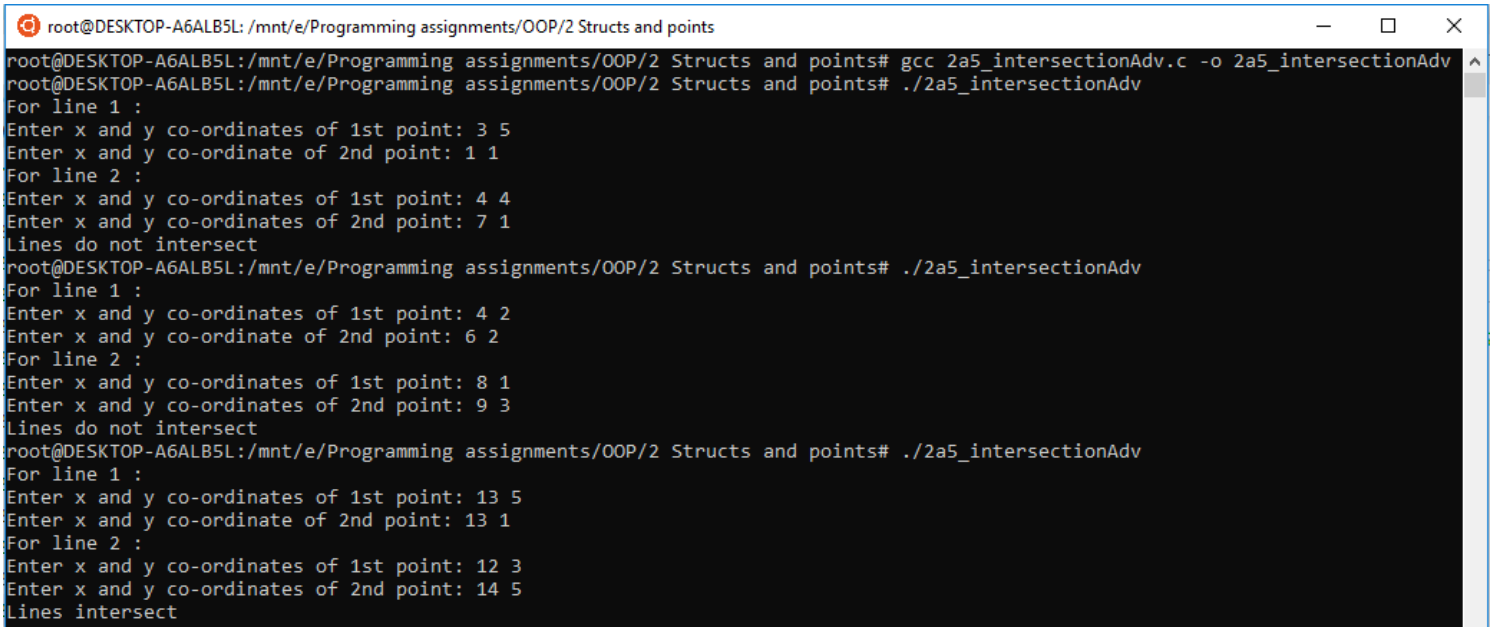
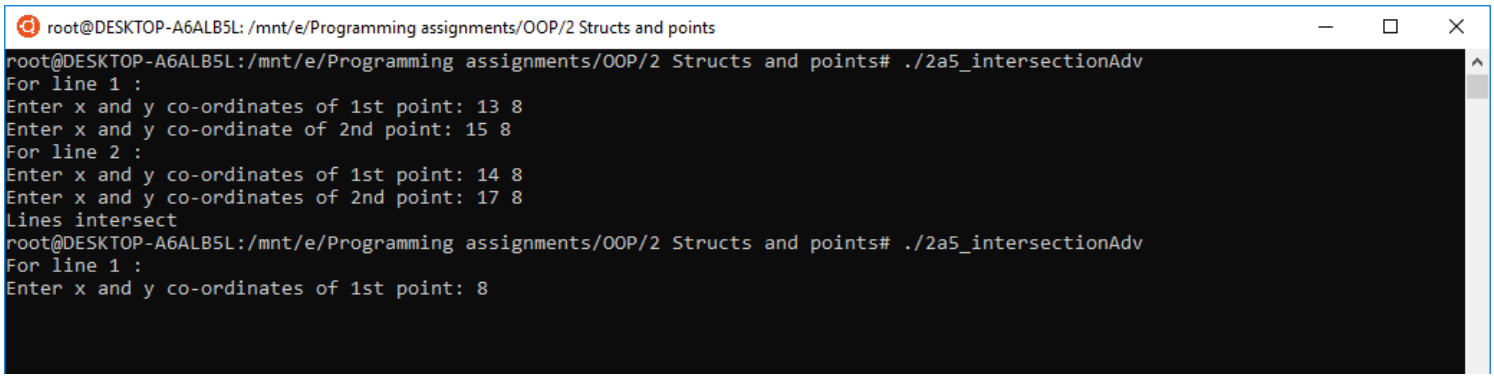
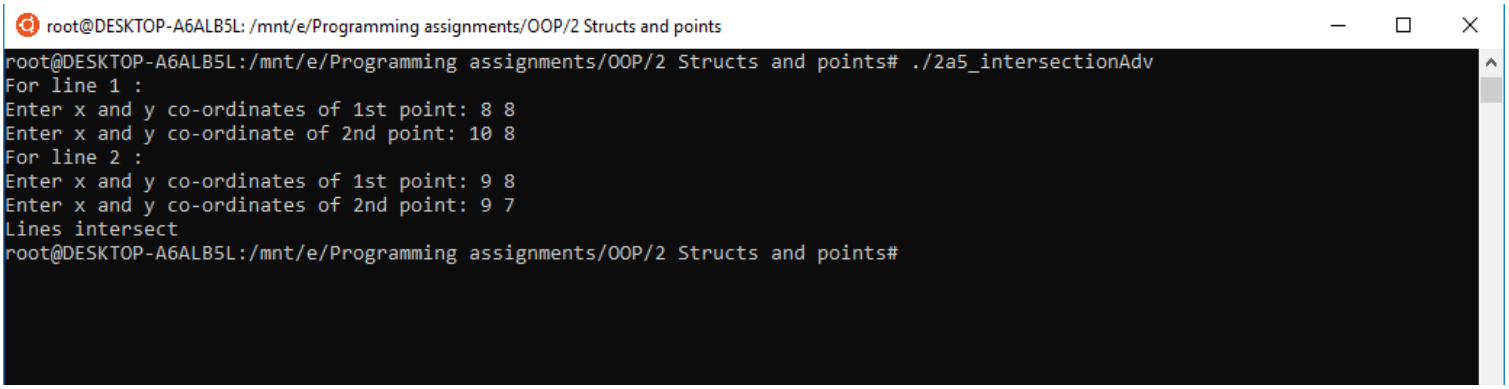
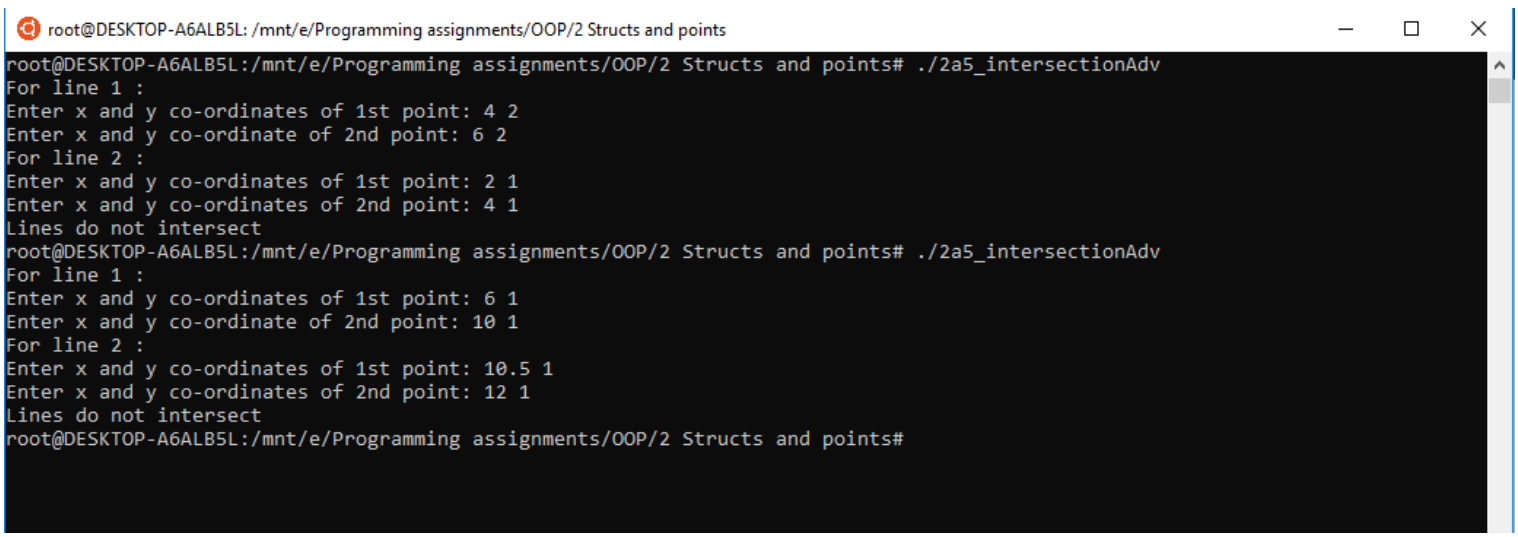
Output :


root@DESKTOP-A6ALB5L: /mnt/e/Programming assignments/OOP/2 Structs and points
root@DESKTOP-A6ALB5L:/mnt/e/Programming assignments/OOP/2 Structs and points# gcc 2a5_intersectionAdv.c -o 2a5_intersectionAdv
root@DESKTOP-A6ALB5L:/mnt/e/Programming assignments/OOP/2 Structs and points# ./2a5_intersectionAdv
For line 1 :
Enter x and y co-ordinates of 1st point: 3 5
Enter x and y co-ordinate of 2nd point: 1 1
For line 2 :
Enter x and y co-ordinates of 1st point: 4 4
Enter x and y co-ordinates of 2nd point: 7 1
Lines do not intersect
root@DESKTOP-A6ALB5L:/mnt/e/Programming assignments/OOP/2 Structs and points# ./2a5_intersectionAdv
For line 1 :
Enter x and y co-ordinates of 1st point: 4 2
Enter x and y co-ordinate of 2nd point: 6 2
For line 2 :
Enter x and y co-ordinates of 1st point: 8 1
Enter x and y co-ordinates of 2nd point: 9 3
Lines do not intersect
root@DESKTOP-A6ALB5L:/mnt/e/Programming assignments/OOP/2 Structs and points# ./2a5_intersectionAdv
For line 1 :
Enter x and y co-ordinates of 1st point: 13 5
Enter x and y co-ordinate of 2nd point: 13 1
For line 2 :
Enter x and y co-ordinates of 1st point: 12 3
Enter x and y co-ordinates of 2nd point: 14 5
Lines intersect


