# **果冻消除**(DevOps)

有一个果冻消除游戏，这个消除游戏的基本规则是:

1. 有一个 8x8 的棋盘，棋盘每个格子可以用一个二维坐标表示[row, col]. 其中从上到下依次是第 0, 1, .....7行，从左到右依次是第 0, 1, .....7 列;
2. 棋盘的每个格子中有一个果冻，可以是以下四种类型中的一种:普通果冻、横炸弹果冻、竖炸弹果冻和方炸弹果冻;
3. 每次用户操作时，会在棋盘上选择一个矩形区域(AABB)。该矩形可用其左上角和右下角的坐标来确定，比如[0,1] [2,2];
4. 矩形内的所有元素都会被消除，对于[0,1] [2,2]这个矩形来说，这 6 个元素都会被消除: [0,1], [0,2], [1,1], [1,2], [2,1], [2,2];
5. 炸弹果冻在消除后会有特殊的效果:横炸弹会消除掉所在行的所有果冻;竖炸弹会消除掉所在列的所有果冻;方炸弹会消除掉周围一圈 8 个果冻;
6. 被炸弹消除的果冻也按照第 5 条规则处理;
7. 消除结束后，被消掉的地方会由上方的果冻掉落进行填补。第 0行的元素被消掉(或下落到下方)时，会由系统随机生成一个果冻进行填补(4 种果冻的生成概率均等)。

你的任务是开发并部署一个服务器端程序(使用HTTP 协议)，实现如下的 2 个接口:

1. 开始某一关卡

/start-level?level=<关卡编号>

返回: 第一行为一个字符串(sessionId)，用于唯一标识一个进行中的关卡，sessionId 的格式可以自行确定; 后续 8行表示关卡初始布局，对应棋盘中的 8行，每行包括 8 个字符。每个字符表示一种果冻，对应关系为: (B 普通果冻，H 横炸弹果冻，V 竖炸弹果冻，S方炸弹果冻)

2. 一次消除操作

/move?
sessionId=<sessionId>&row0=<row0>&col0=<col0>&row1=<row1>&col1=<col1>

参数:

sessionId:第 1 个接口返回的 sessionId

row0, col0, row1, col1: 表示一次操作的左上角和右下角的果冻坐标，比如: row0=0&col1=1&row1=2&row2=2

返回:

8行文本，表示消除结束且完成下落后的局面，格式与接口 1 的返回中后 8行相同。

**举例**:

请求:
/start-level?level=1
响应:
40ab3f2j
HBBBSBBB
BBBBBBBB
BSBBBBBB
BBBBVBBB
BHBBBBBB
SBBBBSBB
BBBBBBBB
BBBBBBBB

请求:
/move?sessionId=40ab3f2j&row0=1&col0=0&row1=1&col1=2
响应:
VSBBSBBB
HBBBBBBB
BSBBBBBB
BBBBVBBB
BHBBBBBB
SBBBBSBB
BBBBBBBB
BBBBBBBB
说明:消除了第 1行的前 3 个普通果冻后，第 0行的前 3 个果冻掉落到第 1行，第 0行由系统补充了 3 个果冻，分别是 V、S、B

请求:
/move?sessionId=40ab3f2j&row0=5&col0=0&row1=6&col1=2
响应
BBSBBBBH
SBBBSBBB
BBHBBBBB
VSBBBBBB
HBBBVBBB
BSBBBSBB
BBBBBBBB
BBBBBBBB
说明:初始消除了 5 个普通果冻和一个方炸弹，方炸弹爆炸时，炸到了[4,0]的普通果冻和[4,1]的横炸弹，炸弹将第 4行全部消除

其他要求:

1. 使用 docker 部署一个关系型数据库或者非关系型数据库做持久化存储，例如 mysql/postgresql/redis/mongodb 等
2. 每一关的初始布局保存在数据库中
3. 使用 docker 部署一个负载均衡服务，例如 nginx,haproxy 等
3. 编写 Dockerfile 对编写好的代码进行打包
4. 在负载均衡器的后面，用 docker 部署游戏服务器的多个副本（例如 2 个），重启其中一个服务器不会影响玩家进行游戏
5. 所有接口均以 HTTP GET 请求的方式由客户端向服务器发送，服务器以 HTTP 响应的方方式返回给客户端，返回格式为纯文本(不带任何 HTML 标签)
6. 如果用户给出的参数非法(无论是类型还是数值)，仅返回一行文本:"INVALID PARAMS"
7. 最后提交的项目需要包含完整的项目文件，源代码(包括必要的注释），Dockerfile，创建数据库的相关脚本，以及部署文档（部署文档中可用文字简要说明监控方案）。
8. 加分项 1：编写 Dockerfile 时，为了减小最终镜像大小和优化打包速度，尽量参考如下两条最佳实践
    a) https://docs.docker.com/develop/develop-images/dockerfile_best-practices/#use-multi-stage-builds
    b) https://docs.docker.com/develop/develop-images/dockerfile_best-practices/#leverage-build-cache
9. 加分项 2：在部署部分可以使用 k8s 代替 docker 来完成（也可使用 k8s 自带的负载均衡功能）。在部署文档和 YAML 中，假设 HTTP 请求的并发很高，应尽量提高游戏服务器的稳定性和高可用性。（无须考虑数据库的稳定性）