```cpp
//Created on December 1st, 2020
//author: Edmond Ngantung

#include<iostream>
using namespace std;

class Rectangle
{
    private:
        int x, y;

    public:
        //Default Constructor
        Rectangle(int x1, int y1) { x = x1; y = y1; }

        // Copy constructor
        Rectangle(const Rectangle &copyConst) { x = copyConst.x; y = copyConst.y; }
        int getX()            {   return x; }
        int getY()            {   return y; }

        //This function is to check whether the given point is inside of the rectangle or not
        bool isInside_Rectangle(int x1, int y1, int x2, int y2, int x, int y)
        {
            if (x > x1 and x < x2 and y > y1 and y < y2)
                return true;
            return false;
        }

        //This function is to return the new point X2 as horizontal extension
        int new_pX2(int X1, int Y1, int X2, int Y2, int num_to_extend_for_horizontal)
        {
            int width;

            width = X2 - X1;
            X2 = X2 + width*num_to_extend_for_horizontal;

            return X2;
        }

        //This function is to return the new point Y2 as vertical extension
        int new_pY2(int X1, int Y1, int X2, int Y2, int num_to_extend_for_vertical)
        {
            int height;

            height = Y2 - Y1;
            Y2 = Y2 + height*num_to_extend_for_vertical;

            return Y2;
        }

};

class Circle
{
    private:
        int xc, yc, r;

    public:
        //Default Constructor
        Circle(int x1, int y1, int r1) { xc = x1; yc = y1; r = r1;}

        // Copy constructor
        Circle(const Circle &copyConst) { xc = copyConst.xc; yc = copyConst.yc; r = copyConst.r;}
        int getXc()            {   return xc; }
        int getYc()            {   return yc; }
        int getRad()           {   return r;  }

        //This function is to check whether the given point is inside of the circle or not
        bool isInside_Circle(int circle_x, int circle_y,
                   int rad, int x, int y)
        {
            // Compare radius of circle with distance
            // of its center from given point
            if ((x - circle_x) * (x - circle_x) +
                (y - circle_y) * (y - circle_y) <= rad * rad)
                return true;
            else
                return false;
        }

};

int main()
{
    //UNIT TESTS
    //Two Rectangle points as components taken from its diagonal (bottom left and top right)
    Rectangle pointP1(0, 0); // Normal constructor is called here
    Rectangle pointP2(10,8); // Normal constructor is called here
    Rectangle pointAnySinglePoint(1,5);

    //Circle point taken from its centroid and radius
    Circle pointCenterAndRadius(3,4,5);

    //Number of fold for the extension in horizontal and vertical using the rectangular shape
    int num_of_Hrect = 1;
    int num_of_Vrect = 1;

    Rectangle cp_pointP1 = pointP1; // Copy constructor is called here
    Rectangle cp_pointP2 = pointP2; // Copy constructor is called here
    Circle cp_pointCenterAndRadius = pointCenterAndRadius; // Copy constructor is called here

    Rectangle cp_pointAnySinglePoint = pointAnySinglePoint; // Copy constructor is called here

    bool answer_Inside_Rectangle;
    bool answer_Inside_Circle;

    int pX1 = cp_pointP1.getX();
    int pY1 = cp_pointP1.getY();
    int pX2 = cp_pointP2.getX();
    int pY2 = cp_pointP2.getY();

    int pXc = cp_pointCenterAndRadius.getXc();
    int pYc = cp_pointCenterAndRadius.getYc();
    int pRad = cp_pointCenterAndRadius.getRad();

    int pX = cp_pointAnySinglePoint.getX();
    int pY = cp_pointAnySinglePoint.getY();

    cout<<"HINTS : 1. Two Rectangle points as components in the class taken from its diagonal (bottom left and top right)"<<endl;
    cout<<"        2. Circle point taken from its centroid and radius"<<endl;

    cout <<"\nGiven Point to check : ("<<pX<<","<<pY<<") for checking whether is located inside/outside of the shapes (Rectangle and Circle) "<<endl;

    cout <<"\nMain Shape : Rectangular (CASE 1: Original or Without Extension)";
    cout <<"\nPoint A : ("<<pX1<<","<<pY1<<")";
    cout <<"\nPoint B : ("<<pX2<<","<<pY1<<")";
    cout <<"\nPoint C : ("<<pX2<<","<<pY2<<")";
    cout <<"\nPoint D : ("<<pX1<<","<<pY2<<")"<<endl;

```

```cpp
    cout <<"\nAdditional Shape : Circle";
    cout <<"\nCentroid : ("<<pXc<<","<<pYc<<")";
    cout <<"\nRadius : "<<pRad << endl;

    answer_Inside_Rectangle = pointAnySinglePoint.isInside_Rectangle(pX1, pY1, pX2, pY2, pX, pY);
    answer_Inside_Circle = pointCenterAndRadius.isInside_Circle(pXc, pYc, pRad, pX, pY);

    cout <<"\nCheck whether the given Point is inside or not in the Rectangle (Original): ";
    if(answer_Inside_Rectangle == 1) cout    <<"YES, it is inside of the Rectangle";
    else            cout    <<"NO, it is outside of the Rectangle";

    cout <<"\nCheck whether the given Point is inside or not in the Circle: ";
    if(answer_Inside_Circle == 1) cout    <<"YES, it is inside of the Circle";
    else            cout    <<"NO, it is outside of the Circle";

    pX2 = pointAnySinglePoint.new_pX2(pX1, pY1, pX2, pY2, num_of_Hrect);
    pY2 = pointAnySinglePoint.new_pY2(pX1, pY1, pX2, pY2, num_of_Vrect);

    cout<<endl;
    cout <<"\nMain Shape : Rectangular (CASE 2: With Extension) which is "<< num_of_Hrect << " fold horizontaly and "<< num_of_Vrect <<" fold vertically";
    cout <<"\nPoint A : ("<<pX1<<","<<pY1<<")";
    cout <<"\nPoint B : ("<<pX2<<","<<pY1<<")";
    cout <<"\nPoint C : ("<<pX2<<","<<pY2<<")";
    cout <<"\nPoint D : ("<<pX1<<","<<pY2<<")"<<endl;

    answer_Inside_Rectangle = pointAnySinglePoint.isInside_Rectangle(pX1, pY1, pX2, pY2, pX, pY);
    //answer_Circle = pointCenterAndRadius.isInside_Circle(pXc, pYc, pRad, pX, pY);

    cout <<"\nCheck whether the given Point is inside or not in the Rectangle (with Extension): ";
    if(answer_Inside_Rectangle == 1) cout    <<"YES, it is inside of the Rectangle";
    else            cout    <<"NO, it is outside of the Rectangle";
    cout <<endl;
    return 0;
}
```