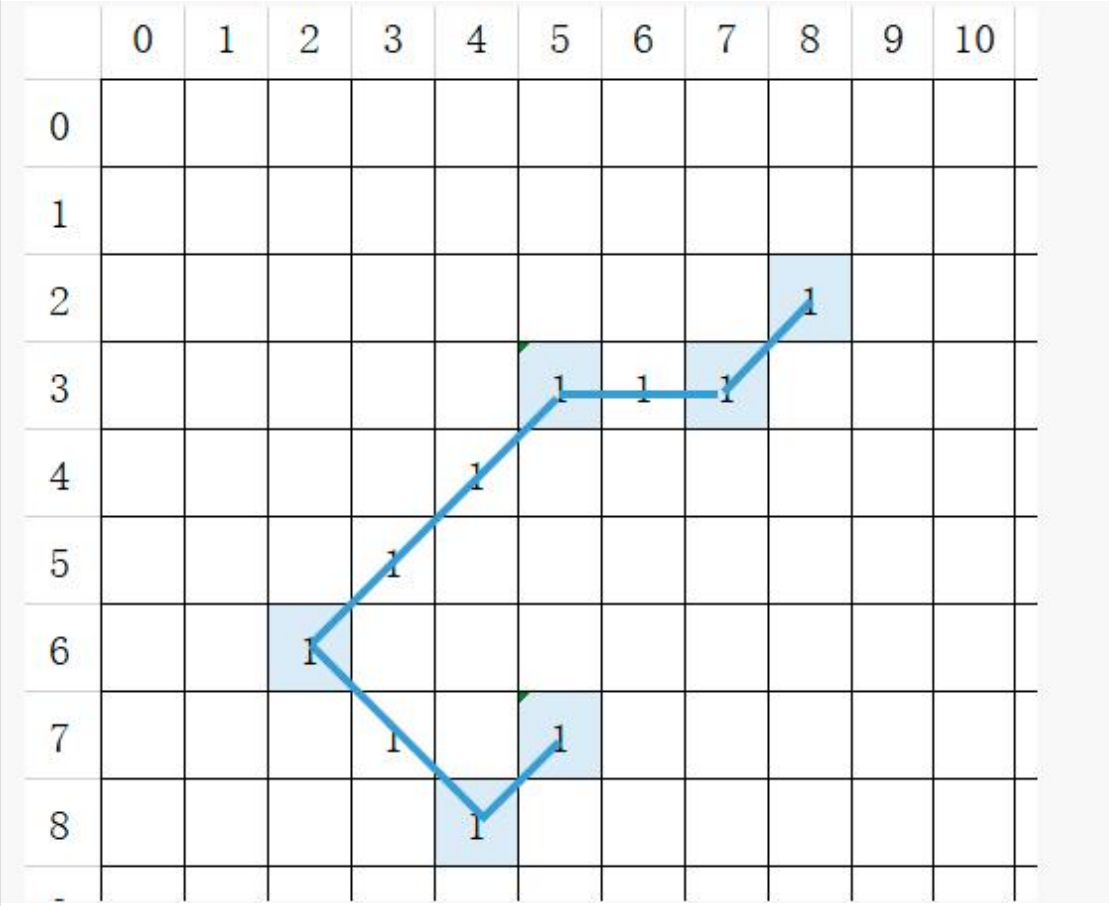
题目描述：

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 scanner = new Scanner(System.in);
        String input = scanner.nextLine();
        List<Integer> point = parsePoint(input);
        List<Integer> simple = simpleMoreLine(point);
        String out = formatPoint(simple);
        System.out.println(out);
    }

    private static String formatPoint(List<Integer> simple) {
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < simple.size(); i+=2) {
            Integer row = simple.get(i);
            Integer col = simple.get(i + 1);
            sb.append(row).append(" ").append(col).append(" ");
```

```
            }
            return sb.toString().trim();
        }

        private static List<Integer> simpleMoreLine(List<Integer> point) {
            List<Integer> simplePoint = new ArrayList<>();
            int size = point.size();
            simplePoint.add(point.get(0));
            simplePoint.add(point.get(1));
            for (int i = 2; i < size - 2; i+=2) {
                int row = point.get(i);
                int col = point.get(i + 1);
               if(isKeyPoint(i,point)){
                    simplePoint.add(row);
                    simplePoint.add(col);
               }
            }
            simplePoint.add(point.get(size-2));
            simplePoint.add(point.get(size-1));
            return simplePoint;
        }

        private static boolean isKeyPoint(int i, List<Integer> point) {
            Integer row = point.get(i);
            Integer col = point.get(i + 1);
            Integer preRow = point.get(i - 2);
            Integer preCol = point.get(i - 1);
            Integer nextRow = point.get(i + 2);
            Integer nextCol = point.get(i + 3);
            return (row-preRow)*(col-nextCol)!=(row-nextRow)*(col-preCol);
        }

        private static List<Integer> parsePoint(String input) {
            List<Integer> point = new ArrayList<>();
            String[] parts = input.split(" ");
            for (String part : parts) {
                point.add(Integer.parseInt(part));
            }
            return    point;
        }
    }
```