```cpp
#include<bits/stdc++.h>

using namespace std;

struct Point{
    double x,y;
    Point(double x=0,double y=0):x(x),y(y){}
};

typedef Point Vector;

struct Circle{
    Point c;
    double r;
    Circle(Point c,double r):c(c),r(r){}

    //通过圆心角确定圆上坐标
    Point point(double a){
        return Point(c.x + cos(a)*r,c.y+sin(a)*r);
    }
};

struct Line{
    Point p;
    Vector v;
    double ang;
    Line(){}
    Line(Point p,Vector v):p(p),v(v){}
    bool operator < (const Line &L) const {
        return ang < L.ang;
    }
};


const double eps = 1e-10;
const double PI = acos(-1);

Vector operator + (Vector A,Vector B){ return Vector(A.x+B.x,A.y+B.y); }
Vector operator - (Point A,Point B){ return Vector(A.x-B.x,A.y-B.y); }
Vector operator * (Vector A,double p){ return Vector(A.x*p,A.y*p); }
Vector operator / (Vector A,double p){ return Vector(A.x/p,A.y/p); }

bool operator < (const Point &a,const Point &b){
    return a.x < b.x || (a.x == b.x && a.y < b.y);
}

int dcmp(double x){
    if( fabs(x) < eps )  return 0;
    else return x < 0 ? -1 : 1;
}

bool operator == (const Point &a,const Point &b){
    return dcmp(a.x-b.x) == 0 && dcmp(a.y-b.y);
}

```

```cpp
double Dot(Vector A,Vector B){ return A.x*B.x + A.y*B.y; }
double Length(Vector A){ return sqrt(Dot(A,A)); }
double Angle(Vector A,Vector B){ return acos( Dot(A,B)/Length(A)/Length(B) ); }

double Cross(Vector A,Vector B){ return A.x*B.y - A.y*B.x; }
double Area2(Point A,Point B,Point C){ return Cross(B-A,C-A); }

Vector Rotate(Vector A,double rad){
    return Vector( A.x*cos(rad)-A.y*sin(rad),A.x*sin(rad)+A.y*cos(rad) );
}

bool OnSegment(Point p,Point a1,Point a2){
    return dcmp(Cross(a1-p,a2-p)) == 0 && dcmp(Dot(a1-p,a2-p)) < 0;
}

//直线与园的交点，返回交点个数，结果存在sol中
//没有清空sol

int getLineCircleIntersecion(Line L,Circle C,double &t1,double &t2,vector<Point> &sol){
    double a = L.v.x, b = L.p.x - C.c.x,
           c = L.v.y, d = L.p.y - C.c.y;
    double e = a*a + c*c, f = 2*(a*b + c*d),
           g = b*b + d*d - C.r*C.r;
    double delta = f*f - 4*e*g;
    if( dcmp(delta) < 0 )  return 0; //相离
    if( dcmp(delta) == 0 ){    //相切
        t1 = t2 = -f/(2*e);
        sol.push_back(C.point(t1));
        return 1;
    }
    t1 = (-f - sqrt(delta)) / (2*e);
    t2 = (-f + sqrt(delta)) / (2*e);
    sol.push_back(C.point(t1));
    sol.push_back(C.point(t2));
    return 2;
}

double angle(Vector v){ return atan2(v.y,v.x); }

//两圆相交

int getCircleCircleIntersection(Circle c1,Circle c2,vector<Point> &sol){
    double d = Length(c1.c - c2.c);
    if( dcmp(d) == 0 ){
        if( dcmp(c1.r - c2.r) == 0 )  return -1;  //两圆重合
        return 0;        //内含
    }
    if( dcmp(c1.r+c2.r-d) < 0 )  return 0;  //相离
    if( dcmp(fabs(c1.r - c2.r) - d) > 0 )  return 0;

    double a = angle(c2.c - c1.c);   // 向量c1 c2 的极角
    double da = acos( (c1.r*c1.r + d*d - c2.r*c2.r) / (2*c1.r*d) );

    Point p1 = c1.point(a-da),p2 = c1.point(a+da);

    sol.push_back(p1);
    if( p1 == p2 )  return 1;
    sol.push_back(p2);
    return 2;
}

//过点做圆切线

int getTangents(Point p,Circle C,Vector* v){
    Vector u = C.c - p;
    double dist = Length(u);
    if( dist < C.r )  return 0;
    else if( dcmp(dist - C.r) == 0 ){   //p在圆上
        v[0] = Rotate(u,PI/2);
        return 1;
    }
    else{
        double ang = asin(C.r / dist);
        v[0] = Rotate(u, -ang);
        v[1] = Rotate(u, +ang);
        return 2;
    }
}

//返回切线的数量 两圆的公切线
//a[i] 和 b[i] 表示 第 i 条切线在 圆A 和 圆B 上的 切点

int getTangents(Circle A,Circle B,Point* a,Point* b){
    int cnt = 0;
    if( A.r < B.r ){ swap(A,B);swap(a,b); }
    int d2 = (A.c.x - B.c.x)*(A.c.x - B.c.x) + (A.c.y - B.c.y)*(A.c.y - B.c.y);
    int rdiff = A.r - B.r;
    int rsum = A.r + B.r;
    if( d2 < rdiff*rdiff )  return 0; // 内含

    double base = atan2(B.c.y - A.c.y,B.c.x - A.c.x);
    if( d2 == 0 && A.r == B.r )  return -1;
    if( d2 == rdiff*rdiff ){                //内切
        a[cnt] = A.point(base);b[cnt] = B.point(base); cnt++;
        return 1;
    }

    double ang = acos( (A.r - B.r) / sqrt(d2) );
    a[cnt] = A.point(base + ang);b[cnt] = B.point(base + ang);  cnt++;
    a[cnt] = A.point(base - ang);b[cnt] = B.point(base - ang);  cnt++;

    if( d2 == rsum*rsum ){
        a[cnt] = A.point(base);b[cnt] = B.point(base + PI);  cnt++;
    }
    else if( d2 > rsum*rsum ){
        ang = acos( (A.r + B.r) / sqrt(d2) );
        a[cnt] = A.point(base + ang);b[cnt] = B.point(base + ang + PI); cnt++;
        a[cnt] = A.point(base - ang);b[cnt] = B.point(base - ang + PI); cnt++;
    }
    return cnt;
}
```

```

int main(int argc,char ** argv){

    return 0;
}
```