Problem

The first two paragraphs (not counting this one) of this problem and "Juggle Struggle: Part 2" are identical. The prob ems can otherwise be solved independently; you do not need to read or solve one in order to read or solve the other.

As manager of the Graceful Chainsaw Jugglers group, you have decided to spice the show up a bit. Instead of havin each juggler individually juggle their own chainsaws, you want them to form pairs, with each pair throwing the chai saws back and forth to each other. In this new performance, 2 × N jugglers will be on stage at the same time, arrange into N pairs, with each juggler belonging to exactly one pair.

You think the show will be more impressive if the chainsaws being juggled by different pairs of jugglers are at risk f collision. Let the stage be a two-dimensional plane, and let the straight line segment in that plane that connects the ositions of two jugglers in a pair be called the pair's juggling path. When two juggling paths instersect, we say the ch insaws juggled by those pairs are at risk of collision. We call the spatial positions and the pairings of the jugglers an rrangement. An arrangement is magnificent if every two pairs of jugglers' chainsaws are at risk of collision.

After a lot of thinking and designing, you came up with a magnificent arrangement. You wrote down the positions o the jugglers on the stage and the pairings of the jugglers on a piece of paper. Unfortunately, a bad chainsaw throw c t the paper in half, and you have lost the half with the pairings. Since the stage decorations have already been design d based on the positions of the jugglers, those positions cannot be changed. The show's highly anticipated debut is a ere few hours away, so you need to find a magnificent arrangement that works! Given every juggler's position on a t o-dimensional stage, find a pairing of them that yields a magnificent arrangement.

Input

The first line of the input gives the number of test cases, T. T test cases follow. Each test case starts with one line co taining a single integer N, the number of pairs of jugglers. Then, 2 × N lines follow. The i-th of these lines contains t o integers Xi and Yi, representing the coordinates of the position of the i-th juggler.

Output

For each test case, output one line containing Case #x: j1 j2 ... j2 × N, representing that jugglers i and ji are to be pai ed together, for every i. Notice that jji = i for every i.

Limits

Memory limit: 1GB.
$-109 \leq Xi \leq 109$, for all i.
$-109 \leq Yi \leq 109$, for all i.
No three juggler positions are collinear. (Note that this also implies that no two jugglers are in the same position.)
There exists at least one way to pair the jugglers such that the resulting arrangement is magnificent.

Test set 1 (Visible)

Time limit: 20 seconds.
$1 \leq T \leq 100$.
$2 \leq N \leq 100$.

Test set 2 (Hidden)

Time limit: 60 seconds.
$1 \leq T \leq 10$.
$2 \leq N \leq 105$.

Sample

Input

Output

```
3
2
-1 -1
-1 1
1 1
1 -1
3
1 2
2 1
2 3
3 1
3 3
4 2
3
7 1
1 1
7 2
5 5
3 5
1 2
```

```
Case #1: 3 4 1 2
Case #2: 6 5 4 3 2 1
Case #3: 5 4 6 2 1 3
```

In Sample Case #1, the jugglers' positions form a square. The only valid solution is to pair up jugglers 1 and 3, and p ir up jugglers 2 and 4.

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

struct Point {
  int x;
  int y;
  int id;
};

inline int Side(int x, int y) {
  return (y > 0 || (y == 0 && x >= 0));
}

mt19937 rng(58);

int main() {
  ios::sync_with_stdio(false);
  cin.tie(0);
  cout << fixed << setprecision(17);
  int tt;
  cin >> tt;
  for (int qq = 1; qq <= tt; qq++) {
    cout << "Case #" << qq << ":";
    int n;
    cin >> n;
    n *= 2;
    vector<Point> p(n);
    for (int i = 0; i < n; i++) {
      cin >> p[i].x >> p[i].y;
      p[i].id = i;
    }
```

