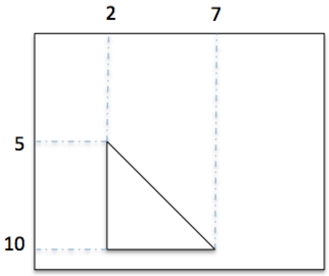
Introduction to Software Design
Fractals

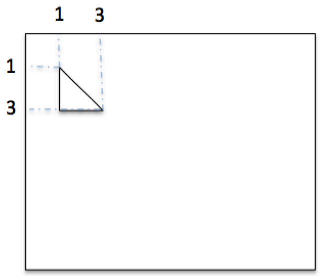
In this problem set, we'll assume that there's a graphics library available that will allow us to draw triangles and squares on the screen. We'll assume that the following functions are available:

```
 /*
  *    Purpose:  draws a right-angled, isosceles triangle on the screen.
  *              The top left corner of the screen is mapped to (0,0).
  *    Param:    int x - x-coordinate of the upper vertex
  *    Param:    int y - y-coordinate of the upper vertex
  *    Param:    int size - length of the equal/shorter sides
  */
void triangle(int x, int y, int size);
```

Sample output for `triangle` follows:

`triangle(2, 5, 5);`   `triangle(1, 1, 2);`

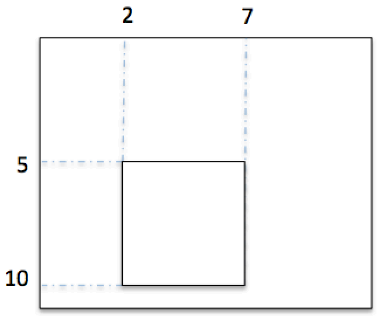


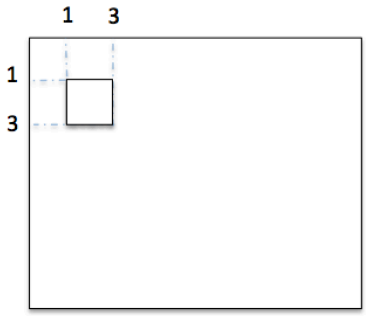


```
 /*
  *    Purpose:  draws a square on the screen.
  *              The top left corner of the screen is mapped to (0,0).
  *    Param:    int x - x-coordinate of the upper left vertex
  *    Param:    int y - y-coordinate of the upper left vertex
  *    Param:    int size - length of the sides
  */
void square(int x, int y, int size);
```

Sample output for `square` follows:

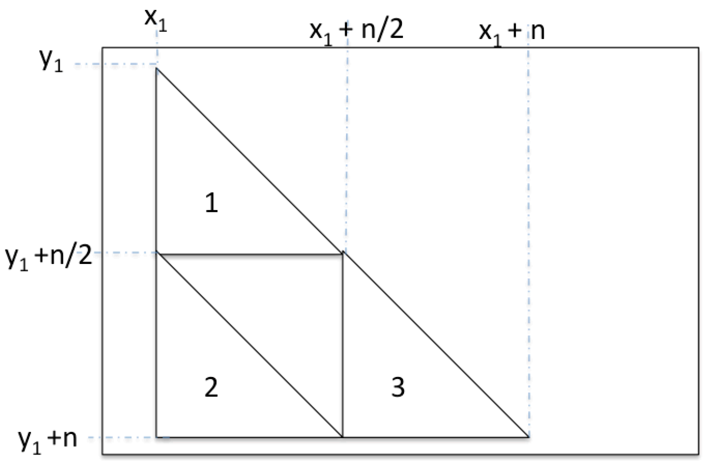
`square(2, 5, 5);`   `square(1, 1, 2);`





1 (a) Write a function `simpleTri` that draws the illustrated picture using three right-angled, isosceles triangle. The (x, y) coordinates of the upper vertex, and an integer that specifies the `size` of the picture are taken as parameters. Sample output for `simpleTri` follows:

