```
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

using System;
using System.Collections.Generic;

public class Program {
  static string UP = "up";
  static string RIGHT = "right";
  static string DOWN = "down";
  static string LEFT = "left";

  // O(n^2) time | O(n^2) space - where n is the number of coordinates
  public static int RectangleMania(Point[] coords) {
    Dictionary<string, Dictionary<string, List<Point> > > coordsTable = getCoordsTable(
      coords);
    return getRectangleCount(coords, coordsTable);
  }

  public static Dictionary<string, Dictionary<string, List<Point> > > getCoordsTable(
    Point[] coords) {
    Dictionary<string, Dictionary<string,
      List<Point> > > coordsTable = new Dictionary<string,
        Dictionary<string,
        List<Point> > >();
    foreach (Point coord1 in coords) {
      Dictionary<string, List<Point> > coord1Directions = new Dictionary<string,
          List<Point> >();
      coord1Directions.Add(UP, new List<Point>());
      coord1Directions.Add(RIGHT, new List<Point>());
      coord1Directions.Add(DOWN, new List<Point>());
      coord1Directions.Add(LEFT, new List<Point>());
      foreach (Point coord2 in coords) {
        string coord2Direction = getCoordDirection(coord1, coord2);
        if (coord1Directions.ContainsKey(coord2Direction)) coord1Directions[
            coord2Direction].Add(coord2);
      }
      string coord1string = coordTostring(coord1);
      coordsTable.Add(coord1string, coord1Directions);
    }
    return coordsTable;
  }

  public static string getCoordDirection(Point coord1, Point coord2) {
    if (coord2.y == coord1.y) {
      if (coord2.x > coord1.x) {
        return RIGHT;
      } else if (coord2.x < coord1.x) {
        return LEFT;
      }
    } else if (coord2.x == coord1.x) {
      if (coord2.y > coord1.y) {
        return UP;
      } else if (coord2.y < coord1.y) {
        return DOWN;
      }
    }
    return "";
  }

  public static int getRectangleCount(Point[] coords, Dictionary<string, Dictionary<string,
    List<Point> > > coordsTable)
  {
    int rectangleCount = 0;
    foreach (Point coord in coords) {
      rectangleCount += clockwiseCountRectangles(coord, coordsTable, UP, coord);
    }
    return rectangleCount;
  }

  public static int clockwiseCountRectangles(
    Point coord,
    Dictionary<string, Dictionary<string, List<Point> > > coordsTable,
    string direction,
    Point origin
    ) {
    string coordstring = coordTostring(coord);
    if (direction == LEFT) {
      bool rectangleFound = coordsTable[coordstring][LEFT].Contains(origin);
      return rectangleFound ? 1 : 0;
    } else {
      int rectangleCount = 0;
      string nextDirection = getNextClockwiseDirection(direction);
      foreach (Point nextCoord in coordsTable[coordstring][direction]) {
        rectangleCount += clockwiseCountRectangles(nextCoord, coordsTable,
            nextDirection, origin);
      }
      return rectangleCount;
    }
  }

  public static string getNextClockwiseDirection(string direction) {
    if (direction == UP) return RIGHT;
    if (direction == RIGHT) return DOWN;
    if (direction == DOWN) return LEFT;
    return "";
  }

  public static string coordTostring(Point coord) {
    return coord.x.ToString() + "-" + coord.y.ToString();
  }

  public class Point {
    public int x;
    public int y;

    public Point(int x, int y) {
      this.x = x;
      this.y = y;
    }

    public bool equals(object a) {
      return this.x == ((Point) a).x && this.y == ((Point) a).y;
    }
  }
}
```