```
// FILE: areaCentroid1_0.c - Works!
// DATE: 02/26/20 09:44
// AUTH: P.Oh
// DESC: Report area and centroid of RAW image
// REFS: areaCentroid0_1b.c

#include<stdlib.h>
#include<stdio.h>
#include<memory.h>
#include<math.h>

#define pix(Im, x, y)   *(Im->Data + (x)*Im->Cols + (y))
#define WHITE           255
#define BLACK           0

struct Image {
    int Rows, Cols;       // image's number of rows and columns
    unsigned char *Data;  // pointer to image data
}; // end of struct Image

struct coord {
  float x, y;             // result's row and column coordinates
};

void Img_in(struct Image *Img) {
    FILE *ifile;
    int i;

    // NB: Assumes RAW image file 256 x 256 size
    // Open file for binary reading
    // Assumes RAW file in same directory as this C-program
    ifile = fopen("16x16-ballRaw.raw", "rb"); // read binary file

    // Read directly into the image array
    for(i=0; i < Img->Rows; ++i)
        fread(Img->Data + i*Img->Cols, Img->Cols, 1, ifile);

    fclose(ifile);

} // end Img_in

void Img_out(struct Image *Out) {
    FILE *ofile;
    int i;

    // Open (or create) binary file for writing
    ofile = fopen("thresholdOutput.raw", "wb");
    // Output the image by rows
    for(i=0; i < Out->Rows; ++i)
        fwrite(Out->Data + i*Out->Cols, Out->Cols, 1, ofile);

    fclose(ofile);
} // end Img_out

float area(struct Image *In, int x1, int y1,
                int x2, int y2, unsigned char ObjVal) {
    // returns calculated area of a RAW image
    long i, j;
    float areaValue = 0.0; // although this is an int, will use for float division

    for(i=x1; i <= x2; ++i)
       for(j=y1; j <= y2; ++j) {
          if(pix(In, i, j)==ObjVal)
```

```
            areaValue = areaValue + 1.0;
        }
    return(areaValue);

} // end function area

struct coord centroid(struct Image *In, int x1,
               int y1, int x2, int y2,
                   unsigned char ObjVal) {
    // returns calculated centroid (as struct) of RAW image
    long i, j;
    float calculatedArea;
    int xSum, ySum;
    struct coord calculatedCentroid;

    calculatedArea = area(In, x1, y1, x2, y2, ObjVal);

    if(calculatedArea == 0) {
       calculatedCentroid.x = -1; calculatedCentroid.y = -1;
       return(calculatedCentroid);
    };

    xSum = ySum = 0;

    for(i=x1; i<=x2; ++i)
       for(j=y1; j<=y2; ++j) {
           if(pix(In, i, j) == ObjVal) {
               xSum += j;
               ySum += i;
           }
       }

    calculatedCentroid.x = xSum/calculatedArea;
    calculatedCentroid.y = ySum/calculatedArea;

    return(calculatedCentroid);
} // end function centroid


int main() {

  struct Image In; // Declare input and output images
  struct coord centroidCoordinates;

  int areaImage;

  // Assumes RAW image is 16-by-16 bytes and allocate memory
  In.Rows = 16;
  In.Cols = 16;
  In.Data = (unsigned char *)calloc(In.Rows, In.Cols);

  Img_in(&In);
  areaImage = area(&In, 0, 0, (In.Rows-1), (In.Cols-1), BLACK);
  printf("Area of 16x16 image is: %d\n", areaImage);

  centroidCoordinates = centroid(&In, 0, 0, (In.Rows-1), (In.Cols-1), BLACK);
  printf("Centroid is (x,y) = (%3.3f, %3.3f)\n", centroidCoordinates.x,
centroidCoordinates.y);

} // end of main
```