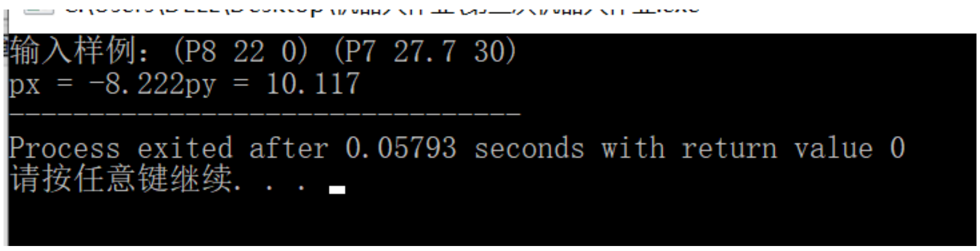
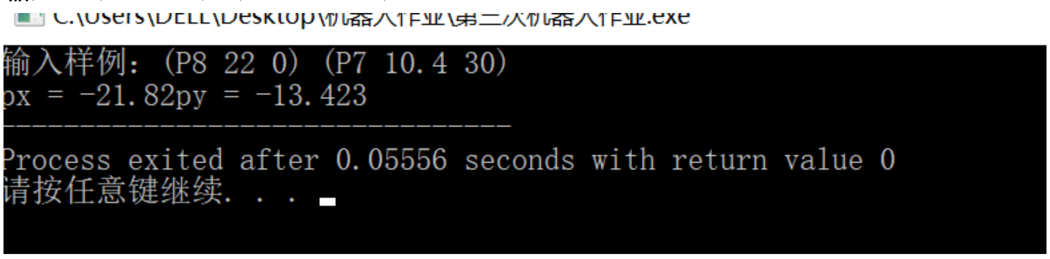
姓名：曾宇杰
学号：2019218164

输入：（P8 22 0）（P7 27.7 30）



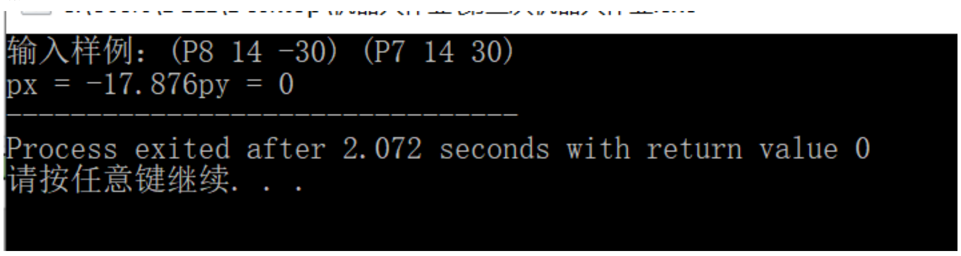
```
输入样例：（P8 22 0）（P7 27.7 30）
px = -8.222py = 10.117
--------------------------------
Process exited after 0.05793 seconds with return value 0
请按任意键继续. . .
```

输入：（P8 22 0）（P7 10.4 30）



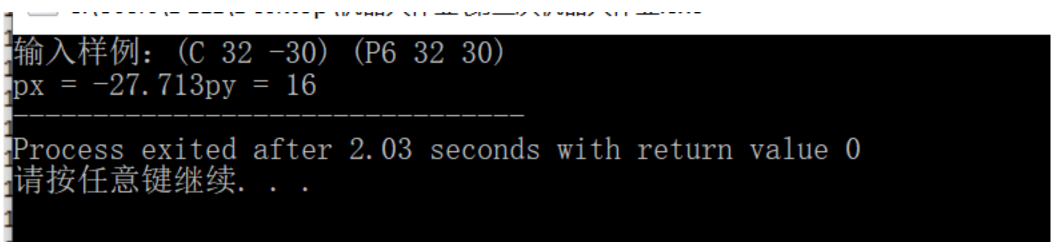
```
C:\Users\DELL\Desktop\机器人作业\第二次机器人作业.exe
输入样例：（P8 22 0）（P7 10.4 30）
px = -21.82py = -13.423
--------------------------------
Process exited after 0.05556 seconds with return value 0
请按任意键继续. . .
```

输入：（P8 14 -30）（P7 14 30）



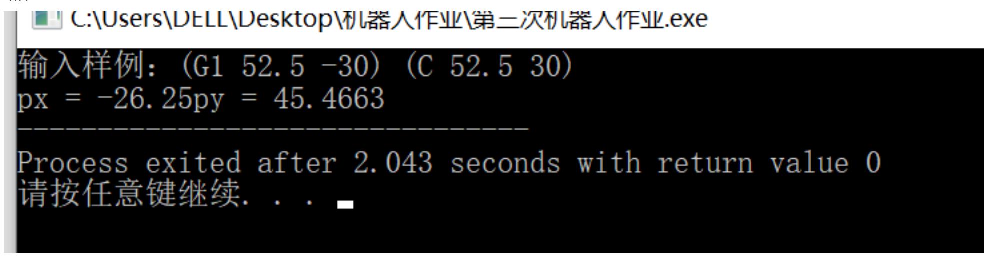
```
输入样例：（P8 14 -30）（P7 14 30）
px = -17.876py = 0
--------------------------------
Process exited after 2.072 seconds with return value 0
请按任意键继续. . .
```

