#########################read me############################

## a)基本功能

## 2.2 作业电梯系统基本描述

1. 固定为10层楼，楼层计数采用中国式计数，即1层显示为1，2层为2，依次类推，直至顶层显示为10。
2. 每个楼层都有电梯请求按钮（下面简称为楼层按钮），告知乘客想要去的楼层与发出请求楼层之间的方向关系。如果乘客按下了向上请求按钮，则表示想搭乘电梯去往上面的楼层；反之，则表示想搭乘电梯去往下面的楼层。规定最底层只有向上运行请求按钮，最顶层只有向下运行请求按钮，中间楼层同时有向上和向下两个方向请求按钮。
3. 一个电梯箱体（剩余文档简称电梯）内有对应每个楼层的目的地请求按钮（下面简称目标），一般显示相应楼层的编号。
4. 此外，电梯还会有开门、关门按钮，以及其他的如报警处理等相关按钮。本次作业不要求在设计中体现这些按钮的设置和功能，但后续作业会逐渐引入这些按钮和功能。

## 2.3 电梯基本运行规则

1. 程序运行开始或重置时设置电梯停靠在一层；
2. 一个楼层按钮同一时刻只能发出一个上行或下行请求，在电梯未执行完毕，即到达该楼层并开关门完毕前（含开关门完毕时刻），该楼层的多个同向请求只认为是一个请求（相当于实际生活中，电梯未到本楼层的时候，请求按钮变亮后，再按不会产生实际效果，但是发出上行请求后可以再发出下行请求，反之亦可，这视为两个不同的请求，执行完一个后另一个仍需执行）。当电梯开关门结束后，可以再产生新的上下行请求；
3. 电梯内的一个目标楼层按钮只能发出对应目标楼层的请求，一旦发出请求后，在电梯未执行完毕，即到达该楼层并开关门完毕前（包括开关门完毕时刻），目标楼层与该按钮楼层相同的多个电梯内请求被认为是一个请求。当电梯开关门结束后，可以再发出任意目标楼层请求。
4. 所有请求按照请求发出的时间顺序被电梯系统管理和调度，按照时间上的先来先服务策略（First Arrived First Served，FAFS）进行调度。
5. 如果电梯同时收到了电梯内请求和楼层请求时，则按照输入时的请求排列顺序执行。
6. 本次作业的电梯系统采用傻瓜式调度策略：不断扫描队列，按照FAFS策略取出待响应请求，只有当该请求被执行完毕之后，才会尝试调度下一个请求。请注意此处不要按照常见的电梯运行调度策略进行优化设计，不要合并“顺路”的请求，如从2层去8层期间，未到6层时有6层的上行请求，应先处理完2层到8层的请求，再处理6层的请求；或者2层到8层时，中间又发出了去6层的请求，同样先执行完到8层的动作，再处理去6层的请求。

c）

## 4.1 输入规范

用户输入为按照请求产生时间排序的请求序列（注意：如果两个请求产生时间相同，则谁在前面都是合法输入），序列通过字符串表示；

请求分为两类：一类是楼层请求，一类是电梯内请求。注意，本次作业不支持开门和关门请求的处理。

楼层请求格式为：(FR, m, UP/DOWN, T)，其中FR为标识，m为发出请求的楼层号，UP为向上请求，DOWN为向下请求，T为发出时刻。

电梯内请求格式为：(ER, n, T)，其中ER为标识，n为请求去往的目标楼层号，T为发出时刻。

括号内的逗号应采用ASCII字符集中的逗号“,”，而不是中文字符逗号“，”。请求之间可以使用空格、退格和换行分隔，这一点请在readme说明。请求内部元素之间可以有空格，要求程序能够自动过滤。

T为请求产生的相对时刻（非负整数），第一个请求的T值要求设置为0。设电梯运行一个楼层距离消耗时间为0.5；达到楼层后停靠、开关门等一系列动作消耗时间为1。合法的请求产生时刻为非负整数（建议为32bit整数），n，m为1~10之间含两端的正整数。

