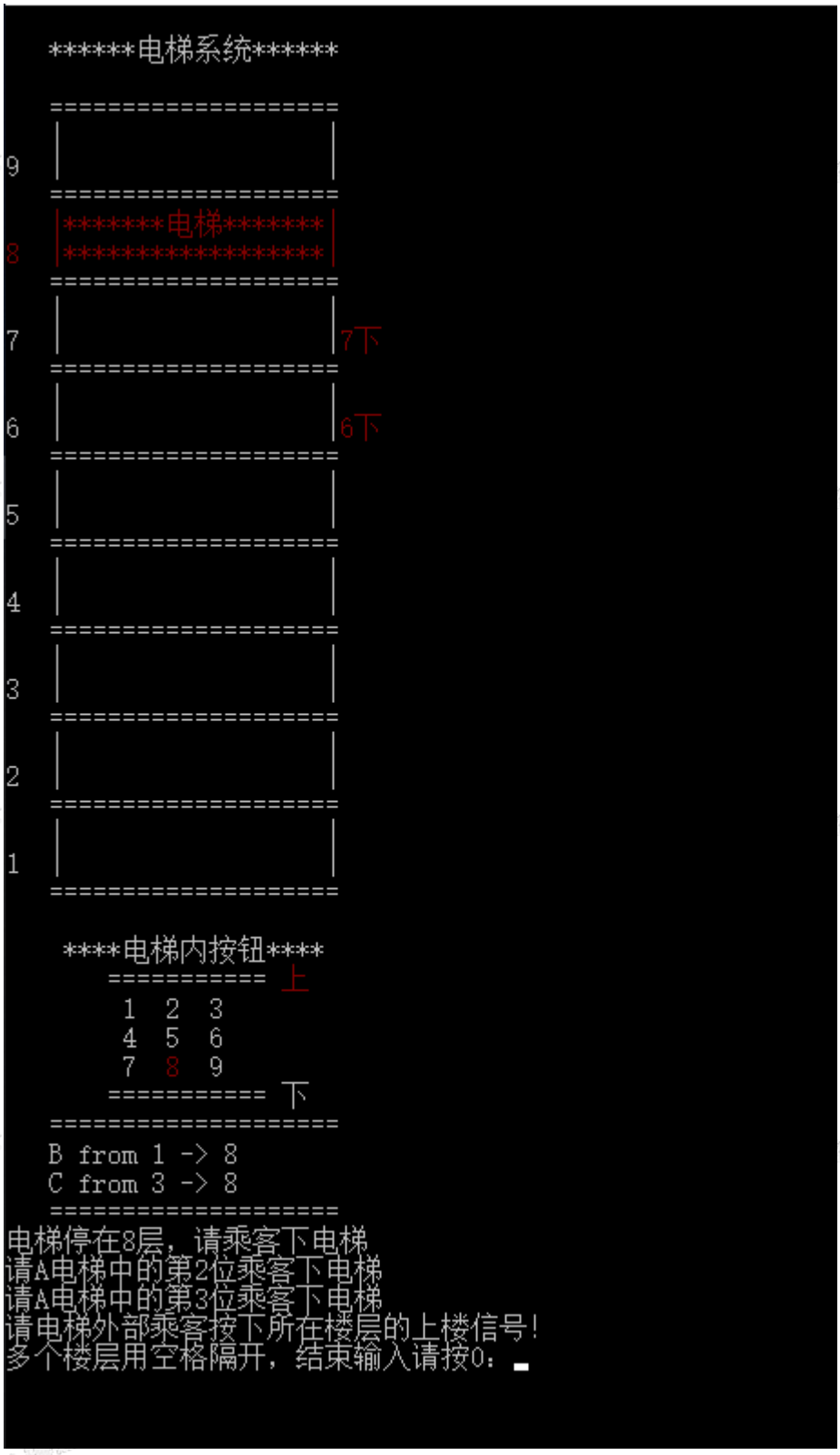
(5) 电梯还在3层时，6楼有一个向下的请求。



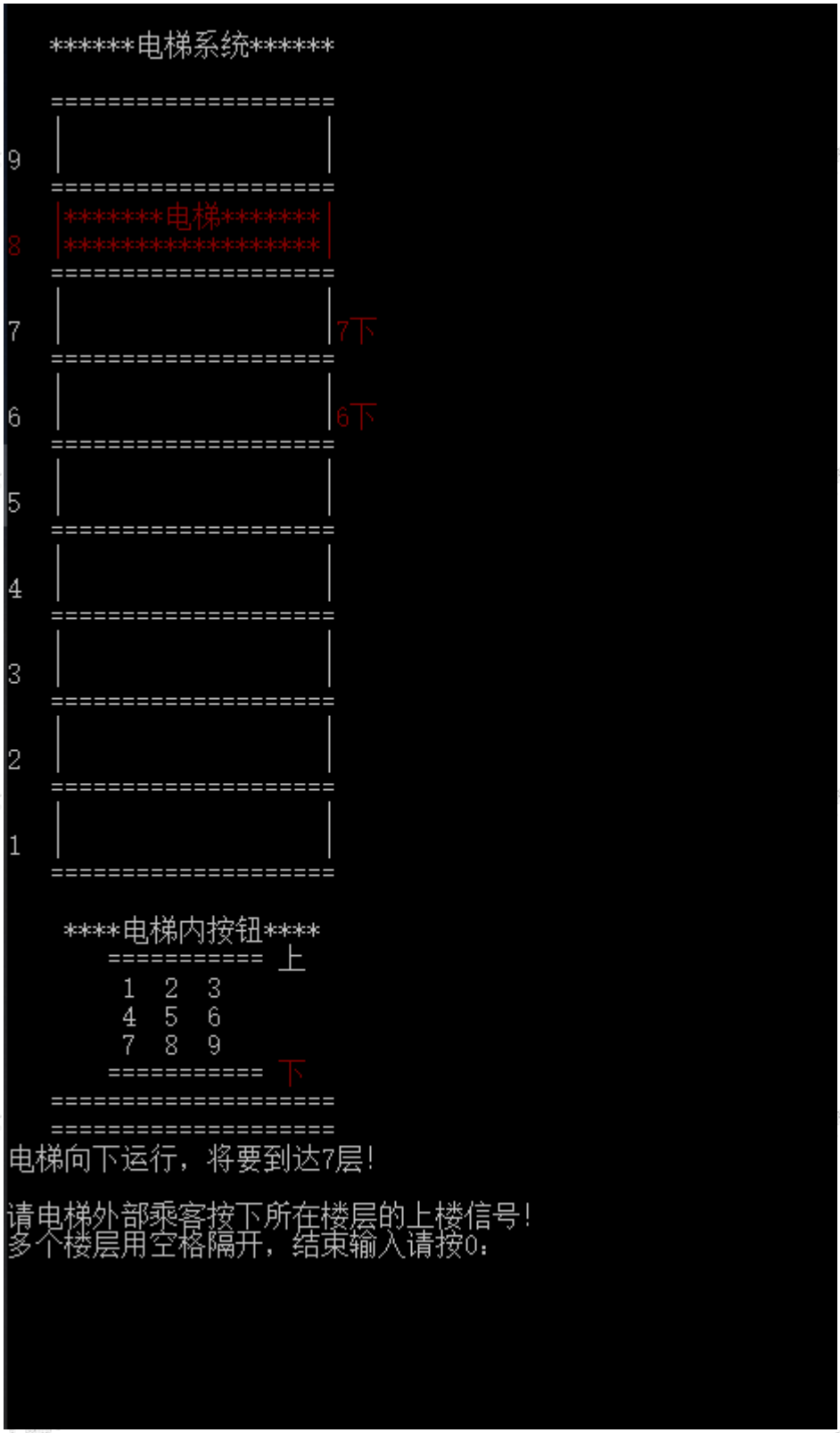
（6）电梯到达第4层时，第一位上电梯的乘客A下了电梯。



