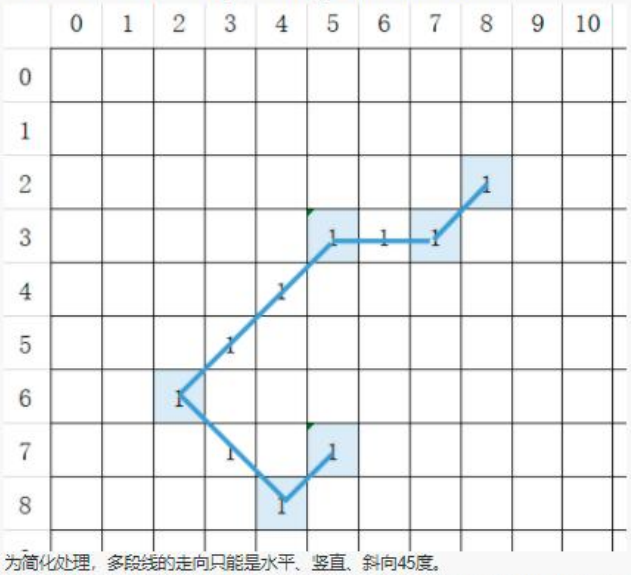
题目描述：下图中，每个方块代表一个像素，每个像素用其行号和列号表示。



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 sys

input_points = input()
points = input_points.strip().split()
points = [[int(points[i * 2]), int(points[i * 2 + 1])] for i in range(len(points) // 2)]
if len(points) < 3:
    print(input_points)
else:
    simple_points = [points[0]]
```

```
cache_points = points[:2]
for x, y in points[2:]:
    last_point = cache_points[-1]
    diff_x = last_point[0] - cache_points[-2][0]
    diff_y = last_point[1] - cache_points[-2][1]
    next_x = last_point[0] + diff_x
    next_y = last_point[1] + diff_y
    if x == next_x and y == next_y:
        if simple_points[-1] != cache_points[0]:
            simple_points.append(cache_points[0])
        cache_points.append([x, y])
    else:
        cache_points = [last_point, [x, y]]
        simple_points.append(last_point)
simple_points.append(cache_points[-1])
print(" ".join([str(x) + " " + str(y) for x, y in simple_points]))
```