#include <iostream>

using namespace std;

class Point {

public: int x\_cor,y\_cor;

void enterPointCoordinate(){

cout<<"\t\tEnter the X-coordinate: ";

cin>>x\_cor;

cout<<"\t\tEnter the Y-coordinate: ";

cin>>y\_cor;

}

void displayPoint(){

cout<<"("<<x\_cor<<" ,"<<y\_cor<<")";

}

};

class Line{

public: template <class A, class B>

void line(Point& a, Point& b){

cout<<"\tFor Starting Point: "<<endl;

a.enterPointCoordinate();

cout<<"\tFor End Point: "<<endl;

b.enterPointCoordinate();

}

};

class TurnTest{

public: template <class A, class B, class C>

void enterPointForTurnTest(A& a, B& b, C& c){

cout<<"\tGiven Line Segment: "<<endl;

cout<<"\t Starting Point: "<<endl;

a.enterPointCoordinate();

cout<<"\t End Point: "<<endl;

b.enterPointCoordinate();

cout<<"\n\tEnter the third point for which turn test is to be done: "<<endl;

c.enterPointCoordinate();

turnTest(a,b,c);

}

template <class A, class B, class C>

void turnTest(A& a, B& b, C& c){

double area;

area=0.5\*(a.x\_cor\*(b.y\_cor-c.y\_cor)+b.x\_cor\*(c.y\_cor-a.y\_cor)+c.x\_cor\*(a.y\_cor-b.y\_cor));

if(area>0){

cout<<"\n\t\t The point ("<<c.x\_cor<<","<<c.y\_cor<<") is LEFT to ";

cout<<"the line segment from: ";

a.displayPoint();

cout<<" to ";

b.displayPoint();

}

else if(area<0){

cout<<"\n\t\t The point ("<<c.x\_cor<<","<<c.y\_cor<<") is RIGHT to ";

cout<<"the line segment from: ";

a.displayPoint();

cout<<" to ";

b.displayPoint();

}

else{

cout<<"\n\t\t The point ("<<c.x\_cor<<","<<c.y\_cor<<") is COLLINEAR";

}

}

};

class LineIntersection{

public: int flag;

template<class A, class B, class C>

int checkBetweeness(A& a, B& b, C& c){

if(((a.x\_cor == c.x\_cor)&&(a.y\_cor == c.y\_cor)) || ((b.x\_cor == c.x\_cor)&&(b.y\_cor ==

c.y\_cor))){

return 1;

}

else if((((c.x\_cor>a.x\_cor)||(c.y\_cor>a.y\_cor))&&((c.x\_cor<b.x\_cor)||(c.y\_cor<b.y\_cor)))){

return 1;

}

else

return 0;

}

template<class A, class B>

void checkIntersection(A& a, B& b){

Point p1,p2,p3,p4;

double p123, p124, p341, p342;

cout<<"\n\t\t FIRST LINE";

cout<<"\n\t\tStart Point:"<<endl;

p1.enterPointCoordinate();

cout<<"\n\t\tEnd Point:"<<endl;

p2.enterPointCoordinate();

cout<<"\n\t\t SECOND LINE";

cout<<"\n\t\tStart Point:"<<endl;

p3.enterPointCoordinate();

cout<<"\n\t\tEnd Point:"<<endl;

p4.enterPointCoordinate();

p123 = computeArea(p1,p2,p3);

p124 = computeArea(p1,p2,p4);

p341 = computeArea(p3,p4,p1);

p342 = computeArea(p3,p4,p2);

if (((p123 > 0 && p124 < 0) && (p341 > 0 && p342 < 0))|| ((p123 > 0 && p124 < 0) &&

(p341 < 0 && p342 > 0)) || ((p123 < 0 && p124 > 0) && (p341 < 0 && p342 > 0)) ||

((p123 < 0 && p124 > 0) && (p341 > 0 && p342 < 0))){

cout<<"\n\t\t-----> Pure Intersection <-----"<<endl;

}

}

else if ((p123 == 0 || p124 == 0 || p341 == 0 || p342 == 0)){

if(p123==0)

flag = checkBetweeness(p1,p2,p3);

else if(p124==0)

flag = checkBetweeness(p1,p2,p4);

else if(p341==0)

flag = checkBetweeness(p3,p4,p1);

else if(p342==0)

flag = checkBetweeness(p3,p4,p2);

cout<<"\n\t Flag = "<<flag;

if(flag==1)

cout<<"\n\t\t-----> Improper Intersection <-----"<<endl;

else

cout<<"\n\t\t-----> Lines do not intersects. <-----"<<endl;

}

else

cout<<"\n\t\t-----> Lines do not intersects. <-----"<<endl;

}

template<class A, class B, class C>

double computeArea(A& a, B& b, C& c){

return 0.5\*(a.x\_cor\*(b.y\_cor-c.y\_cor)+b.x\_cor\*(c.y\_cor-a.y\_cor)+c.x\_cor\*(a.y\_cor-b.y\_cor));

}

};

int main(){

int choice;

cout<<"\t\t\t 1. Turn Test."<<endl;

cout<<"\t\t\t 2. Check Intersection between two lines."<<endl;

cout<<"\n\t\t Enter the choice(1/2): ";

cin>>choice;

switch(choice){

case 1: Point point1, point2, point3;

TurnTest t1;

t1.enterPointForTurnTest(point1, point2, point3);

break;

case 2: LineIntersection li;

Line l1,l2;

li.checkIntersection(l1,l2);

break;

default: cout<<"Invalid choice.\n\tEnter the correct choice number(1/2): ";

}

return 0;

}