## //C++ code to check if the given point lies inside or //outside a polygon

#include <iostream>

using namespace std;

// Define Infinite

#define INF 10000

struct Point

{

    int x;

    int y;

};

// Given three colinear points a, b, c, the function checks if

// point b lies on line segment 'ac'

bool collinear(Point a, Point b, Point c)

{

    if (b.x <= max(a.x, c.x) && b.x >= min(a.x, c.x) &&

            b.y <= max(a.y, c.y) && b.y >= min(a.y, c.y))

        return true;

    return false;

}

// To find orientation of ordered triplet (a, b, c)

int orientation(Point a, Point b, Point c)

{

    int val = (b.y - a.y) \* (c.x - b.x) -

              (b.x - a.x) \* (c.y - b.y);

    if (val == 0) return 0;  // colinear

    return (val > 0)? 1: 2; // clock or counterclock wise

}

// The function that returns true if line segment intersect.

bool Intersect(Point a1, Point b1, Point a2, Point b2)

{

    int orn1 = orientation(a1, b1, a2);

    int orn2 = orientation(a1, b1, b2);

    int orn3 = orientation(a2, b2, a1);

    int orn4 = orientation(a2, b2, b1);

    if (o1 != o2 && o3 != o4)

        return true;

    if (o1 == 0 && collinear(a1, a2, b1))

return true;

    if (o2 == 0 && collinear(a1, b2, b1))

return true;

    if (o3 == 0 && collinear(a2, a1, b2))

return true;

    if (o4 == 0 && collinear(a2, b1, b2))

return true;

    return false;

}

// Returns true if the point p lies inside the polygon[] with n vertices

bool CheckForInside(Point poly[], int n, Point p)

{

    if (n < 3)  return false;

    Point extpos = {INF, p.y};

    int count = 0, i = 0;

do

    {

        int next = (i+1)%n;

        if (Intersect(poly[i], poly[next], p, extpos))

        {

            if (orientation(poly[i], p, poly[next]) == 0)

               return collinear(poly[i], p, poly[next]);

            count++;

        }

        i = next;

    } while (i != 0);

    if(count%2 == 1)

return true;

}

int main()

{

    Point poly1[] = {{1, 0}, {8, 3}, {8, 8}, {1, 5}};

    int len = sizeof(poly1)/sizeof(poly1[0]);

    Point p = {3, 5};

    CheckForInside(poly1, len, p)? cout << "True \n": cout << "False \n";

}