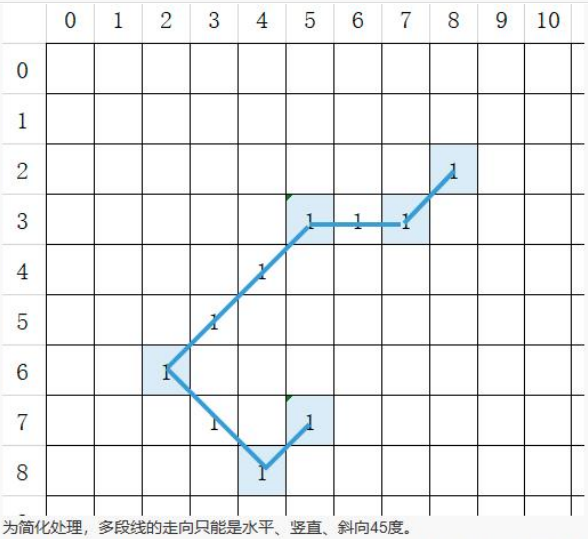
题目描述：

下图中，每个方块代表一个像素，每个像素用其行号和列号表示。



为简化处理，多段线的走向只能是水平、竖直、斜向45度。

上图中的多段线可以用下面的坐标串表示：(2, 8), (3, 7), (3, 6), (3, 5), (4, 4), (5, 3), (6, 2), (7, 3), (8, 4), (7, 5)。

但可以发现，这种表示不是最简的，其实只需要存储6个蓝色的关键点即可，它们是线段的起点、拐点、终点，而剩下4个点是冗余的。

现在，请根据输入的包含有冗余数据的多段线坐标列表，输出其最简化的结果。

输入描述：2 8 3 7 3 6 3 5 4 4 5 3 6 2 7 3 8 4 7 5

1、所有数字以空格分隔，每两个数字一组，第一个数字是行号，第二个数字是列号；

2、行号和列号范围为[0,64)，用例输入保证不会越界，考生不必检查；

3、输入数据至少包含两个坐标点。

输出描述：2 8 3 7 3 5 6 2 8 4 7 5

压缩后的最简化坐标列表，和输入数据的格式相同。

补充说明：输出的坐标相对顺序不能变化。

示例 1

输入：

2 8 3 7 3 6 3 5 4 4 5 3 6 2 7 3 8 4 7 5

输出：

2 8 3 7 3 5 6 2 8 4 7 5

说明：

如上图所示，6 个蓝色像素的坐标依次是（2,8）、（3,7）、（3,5）、（6,2）、（8,4）、（7,5）。

将他们按顺序出即可。

```
import java.util.*;

// 注意类名必须为 Main, 不要有任何 package xxx 信息
public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        // 注意 hasNext 和 hasNextLine 的区别
        List<String> list = new ArrayList<>();
        List<String> originList = new ArrayList<>();
```

```
int count = 1;
int lastStatus = 0;
//1 水平左 2 水平右 3 垂直上 4 垂直下 5 斜左上 6 斜左下 7 斜右上 8 斜右
下
while (in.hasNextInt()) { // 注意 while 处理多个 case
    int a = in.nextInt();
    int b = in.nextInt();

    if(count == 1){
        originList.add(a + " " + b);
    }
    if(count > 1){
        String[] ss = originList.get(originList.size()-1).split(" ");
        int x = Integer.valueOf(ss[0]);
        int y = Integer.valueOf(ss[1]);
        int curStatus = 0;
        if(x - a == 0){
            if(y - b > 0){
                curStatus = 4;
            }else{
                curStatus = 3;
            }
        }
        if(y -b == 0){
            if(x -a > 0){
                curStatus = 1;
            }else{
                curStatus = 2;
            }
        }
        if(x-a >0 && y-b > 0){
            curStatus = 6;
        }
        if(x-a <0 && y-b > 0){
            curStatus = 8;
        }
        if(x-a > 0 && y-b <0){
            curStatus = 5;
        }
        if(x-a <0 && y-b < 0){
            curStatus = 7;
        }
        if(curStatus != lastStatus){
            lastStatus = curStatus;
```

```
                    list.add(x + " " + y);
                }
                originList.add(a + " " + b);
            }

            count++;


        }
        for(String s : list){
            System.out.print(s + " ");
        }
        System.out.print(originList.get(originList.size()-1));

    }
}
```