[Problem G]　レーザー光の反射

この問題の場合、計算結果には誤差が含まれるので、結果の比較には誤差 0.001 以内で一致しているかを判定するプログラムを作成してある。

鏡の枚数は５枚以下で、反射回数も６回未満なので、６回未満の鏡の反射順各々に対して、実現可能性をチェックすると共に、実現可能なものの経路長の中から最短経路を求める。

0 回反射の場合

実現可能性：　レーザー光源とターゲットを結ぶ線分が、どの鏡とも交わらない
→　２線分の交差判定

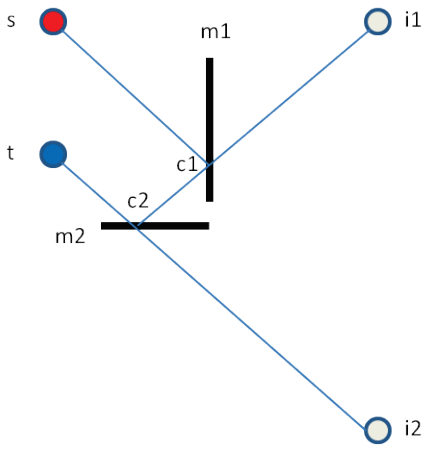
経路の距離：　レーザー光源とターゲットの距離

1 回反射の場合

実現可能性：　光源の鏡による鏡像とターゲットを結ぶ線分が鏡と交差する。
交差点を c とする時、線分 sc と線分 ct は他の鏡と交わらない

経路の距離：　鏡像とターゲットの距離

２回反射の場合



光源 s の鏡 m1 による鏡像を i1、鏡像 i1 の鏡 m2 による鏡像を i2 とする。

実現可能性：　ターゲット t と鏡像 i2 を結ぶ線分は鏡 m2 と交差する。
交差点を c2 とすると、c2 と鏡像 i1 を結ぶ線分は鏡 m1 と交差する。
その交差点を c1 とする。
線分 s – c1, c1 – c2, c2 – t は他の鏡と交差しない。

