Problem
Note that it is not necessary to know anything about the rules of chess to solve this problem.

There are N kings on an infinite chessboard (two-dimensional grid), located in cells with coordinates (X1, Y1), (X2, 2), ..., (XN, YN). Both N and the kings' coordinates are unknown to you. However, you do know the following thing :

N is at least 1 and at most Nmax.
No king's coordinates (X or Y) have an absolute value exceeding M.
The N kings are located in N different cells.
The kings want to meet in a single cell of the board. If some cell (X, Y) were to be chosen as the meeting cell, then i order to get there, the i-th king would use a number of moves equal to the maximum of the absolute values of the di ferences of coordinates between its cell and the meeting cell: max(|X-Xi|,|Y-Yi|). The total number of moves used b all kings is thus equal to the sum of those maximums over all values of i. Note that it is not relevant to this problem xactly how the kings move on the board — only the source and destination cells matter, and the number of moves ca always be computed using the above formula.

This problem has two phases. In the first phase, you may repeatedly do the following: propose a meeting location (A B) (with each of A and B between -10×M and 10×M, inclusive), and have the judge tell you the total number of mo es the kings would use to get there — the sum (over all i) of max(|Xi-A|,|Yi-B|). You can have at most R such excha ges with the judge, choosing your values of A and B each time. Note that the kings do not actually move, so their loc tions (Xi, Yi) stay the same for all requests within one test case.

In the second phase, the roles are swapped: the judge gives you a meeting cell location (C, D) (with each of C and D between -10×M and 10×M, inclusive), and you must respond with the total number of moves the kings would use to get there, assuming that the kings are in the same locations as in the first phase. There are at most R such exchanges, and you must correctly respond to all of the judge's requests.

Input and output
This is an interactive problem. You should make sure you have read the information in the Interactive Problems sect on of our FAQ.

Initially, your program should read a single line containing four integers T, Nmax, M and R: the number of test case , the maximum number of kings, the maximum absolute value for any coordinate for any king, and the maximum nu ber of requests per phase, respectively. (Note that the values of M and R are fixed, and are provided as input only for convenience; see the Limits section for more details.) Then, you need to process T test cases.

In each test case, there are two phases. In the first phase, the i-th exchange is as follows:

Your program sends one line containing two integers Ai and Bi, representing the x and y coordinates of a cell.
Both Ai and Bi must be between -10×M and 10×M, inclusive.
The judge responds with one line containing a single integer: the total number of moves the kings need to use to get rom their unknown locations to your cell.
You may initiate at most R such exchanges in this phase. If you make more than R exchanges, or send a request that he judge can not parse or is out of bounds, the judge responds with one line with a single string ERROR.

To end the first phase and switch to the second phase, you must send one line with the string READY (the case does not matter), to which the judge responds with the first request of the second phase.

In the second phase, the i-th exchange is as follows:

The judge sends one line containing two integers Ci and Di, representing the x and y coordinates of a cell.

Each of Ci and Di will be between -10×M and 10×M, inclusive.
Your program must respond with one line containing a single integer: the total number of moves the kings would ne d to use to get to the given cell.
The judge is guaranteed to send at least 1 and at most R such requests. If you send an answer that is incorrect or unp rseable, the judge responds with ERROR as described above. If you answer all of the requests correctly, the judge se ds one line with a single string DONE, at which point your program should initiate the next test case, or terminate wi h no error if all T test cases have been handled.

After the judge sends a line with ERROR, it does not send any other output. If your program continues to wait for th judge after receiving ERROR, your program will time out, resulting in a Time Limit Exceeded error. Notice that it i your responsibility to have your program exit in time to receive a Wrong Answer judgment instead of a Time Limit xceeded error. As usual, if the memory limit is exceeded, or your program gets a runtime error, you will receive the ppropriate judgment.

The number and location of the kings, as well as the number and positions of the requests that the judge sends durin the second phases, are chosen before any exchanges occur.