`simpleTri(x1, y1, n);`



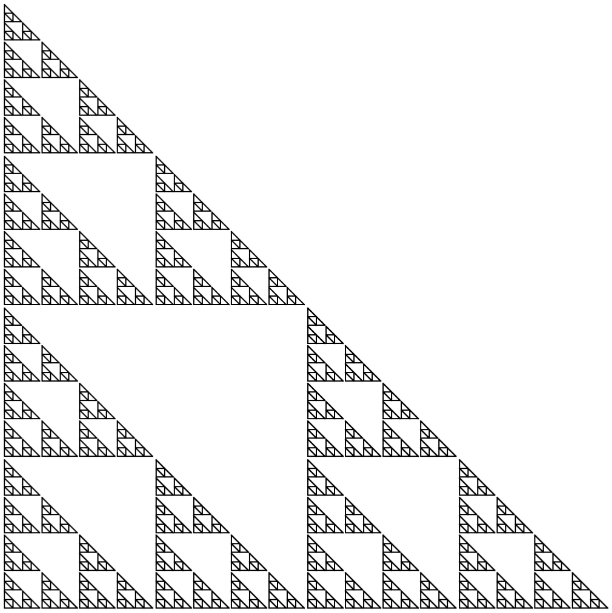
Note: This question is not to be answered recursively.

```
/*
 *    Purpose: draws a simple picture using triangles as
 *             illustrated in the Fractals worksheet.
 *    Param:   int x – x-coordinate of the upper vertex
 *    Param:   int y – y-coordinate of the upper vertex
 *    Param:   int size – size of the picture
 */

void simpleTri (int x, int y, int size){
   triangle (x, y, size/2);
   triangle(x, y+size/2, size/2);
   triangle(x+size/2, y+size/2, size/2);
}
```

1 (b) Write a function `fancyTri` that draws a fancy picture using triangles, as illustrated below. The (x, y) coordinates of the upper vertex, and an integer that specifies the `size` of the picture are taken as parameters. The size of the biggest triangle used must be smaller than 10.

`fancyTri(0, 0, 500);`

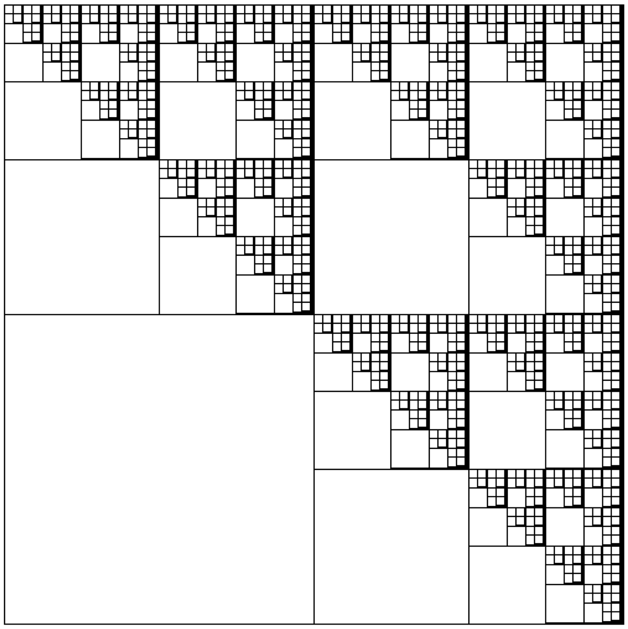


```
/*
  *    Purpose: draws a fancy picture using triangles as illustrated in
  *             the Fractals worksheet.
  *
  *    Param:   int x - x-coordinate of the upper vertex
  *    Param:   int y - y-coordinate of the upper vertex
  *    Param:   int size - size of the picture
  */

void fancyTri(int x, int y, int size){
   if(size<10)
      triangle (x, y, size);
   else{
     fancyTri(x, y, size/2);
     fancyTri(x, y+size/2, size/2);
     fancyTri(x+size/2, y+size/2, size/2);
    }
}
```

2 (a) Write a function `fancySquare` that draws a fancy picture using squares, as illustrated below. The (x, y) coordinates of the upper vertex, and an integer that specifies the `size` of the picture are taken as parameters. The size of the biggest square used must be smaller than 10.

`fancySquare(0, 0, 500);`



```
 /*
  *    Purpose: draws a fancy picture using squares as illustrated in
  *             the Fractals worksheet.
  *    Param:   int x – x-coordinate of the upper left vertex
  *    Param:   int y – y-coordinate of the upper left vertex
  *    Param:   int size – size of the picture
  */
void fancySquare(int x, int y, int size){
   if(size<10){
      square(x, y, size);
   }
   else{
      fancySquare(x, y, size/2);
      fancySquare(x+size/2, y, size/2);
      fancySquare(x+size/2, y+size/2, size/2);
      }
 }
```