```
auto FindPair = [&](int i, vector<int> ids) {
  vector<Point> q;
  for (int j : ids) {
    q.push_back({p[j].x - p[i].x, p[j].y - p[i].y, p[j].id});
    q.push_back({p[i].x - p[j].x, p[i].y - p[j].y, ~p[j].id});
  }
  sort(q.begin(), q.end(), [&](const Point& qj, const Point& qk) {
    int xj = qj.x;
    int yj = qj.y;
    int xk = qk.x;
    int yk = qk.y;
    int sj = Side(xj, yj);
    int sk = Side(xk, yk);
    if (sj != sk) {
      return sj == 1;
    }
    long long vmul = (long long) xj * yk - (long long) xk * yj;
    return (vmul > 0);
  });
  int L = 0;
  for (int j = 0; j < (int) q.size(); j++) {
    if (Side(q[j].x, q[j].y) == 0) {
      break;
    }
    L += (q[j].id >= 0);
  }
  int R = (int) ids.size() - L;
  int pai = -1;
  for (int j = 0; j < (int) q.size(); j++) {
    if (q[j].id >= 0) {
      --L;
      if (L == R) {
        pai = q[j].id;
        break;
      }
      ++R;
    } else {
      --R;
      ++L;
    }
  }
  return pai;
};
vector<int> res(n, -1);
vector<int> a(n);
function<bool(int, int)> Solve = [&](int from, int to) {
  if (from == to) {
    return true;
  }
  int i = a[from];
  vector<int> ids;
  for (int j = from + 1; j < to; j++) {
```

```
  ids.push_back(a[j]);
 }
 int ip = FindPair(i, ids);
 if (ip == -1) {
  return false;
 }
 res[i] = ip;
 res[ip] = i;
 long long A = p[ip].y - p[i].y;
 long long B = p[i].x - p[ip].x;
 long long C = -A * p[i].x - B * p[i].y;
 if (from == 0 && to == n) {
  vector<int> a0, a1;
  for (int j = from + 1; j < to; j++) {
   if (a[j] == ip) {
    continue;
   }
   long long z = A * p[a[j]].x + B * p[a[j]].y + C;
   if (z > 0) {
    a0.push_back(a[j]);
   } else {
    a1.push_back(a[j]);
   }
  }
  if (a0.size() != a1.size()) {
   return false;
  }
  int ptr = from + 2;
  for (int j : a0) {
   a[ptr++] = j;
  }
  for (int j : a1) {
   a[ptr++] = j;
  }
  return Solve(from + 2, to);
 } else {
  int mid = (from + to) >> 1;
  vector<int> a0, a1, a2, a3;
  for (int j = from + 1; j < to; j++) {
   if (a[j] == ip) {
    continue;
   }
   long long z = A * p[a[j]].x + B * p[a[j]].y + C;
   if (z > 0) {
    if (j < mid) {
     a0.push_back(a[j]);
    } else {
     a2.push_back(a[j]);
    }
   } else {
    if (j < mid) {
     a1.push_back(a[j]);
```

```
          } else {
            a3.push_back(a[j]);
          }
        }
      }
      if (a0.size() != a3.size() || a1.size() != a2.size()) {
        return false;
      }
      int ptr = from + 2;
      for (int j : a0) {
        a[ptr++] = j;
      }
      for (int j : a3) {
        a[ptr++] = j;
      }
      int b0 = ptr;
      for (int j : a1) {
        a[ptr++] = j;
      }
      for (int j : a2) {
        a[ptr++] = j;
      }
      if (!Solve(from + 2, b0)) {
        return false;
      }
      if (!Solve(b0, to)) {
        return false;
      }
      return true;
    }
  };
  while (true) {
    iota(a.begin(), a.end(), 0);
    shuffle(a.begin(), a.end(), rng);
    fill(res.begin(), res.end(), -1);
    if (Solve(0, n)) {
      break;
    }
  }
  for (int i = 0; i < n; i++) {
    cout << " " << res[i] + 1;
  }
  cout << '\n';
 }
 return 0;
}
```