root@DESKTOP-A6ALB5L: /mnt/e/Programming assignments/OOP/2 Structs and points
root@DESKTOP-A6ALB5L:/mnt/e/Programming assignments/OOP/2 Structs and points# ./2a5_intersectionAdv
For line 1 :
Enter x and y co-ordinates of 1st point: 13 8
Enter x and y co-ordinate of 2nd point: 15 8
For line 2 :
Enter x and y co-ordinates of 1st point: 14 8
Enter x and y co-ordinates of 2nd point: 17 8
Lines intersect
root@DESKTOP-A6ALB5L:/mnt/e/Programming assignments/OOP/2 Structs and points# ./2a5_intersectionAdv
For line 1 :
Enter x and y co-ordinates of 1st point: 8


root@DESKTOP-A6ALB5L: /mnt/e/Programming assignments/OOP/2 Structs and points
root@DESKTOP-A6ALB5L:/mnt/e/Programming assignments/OOP/2 Structs and points# ./2a5_intersectionAdv
For line 1 :
Enter x and y co-ordinates of 1st point: 8 8
Enter x and y co-ordinate of 2nd point: 10 8
For line 2 :
Enter x and y co-ordinates of 1st point: 9 8
Enter x and y co-ordinates of 2nd point: 9 7
Lines intersect
root@DESKTOP-A6ALB5L:/mnt/e/Programming assignments/OOP/2 Structs and points#


root@DESKTOP-A6ALB5L: /mnt/e/Programming assignments/OOP/2 Structs and points
root@DESKTOP-A6ALB5L:/mnt/e/Programming assignments/OOP/2 Structs and points# ./2a5_intersectionAdv
For line 1 :
Enter x and y co-ordinates of 1st point: 4 2
Enter x and y co-ordinate of 2nd point: 6 2
For line 2 :
Enter x and y co-ordinates of 1st point: 2 1
