

#include <iostream>

#include <set>

using namespace std;

class Point {

public:

int x, y;

Point(int \_x, int \_y) : x(\_x), y(\_y) {}

};

class Node {

public:

Point point;

Node\* next;

Node(Point p) : point(p), next(NULL) {}

};

class CircularLinkedList {

private:

Node\* head;

public:

CircularLinkedList() : head(NULL) {}

void insert(Point p) {

Node\* newNode = new Node(p);

if (!head) {

head = newNode; //To check for empty list

head->next = head;

return;

}

Node\* current = head;

while (current->next != head) {

if (current->point.x == p.x && current->point.y == p.y) {

delete newNode;

return; // Ignore duplicate point

}

current = current->next;

}

current->next = newNode;

newNode->next = head;

}

int maxPointsOnLine() {

if (!head) { //For empty list

return 0;

}

int maxPoints = 1;

Node\* current = head;

while (current->next != head) {

int duplicate = 0;

Node\* temp = current->next;

while (temp != current) {

if ((current->point.x == temp->point.x) && (current->point.y == temp->point.y)) {

duplicate++;

} else {

int count = 2;

Node\* check = temp->next;

while (check != current) {

if ((current->point.x \* (temp->point.y - check->point.y) +

temp->point.x \* (check->point.y - current->point.y) +

check->point.x \* (current->point.y - temp->point.y)) == 0) {

count++;

}

check = check->next;

}

maxPoints = max(maxPoints, count);

}

temp = temp->next;

}

maxPoints = max(maxPoints, duplicate + 1);

current = current->next;

}

return maxPoints;

}

};

int main() {

int n;

cout << "Enter the number of points: ";

cin >> n;

CircularLinkedList cll;

cout << "Enter points in the format [x y]: " << endl;

for (int i = 0; i < n; i++) {

int x, y;

cin >> x >> y;

cll.insert(Point(x, y));

}

int result = cll.maxPointsOnLine();

cout << "Maximum points on the same line: " << result << endl;

return 0;

}