﻿// CodingChallenge.cpp : Defines the entry point for the console application.

//

#include "stdafx.h"

#include <iostream>

using namespace std;

typedef struct

{

int m\_integer1;

float m\_float;

char\* m\_string;

int m\_integer2;

char m\_char2;

bool m\_bool;

} Packing;

float doRoundFoatAfterComma(float \_number, int \_n\_AfterComma)

{

long \_number\_adjust\_rounded = long(\_number \* pow(10, \_n\_AfterComma));

return \_number\_adjust\_rounded / (pow(10, \_n\_AfterComma));

}

int main()

{

int X0, Y0, A, B, R;

cout << "Input the coordinate of center of circle : (X0, Y0), radius of circle (R > 0), and paramater (A, B) of equation of straight-line";

cout << "\nInput following the order : X0 Y0 R A B : ";

cin >> X0 >> Y0 >> R >> A >> B;

if (R <= 0)

{

//Wrong input for R since R has to > 0

return -1;

}

// The equation of circle : (x - X0)^2 + (y - Y0)^2 = R^2 (1)

// Replace y = A\*x + B to (1) to have quadratic equation

// (x - X0)^2 + (A\*x + B - Y0)^2 = R^2 ---> is same to

// x^2\*(1 + A^2) + x\*(2\*A\*Y0 - 2\*X0 - 2\*A\*B) + (X0^2 + B^2 + Y0^2 - 2\*B\*Y0 - R^2) = 0

// Let's replace m\_coefficient\_x\_2 = 1 + A^2, m\_coefficient\_x\_1 = 2\*A\*Y0 - 2\*X0 - 2\*A\*B, m\_coefficient = X0^2 + B^2 + Y0^2 - 2\*B\*Y0 - R^2

long m\_coefficient\_x\_2 = 1 + A\*A;

int m\_coefficient\_x\_1 = 2\*A\*Y0 - 2\*X0 - 2\*A\*B;

long m\_coefficient = X0\*X0 + B\*B + Y0\*Y0 - 2\*B\*Y0 - R\*R;

//Check how to solve quadratic equation from: https://en.wikipedia.org/wiki/Quadratic\_equation

//Check m\_coefficient\_x\_1\*m\_coefficient\_x\_1 - 4\*m\_coefficient\_x\_2\*m\_coefficient

long m\_delta\_coefficient = m\_coefficient\_x\_1\*m\_coefficient\_x\_1 - 4 \* m\_coefficient\_x\_2\*m\_coefficient;

if (m\_delta\_coefficient < 0)

{

cout << "\nNO";

}

else if (m\_delta\_coefficient == 0)

{

float m\_x\_intersect = doRoundFoatAfterComma((-m\_coefficient\_x\_1) / (2 \* m\_coefficient\_x\_2),3);

float m\_y\_intersect = doRoundFoatAfterComma(A\*m\_x\_intersect + B,3);

cout << "\nYES";

cout << "\n(" << m\_x\_intersect << "," << m\_y\_intersect << ")";

}

else

{

float m\_delta\_coefficient\_SQRT = sqrt(m\_delta\_coefficient);

float m\_x1\_intersect = doRoundFoatAfterComma((-m\_coefficient\_x\_1 + m\_delta\_coefficient\_SQRT) / (2 \* m\_coefficient\_x\_2),3);

float m\_y1\_intersect = doRoundFoatAfterComma(A\*m\_x1\_intersect + B,3);

float m\_x2\_intersect = doRoundFoatAfterComma((-m\_coefficient\_x\_1 - m\_delta\_coefficient\_SQRT) / (2 \* m\_coefficient\_x\_2),3);

float m\_y2\_intersect = doRoundFoatAfterComma(A\*m\_x2\_intersect + B,3);

cout << "\nYES";

if (m\_x1\_intersect <= m\_x2\_intersect)

{

cout << "\n(" << m\_x1\_intersect << "," << m\_y1\_intersect << ")";

cout << "(" << m\_x2\_intersect << "," << m\_y2\_intersect << ")";

}

else

{

cout << "\n(" << m\_x2\_intersect << "," << m\_y2\_intersect << ")";

cout << "(" << m\_x1\_intersect << "," << m\_y1\_intersect << ")";

}

}

cout << "\n";

system("pause");

return 0;

}