输入：（C 32 -30）（P6 32 30）



```
输入样例：（C 32 -30）（P6 32 30）
px = -27.713py = 16
--------------------------------
Process exited after 2.03 seconds with return value 0
请按任意键继续. . .
```

输入：（G1 52.5 -30）（C 52.5 30）



```
C:\Users\DELL\Desktop\机器人作业\第二次机器人作业.exe
输入样例：（G1 52.5 -30）（C 52.5 30）
px = -26.25py = 45.4663
--------------------------------
Process exited after 2.043 seconds with return value 0
请按任意键继续. . .
```

代码：

```cpp
#include <iostream>
#include <math.h>
#include <string>
#include <vector>
#ifndef ROBOCUB_SECOND_HOMEWORK

using namespace std;
class point
{
private:
    double x;
    double y;
public:
    point(double xValue, double yValue);
    point();
    ~point();
    point & operator = (const point &p);
    inline void print(){
        cout << "(" << x << "," << y << ")";
    }
    inline void print_this(){
        cout << "px = " << this->x << "py = " << this->y;
    }

    friend class line;
    friend class circle;
    friend void printResult();
    friend double direction(const point& p,const point &p1,const point
&p2);
};
point::point()
{
};
point::point(double xValue, double yValue):x(xValue),y(yValue)
{
};
point::~point()
{
};

point & point::operator = (const point &p)
{
    x = p.x;
    y = p.y;
    return *this;
}

class line
{
private:
    double A;
    double B;
    double C;

public:
    line(double AValue, double BValue, double CValue);
    line();
    ~line();
    friend int intersection_line(const line & FirstLine, const line & S
econdLine, point &t);
    friend class circle;
    line & operator = (const line & l);
    inline void print()
    {
        cout << A << "x+" << B << "y+" << C << "=0";
    }
    /**
     * 求点到直线的距离
     */
    inline double disOfPointToLine(const point &p)
    {
        return fabs((A*p.x + B*p.y + C)/sqrt(A*A+B*B));
    }
};

/**
 * return : -1 代表无数个交点，0 代表没有交点，1 代表一个交点
 */
int intersection_line(const line & FirstLine, const line & SecondLine,
point &t)
{
    double m = FirstLine.A * SecondLine.B - FirstLine.B * SecondLine.A;
    if (m == 0)
    {
        if (FirstLine.C == SecondLine.C && FirstLine.A == SecondLine.A
&& FirstLine.B == SecondLine.B)
        {
            return -1;
        }
        else
        {
            return 0;
        }
    }
    else
    {
        t = point(((-FirstLine.C) * SecondLine.B - FirstLine.B * (-
SecondLine.C))/m, (FirstLine.A * (-SecondLine.C) - (-
FirstLine.C) * SecondLine.A)/m);
        return 1;
    }
}


line & line::operator = (const line & l)
{
        A = l.A;
        B = l.B;
        C = l.C;
}

line::line()
{}

line::line(double AValue, double BValue, double CValue)
{   if (A == 0&&B == 0)
    {
        cout << "This line don't exit!";
    }
    else
    {
        A = AValue;
        B = BValue;
        C = CValue;
    }
}

line::~line()
{}
```

```cpp
///////////////////////////////////////////////////////////////////////
/////////////
inline double myRound(double x)
{
    return floor(x*1000+0.5)/1000.0;
}

class circle
{
private:
    point centre;
    double R;
public:
    circle(point c, double r);
    ~circle();

    inline void print()
    {cout<<"圆心坐标：";centre.print();cout<<"半径："<<R;}

    int intersection(line & l,point &p1, point &p2);        //直线和圆交
点
    int intersectionWithCircle(const circle &c, point &p1, point &p2);
        //圆和圆交点
    friend void printResult();
};

int circle::intersection(line & l,point &p1, point &p2)
{
    double dis = l.disOfPointToLine(centre);
    //cout << "*" << dis;
    line m (l.B,-l.A,-l.B*centre.x + l.A*centre.y);
    //m.print();
    if (myRound(dis) > myRound(R))
    {
        return 0;
    }
    else if (myRound(dis) < myRound(R))
    {
        point s (0,0);
        point s1 (0,0);
        point s2 (0,0);
        intersection_line(l,m,s);
        //s.print();
        //l.print();
        if ( myRound(l.B) == 0)
        {
            s1.x = s.x;
            s2.x = s.x;
            s1.y = sqrt(R*R - dis*dis) + s.y;
            s2.y = -sqrt(R*R - dis*dis) + s.y;
        }
        else
        {
            double sinValue = sin(atan(-l.A/l.B));
            double cosValue = cos(atan(-l.A/l.B));
            s1.x = myRound(sqrt(R*R - dis*dis)*cosValue + s.x);
            s2.x = myRound(-sqrt(R*R - dis*dis)*cosValue + s.x);
            s1.y = myRound(sqrt(R*R - dis*dis)*sinValue + s.y);
            s2.y = myRound(-sqrt(R*R - dis*dis)*sinValue + s.y);
