Problem setup

We are given a set of $N + M$ house locations. This set of locations is divided into two classes, namely N *nobles* houses and M *common* houses. Each house location is represented by a data point lying on the $\mathbb{R}^2$ plane. The objective of the *Lannister* problem is to find the coefficients of $L1$ and $L2$: two non-horizontal and orthogonal lines lying in the $\mathbb{R}^2$ plane satisfying and/or minimizing the following constraints:

- Constraint 1: $L1$ must separate the two classes of houses, having all the noble house locations on its left and the common house locations on its right.
- Constraint 2: The sum of every horizontal distance x between any house location and its intercept point at the same y height on $L1$ must be less than a given threshold s.
- Constraint 3: $L2$ position has to minimize the greatest distance y between any house location and its intercept point at the same x coordinate on $L2$.

Problem Modeling

We are seeking the line coefficients of $L1$ and $L2$. Because we know that they are orthogonal, we can reduce the problem to that of finding the following correlated coefficients. We can even check that $m_1 m_2 = -\frac{b}{a}\frac{a}{b} = -1$ which is indeed the orthogonality criterion.

$$L1 : ax + by + c_S$$

$$L2 : -bx + ay + c_W$$

Constraint 1

The first constraint is a linear separation problem using $L1$. This can be easily optimized with CGAL Linear Program (LP) solver. To be more explicit, we are optimizing $ax + by + c_S$ in such a way that the N noble house locations are on the left of $L1$ and the M common house locations are on the right of $L1$. This is translated to the following set of constraints. Note that subscripts n and c indicate belonging to one of the two classes.

$$\forall (x_n, y_n) : ax_n + by_n \leq c_S$$

$$\forall (x_c, y_c) : ax_c + by_c \geq c_S \Leftrightarrow -ax_c - by_c \leq -c_S$$

Constraint 2

More formally, we require $\sum_{i\in h} |x_i - x_{L_1}| \leq s$ where h denote the set of all houses combined and x_{L_1} the horizontal $L1$ intercept. The situation looks a bit tricky but we know that if the first constraint is achieved, all the noble houses are on the left of $L1$ and the common ones on the right of it. We can therefore get rid of the absolute value and define $|x_i - x_{L_1}|$ distance d_i according to x_i class. We can further simplify the calculation with the line equation definition noting that x_i and x_{L_1} have the same height $y_i = y_{L_1}$: $x_{L_1} = \frac{by_{L_1} - c_S}{a} = \frac{by_i - c_S}{a}$

$$d_n = x_{L_1} - x_n = \frac{by_n - c_S}{a} - x_n$$
$$d_c = x_c - x_{L_1} = x_c - \frac{by_c - c_S}{a}$$

Now the problem is to formulate a CGAL constraint that enforces $\sum_{i\in h} d_i \leq s$. For that purpose we can unfold the inequality and see that what comes out is a valid CGAL constraint expression because we know every variable's coefficients:

$$\begin{aligned}
&\sum_{i\in h} d_i \\
&= \sum_n d_n + \sum_c d_c \\
&= \sum_n \Big(\frac{by_n - c_S}{a} - ax_n\Big) + \sum_c \Big(ax_c - \frac{by_c - c_S}{a}\Big) \\
&= \Big(\sum_c x_c - \sum_n x_n\Big) + \frac{b}{a}\Big(\sum_n y_n - \sum_c y_c\Big) + \frac{1}{a}c_S(M-N) \quad \leq \quad s \\
&\Leftrightarrow \\
&= a\Big(\sum_c x_c - \sum_n x_n\Big) + b\Big(\sum_n y_n - \sum_c y_c\Big) + c_S(M-N) \quad \leq \quad as \\
&\Leftrightarrow \\
&= a\Big(\sum_c x_c - \sum_n x_n - s\Big) + b\Big(\sum_n y_n - \sum_c y_c\Big) + c_S(M-N) \quad \leq \quad 0
\end{aligned}$$

Constraint 3

Below is the formal description of the problem we are facing with this last constraint optimization. Note that we are now forced to keep the introduced absolute value as we have no clue of whether a house location is on the left or on the right of L_2.

$$\min \Big(\max_{i \in h} \quad |y_i - y_{L_2}| \Big)$$

Because CGAL doesn't allow us to deal with a maximization problem, we have to figure out another trick for the inner optimization. Here it is: we introduce a new variable l which represents the maximal length we are searching for. The inner problem then simply reduces to $\forall i \in h : |y_i - y_{L_2}| \leq l$. We can again unfold $|y_i - y_{L_2}|$ and see that it gives us the following valid CGAL constraint to feed to our LP solver:

$$|y_i - y_{L_2}| \leq l \quad \Leftrightarrow \quad -l \leq y_i - y_{L_2} \leq l \quad \Leftrightarrow \quad -l \leq y_i + \frac{c_W - bx_i}{a} \leq l$$

For the sake of completeness, if we unfold only one of the 2 side of the inequality, we find the last and extra subtlety we face in *Lannister* problem. Here is what happens:

$$y_i + \frac{c_W - bx_i}{a} \leq l \quad \Leftrightarrow \quad ay_i - bx_i + c_W \leq al$$

Notice that this is not anymore a linear problem. Indeed, because we have introduced l as a new *variable* the product al bring our inequality to a quadratic form. To get around this concern, we can set $a = 1$. This will simplify the equation of constraint 2. Note that it also enforce $L1$ and $L2$ to be non-horizontal, which was required initially in the problem setup.