経路の距離：　線分 t – i2 の距離

３回以上反射する場合も同様に計算できる。

プログラム例

```
// ACM-ICPC 2010 Japan Online Contest Problem G
//
//      ファイル名: pg.c
//      コンパイル方法: cc pg.c
//      実行方法: ./a.out < G0 > G0.result など
//      チェック方法: ./compare
//
// アルゴリズム概略
// 鏡の枚数は 5 枚以下で、反射回数も 6 回未満なので、
// このようなケースを実現しうる鏡の反射順を全て生成し、
// その反射順が実現可能かどうかをチェックし、
// 実現可能なものについての経路長から最短経路長を求める。
// 選択された反射順に対して、まずレーザー発射位置の最初の鏡による鏡像を求め
// 次に、その鏡像の 2 番目の鏡により鏡像を求める。これを繰り返し、
// 最後の鏡による鏡像の位置もとめる。この位置とターゲットの位置との距離が
// 経路長となる(経路が実現可能な場合)。
// 実現可能性は、経路の反射位置が対応する鏡の上に存在し、間に経路を妨害する
// 他の鏡が存在しないことを経路に沿ってチェックすることで判定できる。
//(本プログラムでは、このチェックを経路の逆順に沿って行っている。)

#include <stdio.h>
#include <math.h>

struct point_type {          // 点またはベクトルの x,y 座標を格納する構造体
  double x;
  double y;
};
typedef struct point_type point;      // 点の x,y 座標
typedef struct point_type vector_t;   // ベクトルの x,y 座標
struct line_type { // 線分の 2 点を格納する構造体
  point p;
  point q;
};
typedef struct line_type line;                    // 線(分)の 2 点

int n;                    // 1 <= n <= 5 鏡の枚数
line mirror[5];           // 鏡の位置データ
point target;             // ターゲットの点
point laser;              // レーザー光発射点
double length;            // レーザー光線の最短経路長
int nr;                   // 鏡の反射回数
int mno[5];               // 反射する鏡の番号（反射順）
```

```
point m_image[6];// m_image[0] = laser, m_image[i] は i 番目の反射のイメージ

double in_prod(vector_t a, vector_t b)            // 内積を求める
{
   return a.x*b.x + a.y*b.y;
}

double out_prod(vector_t a, vector_t b)           // 外積を求める
{
   return a.x*b.y - a.y*b.x;
}

double angle(vector_t a, vector_t b)  // ベクトル a に対するベクトル b の角度を求める
{
   return atan2(out_prod(a,b), in_prod(a,b));
}

double distance(point a, point b)       // 2 点間の距離を求める
{
   return sqrt((a.x - b.x)*(a.x - b.x) + (a.y - b.y)*(a.y - b.y));
}

point mirror_image(line m, point s)  // 点 s の鏡 m に対する鏡像を求める
{
   point image;
   double a_m, a_s, a_i, d;
   a_m = atan2(m.q.y - m.p.y, m.q.x - m.p.x);   // x 軸に対する鏡の角度
   a_s = atan2(s.y - m.p.y, s.x - m.p.x);       // x 軸に対するベクトル ps の角度
   a_i = a_m - (a_s - a_m);                     // x 軸に対するベクトル pi の角度
   d = distance(m.p, s);
   image.x = d*cos(a_i)+m.p.x;
   image.y = d*sin(a_i)+m.p.y;
   return image;
}

// 線分 ab と線分 cd の交差判定。交差->1, 交差しない->0
int is_crossing(point a, point b, point c, point d)
{
   double ta,tb,tc,td;
   tc = (a.x - b.x)*(c.y - a.y) + (a.y - b.y)*(a.x - c.x);
   td = (a.x - b.x)*(d.y - a.y) + (a.y - b.y)*(a.x - d.x);
   if(tc*td >= 0) return 0;
   ta = (c.x - d.x)*(a.y - c.y) + (c.y - d.y)*(c.x - a.x);
   tb = (c.x - d.x)*(b.y - c.y) + (c.y - d.y)*(c.x - b.x);
   if(ta*tb < 0) return 1;
   else return 0;
}

// 直線 ab と直線 cd の交点を求める。 2 直線は平行でないと仮定。
point cross_point(point a, point b, point c, point d)
{
   double acx, acy, bunbo, r, s;
   point cross;
   acx = c.x - a.x;
   acy = c.y - a.y;
   bunbo = (b.x-a.x)*(d.y-c.y) - (b.y-a.y)*(d.x-c.x);
   r = ((d.y-c.y)*acx - (d.x-c.x)*acy)/bunbo;
```

```
  s = ((b.y-a.y)*acx - (b.x-a.x)*acy)/bunbo;
  cross.x = a.x + r*(b.x - a.x);
  cross.y = a.y + r*(b.y - a.y);
  return cross;
}

// 与えられた鏡の反射回数に対して、可能性のある反射順を全て生成し、
// その反射順によりレーザー光がターゲットに到達できる場合は
// その経路の距離を求め、既に求まっている経路長よりも短ければ
// 経路長を更新する
// level = 現時点での反射回数　nr = 今回チェックする反射回数
void check(int level, int nr)
{
  int i;
  if(level<nr){                    // 今回チェックする反射回数に達していなければ
    for(i=0; i<n; i++){        // 各鏡に対して順番に
      if(level>0 && mno[level-1] == i) continue;      // 直前に反射した鏡はスキップ
      mno[level] = i;            // それを今回反射する鏡とする
      // 直前の鏡によるレーザー発射装置の鏡像に対して、
      // この鏡での反射による鏡像を求め、それを level+1 番目の鏡像とする
      m_image[level+1] = mirror_image(mirror[i],m_image[level]);
      check(level+1, nr);     // level を一つ増やして再帰的に check を呼び出すことにより
                                  // 次に反射する鏡を決めていく
    }
  }
  else{                // level == nr　反射する順に nr 個の鏡が選ばれた
    double d;
    point t = target;
    int j,k;
    point cp;
    d = distance(m_image[level],t);        // 最後の鏡によるレーザー発射装置の鏡像位置と
                                               // ターゲットとの距離 d をもとめる
                                               // 選択した鏡の順に反射してターゲットに
                                               // 到達可能なら、d はその経路の距離になる
    if(d >= length) return;   // d が既に求まっている最短経路長以上ならばスキップ
    for(j=level; j>0; j--){
      if(is_crossing(mirror[mno[j-1]].p, mirror[mno[j-1]].q,
                          m_image[j], t)==0) break;   // j 番目の鏡による反射が実現不可なら
                                                          // チェック終了
      // その鏡上の反射位置 cp を求める
      cp = cross_point(mirror[mno[j-1]].p, mirror[mno[j-1]].q, m_image[j], t);
      // cp とこの鏡の反射によるターゲット位置 t との間に他の鏡があれば実現不可
      for(k=0; k<n; k++){
        if(k== mno[j-1]) continue;   // cp がある鏡は OK
        if(j<level){                        // 最後に反射する鏡で無ければ
          if(k==mno[j]) continue;     // ターゲット位置 t がある鏡も OK
        }
        // 間に一つでも他の鏡があれば実現不可
        if(is_crossing(mirror[k].p, mirror[k].q, cp, t)) break;
      }
      if(k<n) break;       // 間に他の鏡が入っていたので実現不可
      t = cp;                  // 現在の鏡上の反射位置 cp を次のターゲットとし、
                                  // 一つ手前の鏡による反射の実現可能性をチェック
    }
    if(j>0) return;        // j 番目の鏡による反射が実現不可
```

```
      // 最初の鏡による反射の実現可能性をチェック
      for(k=0; k<n; k++){
        if(nr>0){              // 鏡による反射が 1 回以上あるとき
          if(k==mno[0]) continue;        // 最初の鏡は OK
        }
        // この時点で t は最初の鏡の反射位置
        // レーザー発射位置と t の間に他の鏡があれば実現不可
        if(is_crossing(mirror[k].p, mirror[k].q, laser, t)) break;
      }
      if(k==n)      // 最後まで実現不可にならなかったので
        length = d;    // 最短経路長を更新（この時点では必ず d < length である）
    }
}

void solve()
{
    // 座標値は 0～100 で鏡による反射回数は 6 回未満なので、
    // 最短経路長は sqrt(2)*100*6 < 1000 以下である
    length = 1000;             // とりあえず 1000 とし、より短い経路が見つかれば更新する
    m_image[0] = laser;    // レーザー光発射位置をセット
    for(nr=0; nr<=5; nr++)       // 鏡の反射回数 0～5 に対して
      check(0,nr);                    // レーザー光線の最短経路長を計算し、
                                      // より短いものが見つかれば length を更新
    printf("%lf¥n", length);   // 求まった最短経路長をプリント
}

int main()
{
    int i;
    while(1){
      scanf("%d", &n);       // 鏡の枚数 n を入力
      if(n==0) break;        // n==0 なら終了
      for(i=0; i<n; i++){        // 各鏡に対して、両端の座標を読み込む
        scanf("%lf %lf %lf %lf", &mirror[i].p.x, &mirror[i].p.y,
                &mirror[i].q.x, &mirror[i].q.y);
      }
      scanf("%lf %lf", &target.x, &target.y);        // 目標物の座標を読み込む
      scanf("%lf %lf", &laser.x, &laser.y);          // レーザー光発生装置の位置を読み込む
      solve();                    // レーザー光の最短経路の長さを求める
    }
}
```