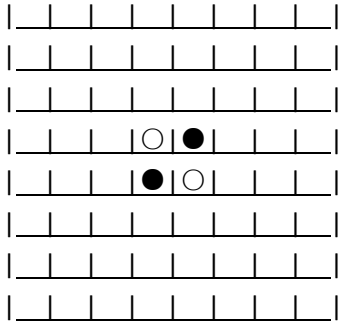
# 黑白棋游戏说明

## [游戏规则]

### 【题目内容】

棋盘为 8X8。黑白双方交替落子，但落子位置必须保证能让自己的子“夹”住对方的子来翻转对方的棋子，使之成为自己的子，同时不考虑连锁反应。若没有这种位置则己方跳过回合。双方均无子可下的时候游戏结束，数出双方棋子数，棋多者胜。

棋盘初始状态为



黑方为先手。

### 【基本要求】

1. 有菜单选择（选择，新开始，存盘，读盘，结束）。
2. 用字符实现画棋盘和棋子。
3. 一方选手是人员（助教），另一方选手是计算机 AI，进行人机对战。
4. 程序的输入是人员的落子位置，程序要根据输入，在棋盘上显示变化。程序根据算法决定 AI 方下一步棋子所放的位置，并显示新的棋盘布局。
5. 允许中途停止游戏。
6. 有读盘的功能（玩到一半，存储棋盘的状态）。

### 【分组】

1. 可以一个人一组，最多两人一组。
2. 两个人一组时，最多只能一位同学的成绩是优秀。
3. 鼓励两人一组，基础好的同学帮助基础差一些的同学。优秀率向两人一组的情况倾斜。

### 【成绩评定】

1. 程序质量：完成基本要求的基础上，鼓励自行发挥。欢迎同学们多动脑，做出好的实验题。
2. 工作量：分工要明确，两个人一组时，每个人的工作的最小单位是函数，在源程序上注明每个函数的完成人，以便提问。

3. 提交内容：将源程序或程序包（包含源程序）压缩，提交到网站上。
4. 实验报告：对程序的设计思路和功能做一个大概的说明，尤其自己认为有独特的地方， 在实验报告中突出出来，提交到网站上。
5. 验收形式：在规定的时间内，到机房找助教，演示程序，并回答助教提出的问题。
6. 评分标准：满分 10 分。助教会根据程序质量、回答问题的正确性、功能的完善等指标评定分数。
7. 打分方式：完成基本功能即可获得基础分 6 分。在此基础之上，在智能能力、交互方式等方面做得优秀，都可以获得加分（具体尺度由助教掌握），**参加 botzone 比赛的作品也可以获得加分。**
8. **成绩的计算：在计算成绩之前，我们会对所有助教的打分进行归一化处理，因此，各位同学不必担心不同助教打分差异的问题。**
9. **截止时间：2023 年 12 月 31 日晚上 11:59，没有经过助教检查的，过期不交没有成绩**。

**【提示】**

1. 在 word 文档中，把制表符拷贝下来，粘贴到程序里，用 cout 输出，可以画出棋盘
2. 用数组记录棋盘上的位置
3. 每次输出棋盘的状态之前，都要用 system("cls");语句或类似的语句清空屏幕

## [Botzone 接口定义]

交互的方式是标准输入、标准输出，具体如下所示。

你的 BOT 每回合都会接到输入：

*[YOURCOLOR] 1 或-1*

(我方分配的颜色，1 为黑，-1 为白)

*[OPP] x y*

(对方前一步的列、行号，x=0..7，y=0..7，黑方第一步将得到-1 -1)

每回合 BOT 都需要输出：

*x y endl*

(己方本步的列、行号，x=0..7，y=0..7，无子可下要输出-1 -1)

如果你的程序崩溃、一步超过 1 秒、输出非法，或者意外退出，那么本局比赛将立即结束并被判输。

## [输入输出交互样例]

*>>[YOURCOLOR] 1 [OPP] -1 -1*

*<<3 2*

*>>[YOURCOLOR] 1 [OPP] 4 2*

*<<5 2*

*>>[YOURCOLOR] 1 [OPP] 2 2*

*<<1 2*

*>>[YOURCOLOR] 1 [OPP] 5 3*

*<<5 4*

*>>[YOURCOLOR] 1 [OPP] 3 1*

*<<3 0*

其中“>>”表示输入、“<<”表示输出，并不是实际输入、输出的字符。至此，己方在本局比赛中获胜（将对方棋子吃尽），程序结束。

## [样例程序]

```
//////////////////////////////
//名称：黑白棋演示 BOT        //
//作者：周昊宇                //
//日期：2012 年 12 月 8 日        //
//说明：可以作为 BOT 提交的样例//
//////////////////////////////
// 注意这只是个伪代码，不能直接运行
#include <iostream>
#include <string>
using namespace std;
int myColor; // 记录己方颜色

int main()
{
    string controlStr; // 记录当前控制命令
    int oppX, oppY; // 记录对手落子位置
    int gridInfo[8][8] = {0}; // 记录棋盘状态
```

```
    // 初始化棋盘
    gridInfo[3][4] = gridInfo[4][3] = 1;
    gridInfo[3][3] = gridInfo[4][4] = -1;

    while (cin >> controlStr) // 循环获取控制命令
    {
        if (controlStr == "[YOURCOLOR]")
            cin >> myColor; // 获得己方颜色
        if (controlStr == "[OPP]")
        {
            cin >> oppX >> oppY; // 获得对方落子坐标
            if (oppX != -1) // 对方若落子则模拟对方落子
                ProcStep(gridInfo, oppX, oppY, -myColor);
            FindPossiblePos(myX, myY, gridInfo, myColor); // 找出自己的合法落点
            if (找不到落子点)
                cout << "-1 -1" << endl; // 无法落子要输出-1 -1
            else
            {
                cout << myX << ' ' << myY << endl; // 分别输出 x、y 坐标
                ProcStep(gridInfo, myX, myY, myColor); // 记得也模拟自己走的一步
            }
        }
    }
    return 0;
}
```