Algorithm Design

We can finally write our routine and wrap up what we discussed above. We therefore have 5 variables (a, b, c_S, c_W, l) to feed to the CGAL LP optimizer under the following constraints:

$$\forall (x_n, y_n) : ax_n + by_n \leq c_S \tag{1}$$

$$\forall (x_c, y_c) : -ax_c - by_c \leq -c_S \tag{2}$$

$$\forall (x_h, y_h) : bx_h - c_W - l \leq y_h \tag{3}$$

$$\forall (x_h, y_h) : -bx_h + c_W - l \leq -y_h \tag{4}$$

$$b\Big(\sum_n y_n - \sum_c y_c \Big) + c_S(M - N) \leq s - \Big(\sum_c x_c - \sum_n x_n \Big) \tag{5}$$

Code submission: 100%

```
#include <CGAL/QP_models.h>
#include <CGAL/QP_functions.h>
#include <CGAL/Gmpz.h>
#include <vector>

using namespace std;

typedef CGAL::Gmpz ET;
typedef CGAL::Quadratic_program<long> Program;
typedef CGAL::Quadratic_program_solution<ET> Solution;

void solve() {
  int n, m; long s;
  cin >> n >> m >> s;

  vector<pair<int, int>> nobles(n);
  vector<pair<int, int>> commons(m);

  long sum_nx = 0, sum_ny = 0;
  long sum_cx = 0, sum_cy = 0;

  for (int i = 0; i < n; i++) {
    int x, y; cin >> x >> y;
    sum_nx += x; sum_ny += y;
    nobles[i] = make_pair(x, y);
  }

  for (int i = 0; i < m; i++) {
    int x, y; cin >> x >> y;
    sum_cx += x; sum_cy += y;
    commons[i] = make_pair(x, y);
  }

  Program lp(CGAL::SMALLER, false, 0, false, 0);
  int a = 0, b = 1, cs = 2, cw = 3, l = 4, row = 0;

  // Enforce a == 1
  lp.set_l(a, true, 1);
  lp.set_u(a, true, 1);

  // Cersei constraint:

  for (int i = 0; i < n; i++) {
```

```
    lp.set_a(a, row, nobles[i].first);
    lp.set_a(b, row, nobles[i].second);
    lp.set_a(cs, row++, 1);
  }

  for (int i = 0; i < m; i++) {
    lp.set_a(a, row, -commons[i].first);
    lp.set_a(b, row, -commons[i].second);
    lp.set_a(cs, row++, -1);
  }

  Solution s1 = CGAL::solve_linear_program(lp, ET());

  if (s1.is_infeasible()) {
    cout << "Yuck!" << endl;
    return;
  }

  // Tywin constraint:

  if (s != -1) {
    lp.set_a(cs, row, m - n);
    lp.set_a(b, row, sum_cy - sum_ny);
    lp.set_b(row++, s - (sum_cx - sum_nx));

    Solution s2 = CGAL::solve_linear_program(lp, ET());

    if (s2.is_infeasible()) {
      cout << "Bankrupt!" << endl;
      return;
    }
  }

  // Jaime optimization:

  int hx, hy = 0;

  for (int i = 0; i < n + m; i++) {
    if (i < n) { hx = nobles[i].first; hy = nobles[i].second; }
    else { hx = commons[i-n].first; hy = commons[i-n].second; }

    lp.set_a(b, row, hx);
    lp.set_a(cw, row, -1);
    lp.set_a(l, row, -1);
```

```
    lp.set_b(row++, hy);

    lp.set_a(b, row, -hx);
    lp.set_a(cw, row, 1);
    lp.set_a(l, row, -1);
    lp.set_b(row++, -hy);
  }

  // Enforce positive length
  lp.set_l(l, true, 0);

  // Minimize length
  lp.set_c(l, 1);

  Solution s3 = CGAL::solve_linear_program(lp, ET());
  cout << fixed << setprecision(0) << ceil(CGAL::to_double(s3.objective_value())) << endl;
}

int main() {
  ios_base::sync_with_stdio(false);
  int t; cin >> t;
  for (int i = 0; i < t; i++) solve();
}
```