University of Massachusetts Dartmouth
Department of Electrical and Computer Engineering

ECE 161
Example program - Intersecting Lines Program v1.1

```
//*****************************************************************
//  Program:    Given two lines, determine if lines are intersecting,
//              parallel or co-linear, and print.  If lines are
//              intersecting, print intersect point.
//  Date:       September 2003
//  Programmer: P. H. Viall
//  Version:    1.0 - initial version
//              1.1 - added functions getpoint, getline, find_mx_b
//-----------------------------------------------------------------
#include <stdio.h>

//  Globally defined structures
typedef struct point
{
    double x;
    double y;
} POINT;

typedef struct line
{
    POINT p0;
    POINT p1;
} LINE;

//  Function Prototypes
POINT getpoint(int point01);
LINE getline(char ab);
void find_mx_b(LINE L, double *ptr_m, double *ptr_b);

//*****************************************************************
void main(void)
{
    LINE LineA;             //  First Line
    LINE LineB;             //  Second Line
    double slopeA, yintA;   //  Slope, yint of first line
    double slopeB, yintB;   //  Slope, yint of second line
    double xsol, ysol;      //  x,y solution (if lines intersect)

    LineA = getline('A');   //  Get points for each of lines
    LineB = getline('B');

    find_mx_b(LineA, &slopeA, &yintA);  //  Find slope and
    find_mx_b(LineB, &slopeB, &yintB);  //  yint for each line

//  Determine if lines are Parallel, co-linear, or intersecting
//  if slopes are equal then lines are either:
//      co-linear (if yint are same)
//      parallel (if yint are differentor)
//  if slopes are different, then lines are intersecting

    if (slopeA == slopeB)
    {
        if (yintA == yintB)
            printf("Lines are co-linear...all points are common\n");
        else
            printf("Lines are parrallell - no points are common\n");
    }
```

```
    else
    {
        xsol = (yintB - yintA) / (slopeA - slopeB);
        ysol = slopeA * xsol + yintA;
        printf("Lines intersect at %lf, %lf\n",xsol,ysol);
    }
}

//*****************************************************************
//  getpoint -  prompt user for x and y coordinate (POINT)
//              return ordered pair
//
//  on entry -  point01 - contains what number point to display
//
//  on exit -   returns POINT value with x, y coordinate
//
//  calls   -   none
//-----------------------------------------------------------------
POINT getpoint(int point01)
{
    POINT p;
    printf("    point %d - enter x and y: ", point01);
    scanf("%lf %lf",&p.x, &p.y);
    return p;
}
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

//*****************************************************************
//  getline -   prompt user for two points which describe a line
//              store in LINE structure
//
//  on entry -  ab - holds character for line id (A, B, C, ...)
//
//  on exit -   returns LINE (.p0.x,.p0.y)-(.p1.x,.p1.y)
//
//  calls   -   getpoint
//-----------------------------------------------------------------
LINE getline(char ab)
{
    LINE L;
    printf("Line %c:\n",ab);
    L.p0 = getpoint(0);
    L.p1 = getpoint(1);
    return L;
}
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

//*****************************************************************
//  find_mx_b - find slope(m) and yint(b) of LINE L
//
//  on entry -  L - LINE structure with two points of a line
//              *ptr_m - address of variable to store slope
//              *ptr_b - address of yint to store y intercept
//
//  on exit -   *ptr_m and *ptr_b updated
//
//  calls   -   none
//-----------------------------------------------------------------
void find_mx_b(LINE L, double *ptr_m, double *ptr_b)
{
    *ptr_m = (L.p1.y-L.p0.y)/(L.p1.x-L.p0.x);
    *ptr_b = L.p0.y - *m * L.p0.x;
}
//%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```