            /*
            s2.x = -sqrt(R*R - dis*dis)*cosValue + s.x;
            s1.y = sqrt(R*R - dis*dis)*sinValue + s.y;
            s2.y = -sqrt(R*R - dis*dis)*sinValue + s.y;
            */
        }

        p1 = s1;
        p2 = s2;
        return 2;
    }
    else
    {
        intersection_line(l,m,p1);
        return 1;
    }
}

//圆和圆相交
int circle::intersectionWithCircle(const circle &c, point &p1, point &p
2)
{
    double a2 = c.centre.x;
    double b2= c.centre.y;
    double r2 = c.R;
    double a1 = this->centre.x;
    double b1 = this->centre.y;
    double r1 = this->R;
```

```cpp
    double A = 2*a2 - 2*a1;
    double B = 2*b2 - 2*b1;
    double C = a1*a1-a2*a2+b1*b1-b2*b2-r1*r1+r2*r2;
    line l(A, B, C);

    int flag = this->intersection(l,p1,p2);
    //p1.print();
    //p2.print();

    return flag;
}

circle::circle(point c, double r)
{
    centre = c;
    R = r;
}

circle::~circle()
{
}

void getRAng(double &r, double &ang, string s,int &index)
{
    int k = index;
    int flag = 0;
    int begin = s.find(" ", index);
    int end = s.find(")", begin);
    r = stod(s.substr(index,begin - index));
    ang = stod(s.substr(begin+1, end - begin -1));
    index = end;
}


vector<circle> cs;            //圆
vector<double> angs;          //极角

void putInto(point &p, string s, int &i)
{
    double r = 0;
    double ang = 0;
    i += 3;
    getRAng(r, ang , s, i);
    i--;
    circle c(p,r);
    cs.push_back(c);
    angs.push_back(ang);
}
//p_p1 向量可以顺时针旋转到 P_p2 则返回值是正否则是负，若返回 0，代表 p1,p2,p 共
线
double direction(const point& p,const point &p1,const point &p2)
{
    point v1,v2;
    v1.x =p2.x -p.x ;
    v1.y=p2.y-p.y;
    v2.x =p1.x -p.x;
    v2.y=p1.y-p.y;
    return v1.x*v2.y-v1.y*v2.x;
}

void printResult()
{
    point t1(0,0);
    point t2(0,0);
    cs[0].intersectionWithCircle(cs[1],t1,t2);

    if (angs[0] < angs[1])
    {
        if (direction(t1,cs[0].centre,cs[1].centre) < 0)
        {
            t1.print_this();
        }
        else
        {
            t2.print_this();
        }
    }
    else if (angs[0] > angs[1] )
    {
        if (direction(t1,cs[0].centre,cs[1].centre) > 0)
        {
            t1.print_this();
        }
        else
        {
            t2.print_this();
        }
    }
```

```cpp
    else
    {
        cout << "两直线没有交点";
    }

}


#endif // !ROBOCUB_SECOND_HOMEWORK
int main()
{
    string s = "(G1 52.5 -30) (C 52.5 30)";
    cout << "输入样例: " << s << endl;
    //cin >> s;
    int flag = 0;

    for (int i = 0;i < s.size(); ++i)
    {

        if (s[i] == 'P')
        {

            if (s[i+1] == '1')
            {
                point p(-52.5, -32);
                putInto(p, s, i);
            }
            else if (s[i+1] == '2')
            {
                point p(-52.5, 32);
                putInto(p ,s, i);
            }
            else if (s[i+1] == '3')
            {
                point p(52.5, 32);
                putInto(p ,s, i);
            }
            else if (s[i+1] == '4')
            {
                point p(52.5, -32);
                putInto(p ,s, i);
            }
            else if (s[i+1] == '5')
            {
                point p(0, -32);
                putInto(p ,s, i);
            }
            else if (s[i+1] == '6')
            {
                point p(0, 32);
                putInto(p ,s, i);
            }
            else if (s[i+1] == '7')
            {
                point p(-30, -7);
                putInto(p ,s, i);
            }
            else if (s[i+1] == '8')
            {
                point p(-30, 7);
                putInto(p ,s, i);
            }
            else if (s[i+1] == '9')
            {
                point p(30, 7);
                putInto(p ,s, i);
            }
            else
            {
                point p(30, -7);
                putInto(p ,s, i);
            }

        }
        else if (s[i] == 'C')
        {
            point p(0, 0);
            i--;
            putInto(p, s, i);
        }
        else if (s[i] == 'G')
        {
            if (s[i+1] == '1')
            {
                point p(-52.5, 0);
                putInto(p, s, i);
            }
            else if (s[i+1] == '2')
            {
                point p(52.5, 0);
                putInto(p ,s, i);
            }
        }
    }

    printResult();

}
```