1.输入

其中格式1为请求输入；格式2为普通出租车信息查询；格式3为状态查询；格式4、5为道路的开闭；格式6为可追踪出租车查询

格式1：[CR,src,dst]

其中src和dst均为(i,j)形式的坐标位置；

横纵坐标均为大于等于0，小于等于79的整数；

一行输入一个请求，每行输入结束就开始进行相应请求的处理

例如： [CR,(3,4),(7,8)]

[CR,(34,28),(33,4)]

[CR,(3,4),(5,5)]

[CR,(7,8),(3,4)]

格式2：taxi:加id

冒号与taxi和id之间不允许有空格，否则当做无效输入 0<=id<=99

输入样例： taxi:1

taxi:89

格式3：state:加0,1,2,3

冒号与taxi和0,1,2,3之间不允许有空格，否则当做无效输入

输入样例：state:1

state:3

格式4：open:(x1,y1),(x2,y2)

不允许有空格，否则当做无效输入

输入样例：open:(1,2),(1,3)

由测试者保证同时改变的连接边不超过5条

由测试者保证前一个点在后一个点的左上（或左或上）

格式5：close:(x1,y1),(x2,y2)

不允许有空格，否则当做无效输入

输入样例：close:(1,2),(1,3)

由测试者保证同时改变的连接边不超过5条

由测试者保证前一个点在后一个点的左上（或左或上）

格式6：specialtaxi:加id

不允许有空格，否则当做无效输入

输入样例：specialtaxi:99

由测试者保证输入的id时可追踪出租车，控制台输出信息

2.输入错误提示：

a.如果不满足基本的输入格式，（括号不匹配，出现小数，非法字符等），控制台输出“非法输入”提示；

b.如何横纵坐标超出范围，控制台输出“坐标不正确”提示；

c.如果相同的请求，控制台输出“相同请求”提示；

关于相同请求：在测试过程中规定最后一条请求与前一条请求的不相同

即不允许出现：[CR,(1,1),(79,79)]

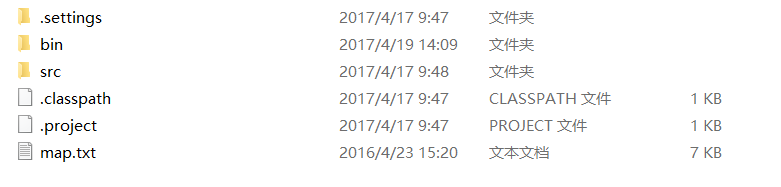
[CR,(1,1),(79,79)]

测试样例： [CR,(1,1),(79,79)]

[CR,(1,1),(79,79)]

[CR,(3,3),(4,4)]

3.地图由测试者保证存在工程包文件下,如图所示



4.红绿灯文件和地图文件在同一路径下，命名为traffic lights.txt

5.输出信息以请求命名，位于工程文件包内，同上图；

每个请求只要有车抢单就会新建一个对应的txt文件，但是有可能出现响应请求的出租车同时接了多单，并不会服务该请求（即txt中不含有路线信息）

（根据指导书中：一辆出租车在同一时刻可以抢不止一单，但每辆车一次服务只能够响应一个乘客请求。）

每次运行结束之后，手动清除工程包里面除map.txt的所有txt文件。

6.关于新增出租车类的LSP原则论证

Taxi2类继承自Taxi类，覆盖了父类除print之外的所有方法。

1.print方法在父类中是起占位作用的空方法，父类抛弃了其返回值；

2.print方法没有输入参数，所以不存在子类方法的前置条件要比父类方法更宽松的问题

3.父类中print方法为空方法，因此子类方法的后置条件显然要比父类更严格。.

7. init\_taxi 说明

主类TaxiSystem第45行为init\_taxi方法的调用；

第55行可以修改普通出租车的id

第63行可以修改可追踪出租车的id

当前状态为1到69号为普通出租车，70到99号为可追踪出租车，测试者保证修改之后的正确性

8.补充

一次输入300条请求之后gui可能有点卡，但是不影响输出正确的结果