Enter x and y co-ordinates of 2nd point: 4 1
Lines do not intersect
root@DESKTOP-A6ALB5L:/mnt/e/Programming assignments/OOP/2 Structs and points# ./2a5_intersectionAdv
For line 1 :
Enter x and y co-ordinates of 1st point: 6 1
Enter x and y co-ordinate of 2nd point: 10 1
For line 2 :
Enter x and y co-ordinates of 1st point: 10.5 1
Enter x and y co-ordinates of 2nd point: 12 1
Lines do not intersect
root@DESKTOP-A6ALB5L:/mnt/e/Programming assignments/OOP/2 Structs and points#

Code :

```c
// Write a function which will take two arguments of type struct LineSegment and
// will return 1 if the two line segments are intersecting and return 0 otherwise
#include<stdio.h>
struct Point
{
    double x;
    double y;
};

struct LineSeg
{
    struct Point p1;
    struct Point p2;
};

double maximum(double a, double b)
{
    if(a>=b)
        return a;
    return b;
}

double minimum(double a, double b)
{
    if(a<=b)
        return a;
    return b;
}

int isOnSeg(struct Point a, struct Point check, struct Point b){
    if( check.x <= maximum(a.x, b.x) &&  check.x >= minimum(a.x, b.x)
     && check.y <= maximum(a.y, b.y) && check.y >= minimum(a.y, b.y) )
        return 1;
}

int findOrientation(struct Point a, struct Point b, struct Point check){
    double result = (b.y - a.y)*(check.x - b.x) - (b.x - a.x)*(check.y - b.y);
    if(result>0)
        return 1;