Limits
Time limit: 60 seconds per test set.
Note that a program that just makes valid exchanges with the judge (and does no other processing) takes the followi g time in our environment: ~13 seconds for C++, ~24 seconds for Java, ~19 seconds for Python and Go.
Memory limit: 1GB.
$1 \le T \le 15$.
M = 106.
$-M \le Xi \le M$, for all i.
$-M \le Yi \le M$, for all i.
The pairs (Xi, Yi) are distinct.
$-10\times M \le Ci \le 10\times M$, for all i.
$-10\times M \le Di \le 10\times M$, for all i.
R = 1000.

Test set 1 (Visible)
Nmax = 1.

Test set 2 (Hidden)
Nmax = 10.

Testing Tool
You can use this testing tool to test locally or on our servers. To test locally, you will need to run the tool in parallel ith your code; you can use our interactive runner for that. For more information, read the Interactive Problems sectio of the FAQ.

Local Testing Tool
To better facilitate local testing, we provide you the following script. Instructions are included inside. You are encou aged to add more test cases for better testing. Please be advised that although the testing tool is intended to simulate he judging system, it is NOT the real judging system and might behave differently.

If your code passes the testing tool but fails the real judge, please check the Coding section of our FAQ to make sure that you are using the same compiler as us.

Sample Interaction

Note that the following sample interaction is for test set 1, in which there is always exactly one king.

```
 // Suppose that the judge has decided that in the first test case, the king
 // is at the coordinates (1, -2), and the requests will be (5, -1) and
 // (7, 7).
 t, nmax, m, r = readline_int_list()   // Reads 10 1 1000000 1000
 // Our solution decides (for whatever reason) to check (3, 3) first.
 printline 3 3 to stdout
 flush stdout
 result = readline_int()              // Reads 5
 // Our solution now decides (for whatever reason) to check (2, 0).
 printline 2 0 to stdout
 flush stdout
 result = readline_int()              // Reads 2
 // Our solution concludes that the king is at (3, -2), which is consistent
 // with the observed information so far, but unfortunately not correct.
 // Our solution moves on to the request phase.
 printline READY to stdout
 request_line = readline()            // Reads 5 -1
 printline 2 to stdout               // Wrong answer!
 request_line = readline()            // Reads ERROR
 exit                                // exits to avoid an ambiguous TLE error
```

