```cpp
/*
    | Monotone Chain |
    Finding Convex Hull in O(n*log(n))
    Source: USACO Guide
    State: Idk its from USACO Guide
*/

#include <bits/stdc++.h>
using namespace std;

using pii = pair<int, int>;

vector<pii> points;
vector<pii> hull;

// cross product, the signed area of these there points
int area(pii O, pii P, pii Q) {
        return (P.first - O.first) * (Q.second - O.second) -
               (P.second - O.second) * (Q.first - O.first);
}

void monotone_chain() {
        // sort with respect to the x and y coordinates
        sort(points.begin(), points.end());
        // distinct the points
        points.erase(unique(points.begin(), points.end()), points.end());
        int n = points.size();

        // 1 or 2 points are always in the convex hull
        if (n < 3) {
                hull = points;
                return;
        }

        // lower hull
        for (int i = 0; i < n; i++) {
                // if with the new point points[i], a right turn will be formed,
                // then we remove the last point in the hull and test further
                while (hull.size() > 1 &&
                       area(hull[hull.size() - 2], hull.back(), points[i]) <= 0)

                        hull.pop_back();
                // otherwise, add the point to the hull
                hull.push_back(points[i]);
        }

        // upper hull, following the same logic as the lower hull
        auto lower_hull_length = hull.size();
        for (int i = n - 2; i >= 0; i--) {
```

```cpp
                // we can only remove a point if there are still points left in the
                // upper hull
                while (hull.size() > lower_hull_length &&
                       area(hull[hull.size() - 2], hull.back(), points[i]) <= 0)
                        hull.pop_back();
                hull.push_back(points[i]);
        }
        // delete point[0] that has been added twice
        hull.pop_back();
}

int main() {
        cin.tie(0)->sync_with_stdio(false);

        int n;
        cin >> n;
        while (n != 0) {
                points.assign(n, {});
                hull = {};
                for (auto &p : points) cin >> p.first >> p.second;
                monotone_chain();

                cout << hull.size() << "\n";
                for (auto &p : hull) cout << p.first << " " << p.second << "\n";

                cin >> n;
        }

        return 0;
}
```