```
package droneWork;

import java.util.*;

public class droneAlgo {

    static class Point {
        int x, y, time;
        Point(int x, int y, int time) {
            this.x = x;
            this.y = y;
            this.time = time;
        }
    }

    public static List<List<Point>> getDronePaths(int[][] drones) {
        List<List<Point>> result = new ArrayList<>();
        int n = drones.length;
        boolean[][][] visited = new boolean[n][101][101]; // to keep track of visited points
        int[][] dirs = {{-1, 0}, {1, 0}, {0, -1}, {0, 1}}; // possible directions
        for (int i = 0; i < n; i++) {
            int[] drone = drones[i];
            Point start = new Point(drone[0], drone[1], drone[4]);
            Point end = new Point(drone[2], drone[3], Integer.MAX_VALUE); // target point with large time
            List<Point> path = new ArrayList<>();
            Queue<Point> queue = new LinkedList<>();
            queue.offer(start);
            visited[i][start.x][start.y] = true;
            while (!queue.isEmpty()) {
                Point curr = queue.poll();
                if (curr.x == end.x && curr.y == end.y) { // target reached
                    path.add(curr);
                    break;
                }
                for (int[] dir : dirs) {
                    int nextX = curr.x + dir[0];
                    int nextY = curr.y + dir[1];
                    int nextTime = curr.time + 1;
                    if (nextX >= 0 && nextX <= 100 && nextY >= 0 && nextY <= 100 &&
                        !visited[i][nextX][nextY]) { // check if point is valid and not visited
                        queue.offer(new Point(nextX, nextY, nextTime));
                        visited[i][nextX][nextY] = true;
                    }
                }
            }
            path.add(0, start);
            result.add(path);
        }
        return result;
    }

    public static void main(String[] args) {
        int[][] drones = {{0, 0, 3, 5, 11}, {1, 1, 5, 7, 13}, {3, 2, 8, 9, 15}};
        List<List<Point>> paths = getDronePaths(drones);
        for (List<Point> path : paths) {
```

```
            for (Point point : path) {
                System.out.print("(" + point.x + "," + point.y + ") ");
            }
            System.out.println();
        }
    }
}
```