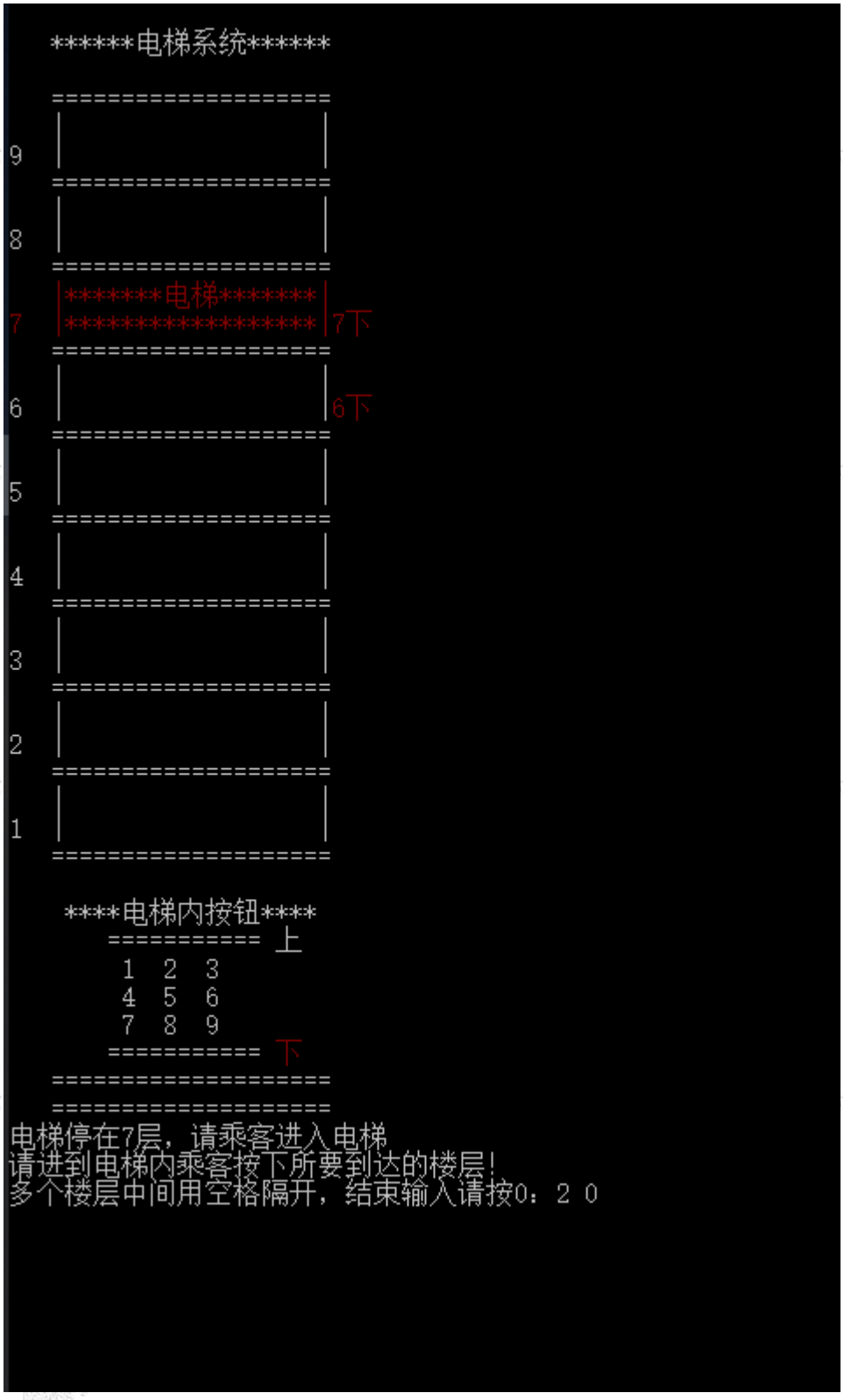
（7）从4楼到7楼大楼里一路都没有向上或向下的请求，电梯持续向上运行，到第8层时，电梯中的第2位和第3位乘客下了电梯。此时也没有向上或者向下的请求。



