```
struct Point {
  int x,y;
  Point() { x = y = -1; };
  Point(int x,int y) : x(x) ,y(y) {};
  void calibrate() { x--,y--; };

  int X(int n) { return x << 1; };
  int Y(int n) { return (n + y) << 1; };
};

istream& operator >> (istream&x,Point&w) { x >> w.x >> w.y; w.calibrate();
 return x; }

class _2_SAT {
  public:
    _2_SAT(int n,int m,int N) : n(n) ,m(m) ,N(N) {
      int NN = ((n+m)<<1) + 100;
      ed = new vector<int> [NN];
      rev = new vector<int> [NN];
      mark = new bool [NN]();
      scc = new int[NN]();
      ans = new int[NN]();
      for(int i = 0;i<NN;i++) mark[i] = false;
    }

    void add_condition(Point x,Point y) {
      if(x.X(n) == y.X(n) && y.Y(n) == x.Y(n)) return ;
      CONS.push_back({x,y});
      if(x.X(n) == y.X(n))
        create_edge(other(x.X(n),y.y< x.y));
      else if(x.Y(n) == y.Y(n))
        create_edge(other(x.Y(n),y.x< x.x));
      else {
//        create_edge(other(x.X(n),y.y>x.y),other(y.X(n),y.y<x.y));
//        create_edge(other(x.X(n),y.y>x.y),other(x.Y(n),y.x<x.x));
//        create_edge(other(y.Y(n),y.x>x.x),other(y.X(n),y.y<x.y));
//        create_edge(other(y.Y(n),y.x>x.x),other(x.Y(n),y.x<x.x));
//        create_edge(other(y.X(n),y.y>x.y),other(x.X(n),
        create_edge(other(x.X(n),y.y<x.y),other(y.X(n),y.y<x.y));
        create_edge(other(x.Y(n),y.x<x.x),other(y.Y(n),y.x<x.x));
        create_edge(other(x.Y(n),y.x<x.x),other(x.X(n),y.y<x.y));
        create_edge(other(y.Y(n),y.x<x.x),other(y.X(n),y.y<x.y));
      }
    }

    bool satisfiable() {
      for(int i = 0;i<n+m;i++) {
        if(!mark[i<<1]) topo_dfs(i<<1);
        if(!mark[other(i<<1)]) topo_dfs(other(i<<1));
      }
      reverse(topo.begin(),topo.end());
```

```
      for(auto w : topo)
        if(mark[w]) scc_dfs(w,w);
      for(int i = 0;i<n+m;i++)
        if( scc[i<<1] == scc[other(i<<1)] ) return false;
      return true;
    }
  private:
    const int n,m;
    const int N;
    vector<int> *ed;
    vector<int> *rev;
    vector<int> topo;
    bool *mark;
    int *scc;
    vector<pair<Point,Point>> CONS;
    int*ans;
    void create_edge(int a,int b = -1) {
      if(!~b) b = a;
      ed[other(a)].push_back(b);
      ed[other(b)].push_back(a);
      rev[a].push_back(other(b));
      rev[b].push_back(other(a));
    }

    inline int other (int x,bool w = true) {
      if(!w) return x;
      return x ^ 1;
    }

    void topo_dfs(int v) {
      mark[v] = true;
      for(auto w : ed[v]) if(!mark[w]) topo_dfs(w);
      topo.push_back(v);
    }
    void scc_dfs(int v,int I ) {
      mark[v] = false;
      scc[v] = I;
      for(auto w : rev[v]) if(mark[w]) scc_dfs(w,I);
    }
};
```