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

#include <vector>
#include <unordered_map>
#include <algorithm>
using namespace std;

struct Point {
  int x;
  int y;

  bool operator==(Point point2) { return x == point2.x && y == point2.y; }
};

string UP = "up";
string RIGHT = "right";
string DOWN = "down";
string LEFT = "left";

unordered_map<string, unordered_map<string, vector<Point>>>
getCoordsTable(vector<Point> coords);
string getCoordDirection(Point coord1, Point coord2);
int getRectangleCount(
    vector<Point> coords,
    unordered_map<string, unordered_map<string, vector<Point>>> coordsTable);
int clockwiseCountRectangles(
    Point coord,
    unordered_map<string, unordered_map<string, vector<Point>>> coordsTable,
    string direction, Point origin);
string getNextClockwiseDirection(string direction);
string coordToString(Point coord);

// O(n^2) time | O(n^2) space - where n is the number of coordinates
int rectangleMania(vector<Point> coords) {
  unordered_map<string, unordered_map<string, vector<Point>>> coordsTable =
      getCoordsTable(coords);
  return getRectangleCount(coords, coordsTable);
}

unordered_map<string, unordered_map<string, vector<Point>>>
getCoordsTable(vector<Point> coords) {
  unordered_map<string, unordered_map<string, vector<Point>>> coordsTable;
  for (Point coord1 : coords) {
    unordered_map<string, vector<Point>> coord1Directions({
        {UP, vector<Point>{}},
        {RIGHT, vector<Point>{}},
        {DOWN, vector<Point>{}},
        {LEFT, vector<Point>{}},
    });
    for (Point coord2 : coords) {
      string coord2Direction = getCoordDirection(coord1, coord2);
      if (coord1Directions.find(coord2Direction) != coord1Directions.end()) {
        coord1Directions[coord2Direction].push_back(coord2);
      }
    }
    string coord1String = coordToString(coord1);
    coordsTable.insert({coord1String, coord1Directions});
  }
  return coordsTable;
}

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 "";
}

int getRectangleCount(
    vector<Point> coords,
    unordered_map<string, unordered_map<string, vector<Point>>> coordsTable) {
  int rectangleCount = 0;
  for (Point coord : coords) {
    rectangleCount += clockwiseCountRectangles(coord, coordsTable, UP, coord);
  }
  return rectangleCount;
}

int clockwiseCountRectangles(
    Point coord,
    unordered_map<string, unordered_map<string, vector<Point>>> coordsTable,
    string direction, Point origin) {
  string coordString = coordToString(coord);
  if (direction == LEFT) {
    bool rectangleFound = find(coordsTable[coordString][LEFT].begin(),
                               coordsTable[coordString][LEFT].end(),
                               origin) != coordsTable[coordString][LEFT].end();
    return rectangleFound ? 1 : 0;
  } else {
    int rectangleCount = 0;
    string nextDirection = getNextClockwiseDirection(direction);
    for (Point nextCoord : coordsTable[coordString][direction]) {
      rectangleCount += clockwiseCountRectangles(nextCoord, coordsTable,
                                                 nextDirection, origin);
    }
    return rectangleCount;
  }
}

string getNextClockwiseDirection(string direction) {
  if (direction == UP)
    return RIGHT;
  if (direction == RIGHT)
    return DOWN;
  if (direction == DOWN)
```

```
    return LEFT;
  return "";
}

string coordToString(Point coord) {
  return to_string(coord.x) + "-" + to_string(coord.y);
}

```