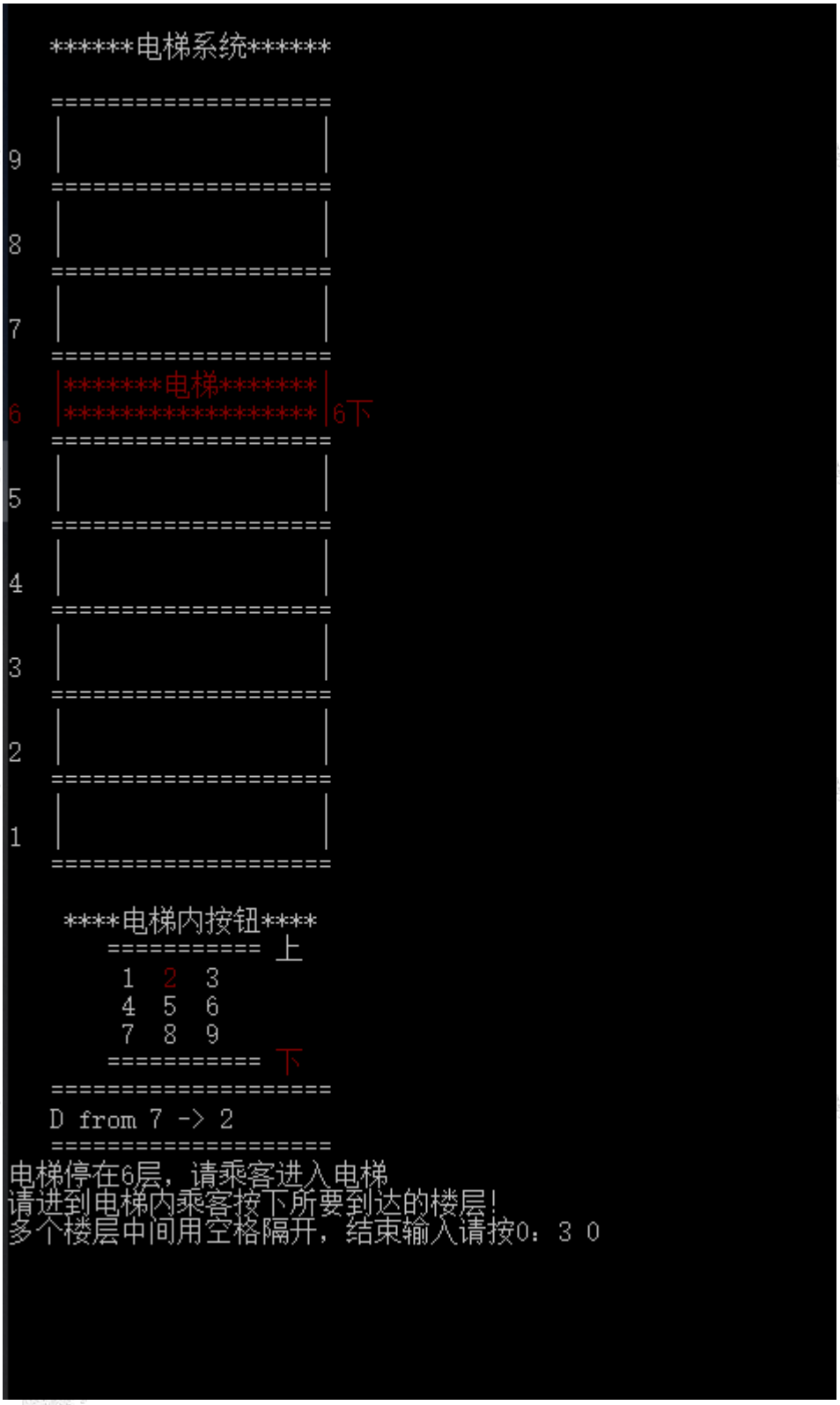
（8）此时电梯没有了向上的请求，就会向下运行。并且也没有接受到向上或向下的请求。



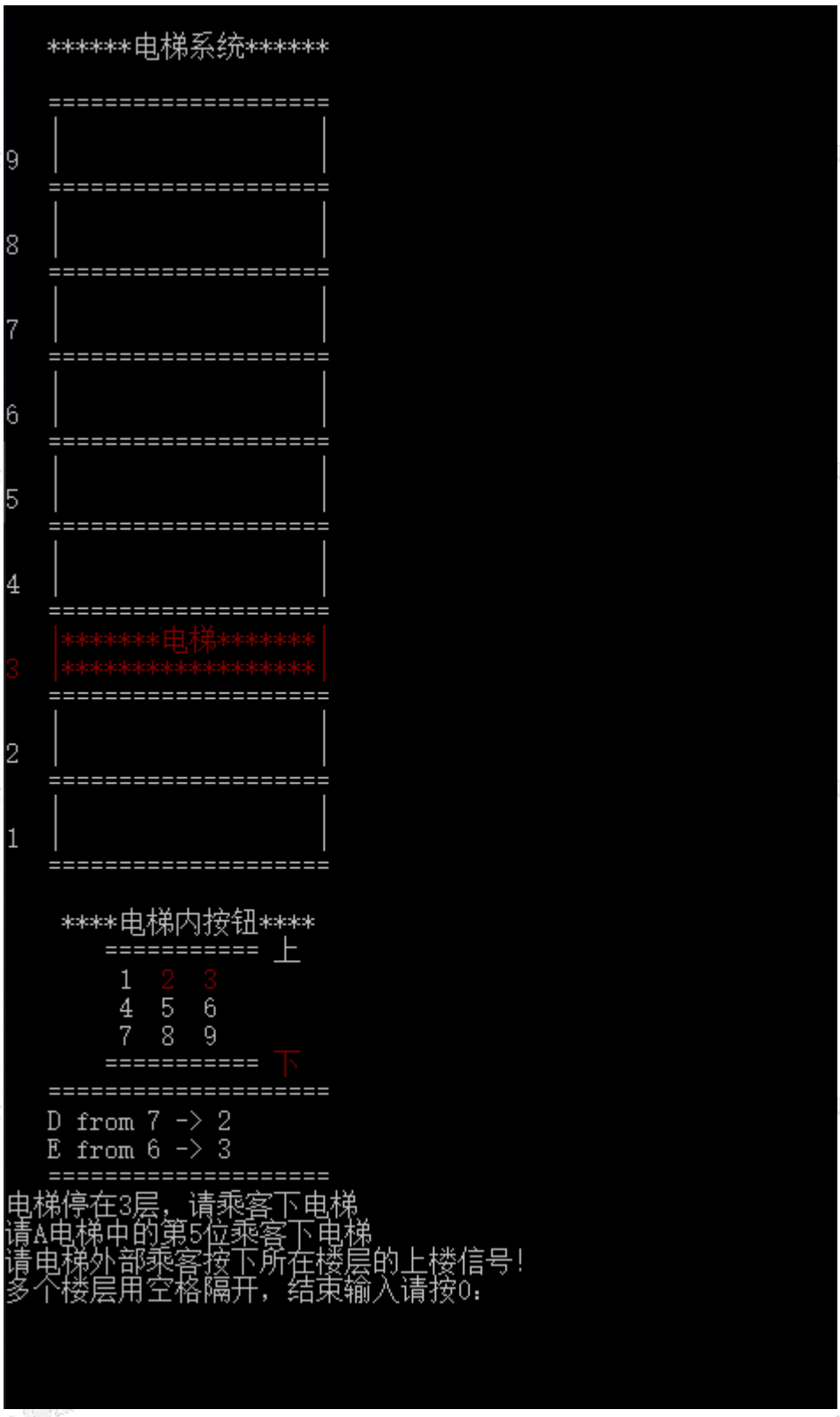