不正确的标识符，不正确的方向，不正确的数字范围，多余的其他非允许字符，均认定为不合法输入，即无效输入。

特别地，对于FR标识符，1楼的DOWN和10楼的UP也认为是无效输入。

至于是必须在一行输入全部请求，回车后电梯运行，还是通过特殊命令结束输入，在此并不做出限定，但是要求在说明文档中说明。附可能的输入样例（例1）：

**(FR,3,DOWN,0)**

**(FR,1,UP,1)**

**(ER,1,2)**

**(ER,6,4)**

**run**

本次作业要求一次性将所有请求输入，然后执行程序进行电梯调度并输出结果。标准输入的请求是按照时间排序的，如果遇到一个乱序的请求，即请求产生时间小于前面一个请求产生时间，则该请求直接被丢掉，继续处理下一个请求。

要求程序能够忽略相同的请求，包括产生时刻相同的相同请求和产生时刻不同但是实质上相同的请求。详情见2.3.2及2.3.3中的规定。

例如（例2）：

**(FR,3,DOWN,0)**

**(FR,3,DOWN,1)**

这里第二条请求发出时第一条请求还没有执行结束，相当于楼梯按钮仍处于按下状态，第二条请求相当于按下了按下的按钮，所以与第一条请求实质上相同。

如有下例（例3）：

**(FR,3,DOWN,0)**

**(FR,3,DOWN,1000)**

第二条请求执行时，一般来说电梯已到3层，该请求相当于是同层请求，应执行一次开关门动作。对于电梯内请求的类似情况同理。

如果一直没有输入run\n程序会一直处于等待输入状态，不会停止

如果指令单独看是正确的，但是由于重复被忽略，出现的时间都会作为目前时刻，而小于这个时刻的都会被忽略

(FR,5,DOWN,0)

(FR,5,DOWN,2)

(ER,2,2)

run

会忽略第二三条指令

但是如果指令本身就存在错误（即永远不可能发出这样的指令），如

(FR,5,DOWN,0)

(FR,1,DOWN,2)

(ER,2,2)

run

第三条指令能正常执行。

指令总数最多为1000条，请留意

时间不能超过5位

## 5.2错误处理

处理原则：

* 1. 如果发现输入请求序列不满足时间排序要求，则输出提示输入有误，并忽略不满足要求的请求，继续处理下一个输入请求。
  2. 遇到无效请求（包括格式或内容不符合要求的），提示为无效输入后，继续处理下一个输入请求直至结束。
  3. 格式有误或数据无效(如楼层超过10)的请求将被直接从输入请求序列中拿掉，不影响对其他有效请求的调度处理。

#########################分类read me#########################

Request（请求） 类

含有 数组last 用来记录所有的时间（无论是否无效），用来判断时间是否僭越。

含有 标识符sign2 由Controller类返回，用来判断读入的一条指令是否读入请求队列Queue。

含有 FormCheck方法 利用正则表达式做最基本的格式检查，返回值true 或者 false

含有 main方法 通过一系列操作将从键盘读入的请求进行细致的区分处理，当sign2为true时将目前的请求写入到Queue.queue中

Queue（请求队列）类

含有 二维数组queue 用来储存已经判定为可执行的请求

含有两个方法 inserter 和insertfr 分别插入ER和FR的有效指令

Controller（控制器）类

含有 judgefr和judgeer两个方法 分别判断FR,ER指令，此时读入的格式已经正确的请求是否真正正确（时间僭越，重复命令，是否第一条指令不从时间零开始）

含有Sum方法 用来辅助judge ，做的是从第一条有效请求开始，直到第一次遇到目前输入的楼层号所需时间的总和

含有Max方法 用来辅助judge，寻找last数组中的最大数字

Elevator（电梯）类

含有run方法 挨条读取Queue.queue的指令并且执行，并且更改t，floor等属性

含有floor属性 电梯目前所在的楼层

含有t属性 电梯目前所执行到的时间

################################################################