**MARKS AWARDED: 98**

/\*\*

\* CS1010 AY2011/2 Semester 2 Lab3 Ex3

\*

\* rectangles.c

\* Description: Read in values representing two rectangles and

\* determine whether the two rectangle

\* overlap, touch or are disjoint.

\* If they overlap, compute the are of overlap

\*

\* Name: Loh Wan Xin

\* Discussion Group: B02

\*/

#include <stdio.h>

#include <math.h>

#define OVERLAP -1

#define TOUCH 0

#define DISJOINT 1

void adjustCorners(float \*, float \*, float \*, float \*);

void typeAndArea(float, float, float, float,

float, float, float, float, int \*, float \*);

int find\_type(float, float, float, float,

float, float, float, float);

float find\_area(float, float, float, float,

float, float, float, float);

int main(void)

{

int type;

float rectAx1, rectAy1, rectAx2, rectAy2; // 1st rectangle

float rectBx1, rectBy1, rectBx2, rectBy2; // 2nd rectangle

float area; // overlapped area

// read inputs

printf("Enter one corner of 1st rectangle: ");

scanf("%f %f", &rectAx1, &rectAy1);

printf("Enter opp corner of 1st rectangle: ");

scanf("%f %f", &rectAx2, &rectAy2);

printf("Enter one corner of 2nd rectangle: ");

scanf("%f %f", &rectBx1, &rectBy1);

printf("Enter opp corner of 2nd rectangle: ");

scanf("%f %f", &rectBx2, &rectBy2);

// user may not enter bottom-left and top-right corners,

// adjust the values so that these 2 corners are captured

adjustCorners(&rectAx1, &rectAy1, &rectAx2, &rectAy2);

adjustCorners(&rectBx1, &rectBy1, &rectBx2, &rectBy2);

typeAndArea(rectAx1, rectAy1, rectAx2, rectAy2,

rectBx1, rectBy1, rectBx2, rectBy2,

&type, &area);

printf("Rectangle A = ([%.1f, %.1f], [%.1f, %.1f])\n",

rectAx1,rectAy1, rectAx2, rectAy2);

printf("Rectangle B = ([%.1f, %.1f], [%.1f, %.1f])\n",

rectBx1,rectBy1, rectBx2, rectBy2);

if (type==OVERLAP)

printf("Rectangles A and B overlap each other.\n");

else if (type == TOUCH)

printf("Rectangles A and B touch each other.\n");

else if (type == DISJOINT)

printf("Rectangles A and B are disjoint.\n");

printf("Overlapped area = %.2f\n", area);

return 0;

}

// after invocation,

// [\*rect\_x1, \*rect\_y1] will be bottom-left corner

// [\*rect\_x2, \*rect\_y2] will be top-right corner

void adjustCorners(float \*rect\_x1, float \*rect\_y1,

float \*rect\_x2, float \*rect\_y2)

{

float temp;

if (\*rect\_x1 > \*rect\_x2){

temp = \*rect\_x2;

\*rect\_x2 = \*rect\_x1;

\*rect\_x1 = temp;}

if (\*rect\_y1 > \*rect\_y2){

temp = \*rect\_y2;

\*rect\_y2 = \*rect\_y1;

\*rect\_y1 = temp;}

}

// return type and overlapped area to the caller

void typeAndArea(float rectAx1, float rectAy1, float rectAx2, float rectAy2,

float rectBx1, float rectBy1, float rectBx2, float rectBy2,

int \*type, float \*overlapping\_area)

{

\*type = find\_type(rectAx1, rectAy1, rectAx2, rectAy2,

rectBx1, rectBy1, rectBx2, rectBy2);

if (\*type == OVERLAP)

\*overlapping\_area = find\_area(rectAx1, rectAy1, rectAx2, rectAy2,

rectBx1, rectBy1, rectBx2, rectBy2);

else

\*overlapping\_area = 0.00;

}

// return position of two rectangles: overlap(-1), disjoint(1), touch(0)

int find\_type(float rectAx1, float rectAy1, float rectAx2, float rectAy2,

float rectBx1, float rectBy1, float rectBx2, float rectBy2)

{

// now check their relative position

if (rectAx1 > rectBx2 || rectAx2 < rectBx1

|| rectAy1 > rectBy2 || rectAy2 < rectBy1)

return DISJOINT;

else if ( rectAx1 == rectBx2 || rectBx1 == rectAx2 ||

rectAy1 == rectBy2 || rectBy1 == rectAy2 )

return TOUCH;

else

return OVERLAP;

}

// return the value of overlapped area to the caller

float find\_area(float rectAx1, float rectAy1, float rectAx2, float rectAy2,

float rectBx1, float rectBy1, float rectBx2, float rectBy2)

{

float area, length, height;

length = fabs(fmin(rectAx2,rectBx2) - fmax(rectAx1,rectBx1));

height = fabs(fmin(rectAy2,rectBy2) - fmax(rectAy1,rectBy1));

area = length\*height;

return area;

}