（9）电梯向下运行到第7层，乘客D进入电梯，按下2层按钮。此时电梯外没有向上或者向下的请求。

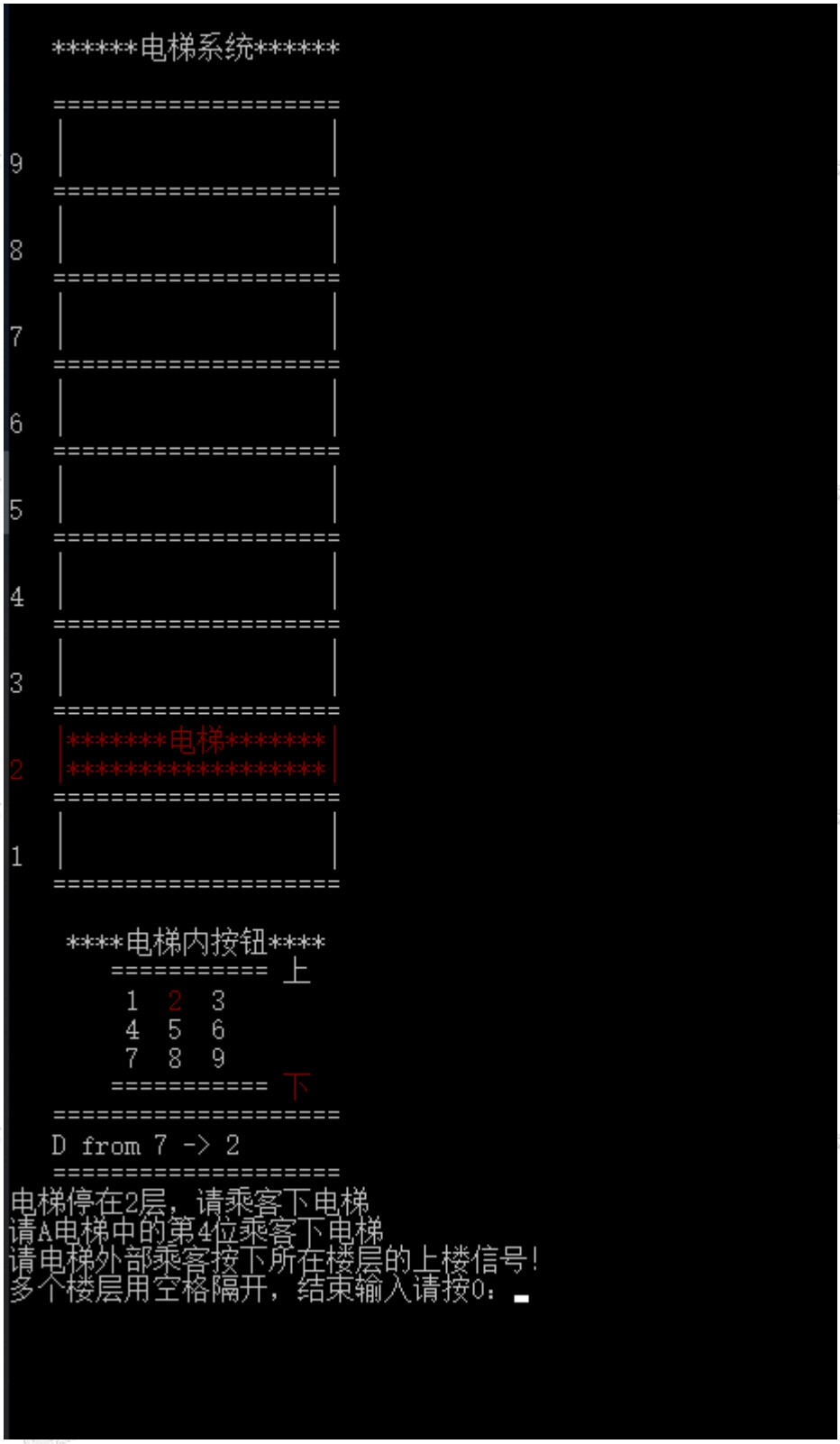


（10）电梯向下运行到第6层，乘客E进入电梯，按下3层按钮。此时电梯外没有向上或者向下的请求。

