Examples of Procedural Recursion

This handout contains code for the file reverse, Sierpinski triangle fractal, and the Towers of Hanoi. Having the printed code is intended to make it easier to follow in class, but as these examples are somewhat sparsely commented, you should make sure to keep up with the discussion in lecture. The tower example is fully covered in your textbook as well.

The implementation of **`ReverseFile`** looks like this:

```
void ReverseFile(FILE *in, FILE *out)
{
    int ch;

    if ((ch = getc(in)) != EOF) {
        ReverseFile(in, out);
        putc(ch, out);
    }
}
```

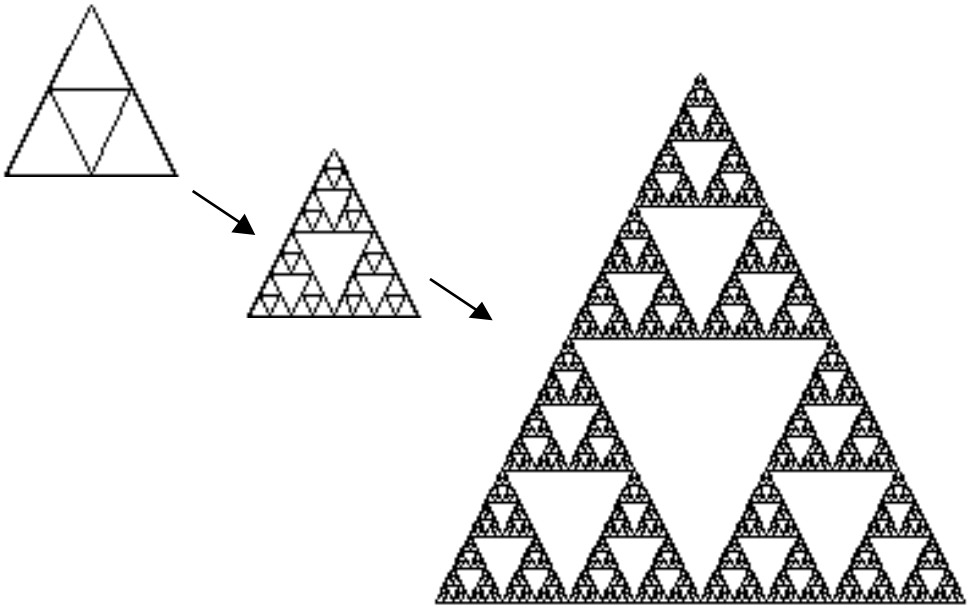
```
#define TooSmall 0.2  /* size at which hits base case */

/*
 * FractalTriangle
 * ---------------
 * A recursive Sierpinski graphic.  The version draws the outer triangle
 * first and then recurs for the three smaller triangles.  Alternatively,
 * we could make the recursive calls and then draw the outer triangle-- the
 * final picture would be the same either way, but the pattern in which the
 * drawing materialized would be different. The base case is when we have a
 * triangle too small to recursively dissect, in which case we just draw
 * the triangle and stop.
 */
static void FractalTriangle(double x, double y, double width, double height)
{
     double halfH = height/2, halfW = width/2;

     DrawTriangle(x, y, width, height);
     if (width < TooSmall || height < TooSmall) return; // stop here

     FractalTriangle(x, y, halfW, halfH);                    // left
     FractalTriangle(x + halfW/2,y + halfH, halfW, halfH);   // top
     FractalTriangle(x + halfW, y, halfW, halfH);            // right
}

static void DrawTriangle(double x, double y, double width, double height)
{
     MovePen(x, y);
     DrawLine(width, 0);
     DrawLine(-width/2.0, height);
     DrawLine(-width/2.0, -height);
 }
```



```
/*
 * File: hanoi.c
 * -------------
 * This program implements a tower-moving algorithm based on the classic
 * Towers of Hanoi problem.  We've got a stack of graduated disks to move from
 * Peg A to Peg C, with one intermediate storage Peg B. The rules are that
 * we can only move one disk at a time and we can only stack a disk on top
 * of a larger disk. We show the progress by printing each time we move
 * a disk from tower to tower.
 */

static void MoveTower(int height, char source, char dest, char temp);
static void MoveSingleDisk(char source, char dest);

main()
{
      MoveTower(5, 'A', 'B', 'C');
}


/*
 * Function: MoveTower
 * -------------------
 * Function to move a tower of height from the source peg to destination peg
 * using temp as intermediate storage, following the rules of the game.
 * We take the recursive leap of faith and assume we can move off the top n-1
 * disks from source to temp using destination as intermediate, then move the
 * bottommost disk from source to dest, then recursively pile on the n-1 tower
 * we earlier moved aside.  Our base case is the empty tower, 0 disks, which
 * requires no work.  The single height tower is handled just as any other
 * recursive call (make things as simple as possible!)
*/
static void MoveTower(int height, char source, char dest, char temp)
{
      if (height > 0) {
         MoveTower(height-1, source, temp, dest);
         MoveSingleDisk(source, dest);
         MoveTower(height-1, temp, dest, source);
      }
}

static void MoveSingleDisk(char source, char dest)
{
      printf("Moving disk from %c to %c\n", source, dest);
}
```