```cpp
/// This version deals with 'Wizard' in TOI 9 at Thammasart University.
/// However, there are several computational redundancies that needs to
///   be removed.
/// Author: Pinyo Taeprasartsit
#include <unordered_map>
#include <vector>
#include <cstdio>

typedef std::pair<int,int> COORD;

using namespace std;

template <class T> class CoordHash;

template<>
class CoordHash<COORD> {
public:
    std::size_t operator()(const COORD& key) const {
        return key.first + key.second;
    }
};

vector<COORD> w1, w2, w3, w4;

void readWizardData(vector<COORD>& vecW, const int N) {
    vecW.resize(N);
    for(int i = 0; i < N; ++i) {
        scanf("%d %d", &vecW[i].first, &vecW[i].second);
    }
}

void buildMap(unordered_map<COORD, COORD, CoordHash<COORD>>& coordMap,
              vector<COORD>& vecW1, vector<COORD>& vecW2)
{
    const int N = (int) vecW1.size();
    for(int i = 0; i < N; ++i) {
        COORD& coord1 = vecW1[i];
        for(int j = 0; j < N; ++j) {
            COORD& coord2 = vecW2[j];
            coordMap[COORD(coord1.first + coord2.first,
                           coord1.second + coord2.second)] = COORD(i, j);

        }
    }
}

unordered_map<COORD, COORD, CoordHash<COORD>> coordMap1(10000000);
unordered_map<COORD, COORD, CoordHash<COORD>> coordMap2(10000000);

int main() {
    int tx, ty, N;
    scanf("%d %d", &tx, &ty);
    scanf("%d", &N);
    readWizardData(w1, N);
    readWizardData(w2, N);
    readWizardData(w3, N);
    readWizardData(w4, N);

    buildMap(coordMap1, w1, w2);
    buildMap(coordMap2, w3, w4);

    COORD key;
    for(auto& pair1 : coordMap1) {
        int dx = tx - pair1.first.first;
        int dy = ty - pair1.first.second;
        key.first = dx;
        key.second = dy;
        if(coordMap2.find(key) != coordMap2.end()) {
            COORD value2 = coordMap2[key];
            int order1 = pair1.second.first;
            int order2 = pair1.second.second;
            int order3 = value2.first;
            int order4 = value2.second;
            printf("%d %d\n", w1[order1].first, w1[order1].second);
            printf("%d %d\n", w2[order2].first, w2[order2].second);
            printf("%d %d\n", w3[order3].first, w3[order3].second);
            printf("%d %d\n", w4[order4].first, w4[order4].second);
        }
    }

    return 0;
}


/**
-2 2
2
1 2 -2 10
-6 -6 -1 3
-1 -2 -6 -5
5 -4 7 0

-1 3
3
1 -10 16 3 -11 -10
-17 7 -15 -2 -7 9
-2 6 -18 -15 5 19
9 -18 -7 -17 19 4
*/
```