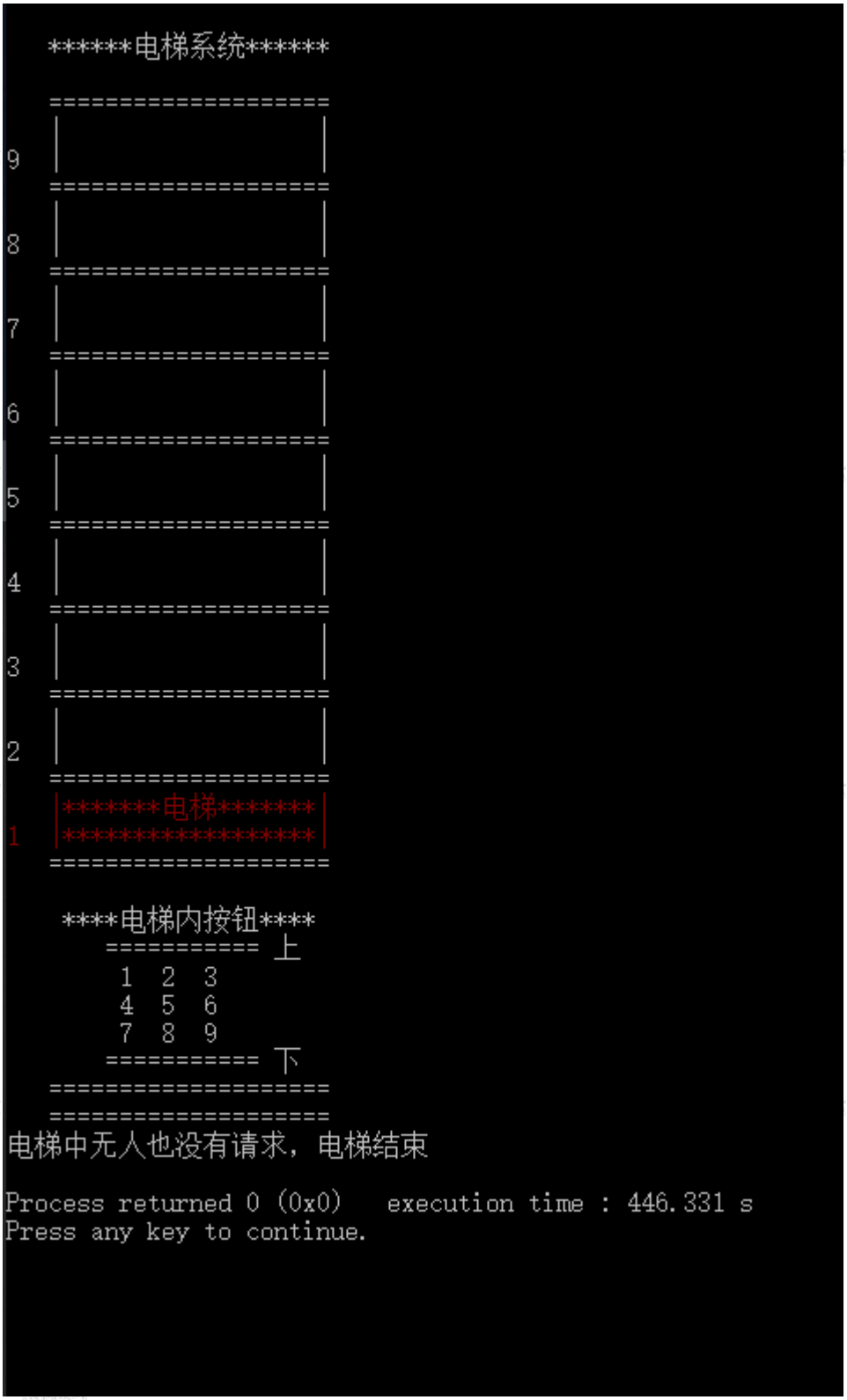


（11）电梯向下运行，以此到达2层和3层，将第5位和第4位乘客送下，中间过程没有向上或者向下的请求。


******电梯系统******
9
8
7
6
5
4
********电梯********
3
2
1
****电梯内按钮****
========== 上
1 2 3
4 5 6
7 8 9
========== 下
D from 7 -> 2
E from 6 -> 3
电梯停在3层，请乘客下电梯
请A电梯中的第5位乘客下电梯
请电梯外部乘客按下所在楼层的上楼信号!
多个楼层用空格隔开，结束输入请按0：



（12）电梯中没有乘客，电梯外也没有向上或者向下的请求，电梯将停止运行，程序结束。



完整源码：

elevator.zip