<html><head></head><body><pre style="word-wrap: break-word; white-space: pre-wrap;">/*
circle_circle_intersection() *
 * Determine the points where 2 circles in a common plane intersect.
 *
 * int circle_circle_intersection(
 *   // center and radius of 1st circle
 *   double x0, double y0, double r0,
 *   // center and radius of 2nd circle
 *   double x1, double y1, double r1,
 *   // 1st intersection point
 *   double *xi, double *yi,
 *   // 2nd intersection point
 *   double *xi_prime, double *yi_prime)
 *
 * This is a public domain work. 3/26/2005 Tim Voght
 *
 */
#include <stdio.h>
#include <math.h>

int circle_circle_intersection(double x0, double y0, double r0,
                               double x1, double y1, double r1,
                               double *xi, double *yi,
                               double *xi_prime, double *yi_prime)
{
  double a, dx, dy, d, h, rx, ry;
  double x2, y2;

  /* dx and dy are the vertical and horizontal distances between
   * the circle centers.
   */
  dx = x1 - x0;
  dy = y1 - y0;

  /* Determine the straight-line distance between the centers. */
  //d = sqrt((dy*dy) + (dx*dx));
  d = hypot(dx,dy); // Suggested by Keith Briggs

  /* Check for solvability. */
  if (d > (r0 + r1))
  {
    /* no solution. circles do not intersect. */
    return 0;
  }
  if (d < fabs(r0 - r1))
  {
    /* no solution. one circle is contained in the other */
    return 0;
  }

  /* 'point 2' is the point where the line through the circle
   * intersection points crosses the line between the circle
   * centers.
   */

  /* Determine the distance from point 0 to point 2. */
  a = ((r0*r0) - (r1*r1) + (d*d)) / (2.0 * d) ;

  /* Determine the coordinates of point 2. */
  x2 = x0 + (dx * a/d);
  y2 = y0 + (dy * a/d);

  /* Determine the distance from point 2 to either of the
   * intersection points.

```
   */
  h = sqrt((r0*r0) - (a*a));

  /* Now determine the offsets of the intersection points from
   * point 2.
   */
  rx = -dy * (h/d);
  ry = dx * (h/d);

  /* Determine the absolute intersection points. */
  *xi = x2 + rx;
  *xi_prime = x2 - rx;
  *yi = y2 + ry;
  *yi_prime = y2 - ry;

  return 1;
}

#define TEST

#ifdef TEST

void run_test(double x0, double y0, double r0,
              double x1, double y1, double r1)
{
  double x3, y3, x3_prime, y3_prime;

  printf("x0=%F, y0=%F, r0=%F, x1=%F, y1=%F, r1=%F :\n",
         x0, y0, r0, x1, y1, r1);
  circle_circle_intersection(x0, y0, r0, x1, y1, r1,
                             &amp;x3, &amp;y3, &amp;x3_prime, &amp;y3_prime);
  printf("  x3=%F, y3=%F, x3_prime=%F, y3_prime=%F\n",
         x3, y3, x3_prime, y3_prime);
}

int main(void)
{
  /* Add more! */
  run_test(-1.0, -1.0, 1.5, 1.0, 1.0, 2.0);
  run_test(1.0, -1.0, 1.5, -1.0, 1.0, 2.0);
  run_test(-1.0, 1.0, 1.5, 1.0, -1.0, 2.0);
  run_test(1.0, 1.0, 1.5, -1.0, -1.0, 2.0);
  exit(0);
}
#endif

</pre></body></html>
```