Solution:

```
#include <bits/stdc++.h>

using namespace std;

template <typename A, typename B>
string to_string(pair<A, B> p);

template <typename A, typename B, typename C>
string to_string(tuple<A, B, C> p);

template <typename A, typename B, typename C, typename D>
string to_string(tuple<A, B, C, D> p);

string to_string(const string& s) {
  return '"' + s + '"';
}

string to_string(const char* s) {
  return to_string((string) s);
}

string to_string(bool b) {
  return (b ? "true" : "false");
}

string to_string(vector<bool> v) {
  bool first = true;
  string res = "{";
  for (int i = 0; i < static_cast<int>(v.size()); i++) {
    if (!first) {
      res += ", ";
    }
    first = false;
    res += to_string(v[i]);
  }
  res += "}";
  return res;
}

template <size_t N>
string to_string(bitset<N> v) {
  string res = "";
  for (size_t i = 0; i < N; i++) {
    res += static_cast<char>('0' + v[i]);
  }
  return res;
}

template <typename A>
string to_string(A v) {
  bool first = true;
  string res = "{";
  for (const auto &x : v) {
    if (!first) {
      res += ", ";
    }
    first = false;
    res += to_string(x);
  }
  res += "}";
  return res;
}

template <typename A, typename B>
string to_string(pair<A, B> p) {
  return "(" + to_string(p.first) + ", " + to_string(p.second) + ")";
}

template <typename A, typename B, typename C>
string to_string(tuple<A, B, C> p) {
  return "(" + to_string(get<0>(p)) + ", " + to_string(get<1>(p)) + ", " + to_string(get<2>(p)) + ")";
}

template <typename A, typename B, typename C, typename D>
string to_string(tuple<A, B, C, D> p) {
  return "(" + to_string(get<0>(p)) + ", " + to_string(get<1>(p)) + ", " + to_string(get<2>(p)) + ", " + to_string(get<3
(p)) + ")";
}
```

```
void debug_out() { cerr << endl; }

template <typename Head, typename... Tail>
void debug_out(Head H, Tail... T) {
  cerr << " " << to_string(H);
  debug_out(T...);
}

#ifdef LOCAL
#define debug(...) cerr << "[" << #__VA_ARGS__ << "]:", debug_out(__VA_ARGS__)
#else
#define debug(...) 42
#endif

int main() {
  ios::sync_with_stdio(false);
  cin.tie(0);
  cout << fixed << setprecision(17);
  int t, nmax, m, r;
  cin >> t >> nmax >> m >> r;
  for (int qq = 1; qq <= t; qq++) {
    const int MAX = 4 * m;
    int qs = 0;
    auto Query = [&](int x, int y) {
      ++qs;
      cout << x << " " << y << endl;
      int foo;
      cin >> foo;
//      debug(x, y, foo);
      return foo;
    };
    auto QueryPM = [&](int xpy, int xmy) {
//      debug(xpy, xmy);
      assert((xpy + xmy) % 2 == 0);
      int x = (xpy + xmy) / 2;
      int y = (xpy - xmy) / 2;
      return 2 * Query(x, y);
    };
    int n = Query(0, MAX) - Query(0, MAX - 1);
    auto AskXpy = [&](int xpy) {
      int xmy = (xpy % 2 != 0 ? MAX - 1 : MAX);
      int p0 = QueryPM(xpy, xmy) + (xmy == MAX - 1 ? n : 0);
      ++xpy;
      xmy = (xpy % 2 != 0 ? MAX - 1 : MAX);
      int p1 = QueryPM(xpy, xmy) + (xmy == MAX - 1 ? n : 0);
//      debug("xpy", xpy - 1, p1 - p0);
      return p1 - p0;
    };
    auto AskXmy = [&](int xmy) {
      int xpy = (xmy % 2 != 0 ? MAX - 1 : MAX);
      int p0 = QueryPM(xpy, xmy) + (xpy == MAX - 1 ? n : 0);
      ++xmy;
      xpy = (xmy % 2 != 0 ? MAX - 1 : MAX);
      int p1 = QueryPM(xpy, xmy) + (xpy == MAX - 1 ? n : 0);
//      debug("xmy", xmy - 1, p1 - p0);
      return p1 - p0;
    };
    vector<int> xpy;
    vector<int> xmy;
    for (int rot = 0; rot < 2; rot++) {
      for (int i = 0; i < n; i++) {
        int low = -MAX;
        int high = MAX;
        while (low < high) {
          int mid = low + (high - low) / 2;
          int v = (rot == 0 ? AskXpy(mid) : AskXmy(mid));
          if (v <= -n + 2 * i) {
            low = mid + 1;
          } else {
            high = mid;
          }
        }
        xpy.push_back(low);
      }
      swap(xpy, xmy);
    }
    debug(qs, n, xpy, xmy);
    cout << "READY" << endl;
    while (true) {
      string foo;
      cin >> foo;
      if (foo == "ERROR") {
        cerr << "NO SAD" << '\n';
        return 0;
      }
      if (foo == "DONE") {
        break;
      }
      stringstream ss;
      ss << foo;
      int x, y;
      ss >> x;
      cin >> y;
      int ans = 0;
      for (int i = 0; i < n; i++) {
        ans += abs(xpy[i] - (x + y)) + abs(xmy[i] - (x - y));
      }
      ans /= 2;
      cout << ans << endl;
//      debug(x, y, ans);
    }
  }
  return 0;
```

}