    if(result<0)
        return -1;
    return 0;
}
```

```c
int doesIntersect(struct LineSeg l1, struct LineSeg l2)
{
    int o1 = findOrientation(l2.p1, l2.p2 , l1.p1) ;
    int o2 = findOrientation(l2.p1, l2.p2 , l1.p2) ;
    int o3 = findOrientation(l1.p1, l1.p2 , l2.p1) ;
    int o4 = findOrientation(l1.p1, l1.p2 , l2.p2) ;

    if(o1!=o2 && o3!=o4)
        return 1;
    if(o1==0 && isOnSeg(l2.p1, l1.p1, l2.p2))
        return 1;
    if(o2==0 && isOnSeg(l2.p1, l1.p2, l2.p2))
        return 1;
    if(o3==0 && isOnSeg(l1.p1, l2.p1, l1.p2))
        return 1;
    if(o4==0 && isOnSeg(l1.p1, l2.p2, l1.p2))
        return 1;

    return 0;
}

int main()
{
        struct LineSeg l1;
        struct LineSeg l2;

        printf("For line 1 :\n");
        printf("Enter x and y co-ordinates of 1st point: ");
        scanf("%lf%lf",&l1.p1.x,&l1.p1.y);
        printf("Enter x and y co-ordinate of 2nd point: ");
        scanf("%lf%lf",&l1.p2.x,&l1.p2.y);

        printf("For line 2 :\n");
        printf("Enter x and y co-ordinates of 1st point: ");
        scanf("%lf%lf",&l2.p1.x,&l2.p1.y);
        printf("Enter x and y co-ordinates of 2nd point: ");
        scanf("%lf%lf",&l2.p2.x,&l2.p2.y);

        if(doesIntersect(l1,l2))
            printf("Lines intersect \n");
        else
            printf("Lines do not intersect \n");

    return 0;
}
```