```c
#include <stdio.h>
#include <math.h>

enum{PAR, INTER, IDEN};

struct Point{
    int x, y;
};

struct Line{
    struct Point p1, p2;
};

void readPoint(struct Point *p){
    scanf("%d %d", &p->x, &p->y);
}

void printPoint(struct Point p){
    printf("%d %d", p.x, p.y);
}

void readLine(struct Line *l){
    readPoint(&l->p1);
    readPoint(&l->p2);
}

void printLine(struct Line l){
    printPoint(l.p1);
    printf(" ");
    printPoint(l.p2);
    printf("\n");
}

// Do these two lines have the same slope?
int isEqualSlope(struct Line l1, struct Line l2){
    int dy1 = l1.p2.y - l1.p1.y;
    int dx1 = l1.p2.x - l1.p1.x;
    int dy2 = l2.p2.y - l2.p1.y;
    int dx2 = l2.p2.x - l2.p1.x;

    if((dy2 * dx1) == (dy1 * dx2))
        return 1;
    return 0;
}

// Do these two lines give us an equal y intercept?
int isEqualYIntercept(struct Line l1, struct Line l2){
    int dy1 = l1.p2.y - l1.p1.y;
    int dx1 = l1.p2.x - l1.p1.x;
    int dy2 = l2.p2.y - l2.p1.y;
    int dx2 = l2.p2.x - l2.p1.x;

    // One is vertical another is not - cannot have same y intercept
    if(dx1 == 0 && dx2 != 0)
        return 0;
    if(dx1 != 0 && dx2 == 0)
        return 0;

    // Hmm ... both vertical - be careful in defining y intercept
    if(dx1 == 0 && dx2 == 0){
        if(l1.p2.x == l2.p2.x)
            return 1;
        return 0;
    }

    // Neither line is vertical - general case
    double m = (double)dy2/(double)dx2;
    double int1 = l1.p2.y * m - l1.p2.x;
    double int2 = l2.p2.y * m - l2.p2.x;
    if(fabs(int1 - int2) < 0.0001)
        return 1;
    return 0;
}

int processPair(struct Line l1, struct Line l2){
    if(isEqualSlope(l1, l2)){
        if(isEqualYIntercept(l1, l2)){
            return IDEN;
        }
        return PAR;
    }
    return INTER;
}

int main(){
    int n, i, j, counts[3] = {0,0,0};
    scanf("%d", &n);
    struct Line lines[n];
    for(i = 0; i < n; i++)
        readLine(&lines[i]);

    for(i = 0; i < n; i++)
        for(j = i + 1; j < n; j++)
            counts[processPair(lines[i], lines[j])]++;

    printf("PARALLEL: %d\n", counts[0]);
    printf("INTERSECT: %d\n", counts[1]);
    printf("IDENTICAL: %d", counts[2]);
    return 0;
}
```