This document is written using MAPLE.

In order to make fractals with iterated function systems it is nice to have a test procedure to make sure you have picked your affine maps correctly (and to help you adjust them later if necessary.) The procedure TESTMAP below, takes a list of affine functions as its input, and the result is a mapping -L picture like the ones we drew in class.

```maple
> restart: with(plots): Digits:=4:

> TESTMAP:=proc(funclist) 
  local num, # the number of functions
    i, # dummy index
    F, # current function in list
    S, # corners of unit square
    L, # corners of letter L
    Sq, # unit square
    Llet, # letter L
    AS, # transf of square corners
    ASq, # transf of square
    AL, # transf of L corners
    ALlet, # transf of letter L
    Pics; # a list of pictures
  S:=[[0,0],[0,1],[1,1],[1,0]]; # four corners of unit square
  L:=[[.1,.9],[.1,.75],[.2,.75],[.2,.75],[.125,.775],[.125,.9]]:
  Sq:=polygonplot(S): # polygonplot connects the dots!
  Llet:=polygonplot(L):
  display({Sq, Llet});
  num:=nops(funclist):
  for i from 1 to num do
    F:=funclist[i]: # select ith map
    AS[i]:=map(F, S):
    AL[i]:=map(F, L):
    ASq[i]:=polygonplot(AS[i]):
    ALlet[i]:=polygonplot(AL[i]):
      # a plot of the transformed square and letter:
    Pics[i]:=display({ASq[i], ALlet[i]}):
  od:
  # finally, display the unit square and all its images:
  display({Sq, seq(Pics[i], i=1..num)}, scaling=constrained,
        title=`fractal template`);
end;
```
Here is the standard affine map, which encodes
\[
\begin{bmatrix}
 a & b \\
 c & d \\
 e & f \\
\end{bmatrix}
\begin{bmatrix}
 x \\
 y \\
\end{bmatrix}
\]

And in case you want to use it, an alternative version called AFFINE2 which lets you specify scaling factors and rotation angles instead;
\[
\begin{bmatrix}
 x \\
 y \\
\end{bmatrix}
\begin{bmatrix}
 r \cos(\alpha) & -s \sin(\beta) \\
 r \sin(\alpha) & s \cos(\beta) \\
\end{bmatrix}
\begin{bmatrix}
 x \\
 y \\
\end{bmatrix}
\]

Example 1:
We illustrate the use of TESTMAP with the Sierpinski triangle, and then on the harder twig example.
\[
> \text{f1:=P->AFFINE1(P,.5,0,0,.5,0,0);}
\]
\[
> \text{f2:=P->AFFINE1(P,.5,0,0,.5,.5,0);}
\]
\[
> \text{f3:=P->AFFINE1(P,.5,0,0,.5,.25,.5);}
\]
\[
> \text{TESTMAP([f1,f2,f3]);}
\]
That’s the right blueprint, so now we will generate the fractal, beginning with a single point.
\[
> \text{S:=\{[0,0]\};#initial set consisting of one point}
\]
\[
> \text{3^9; #good to keep point numbers below 100,000,}
\]
\[
> \text{#because Maple is not the most efficient calculator}
\]
\[
> \text{for i from 1 to 9 do}
\]
\[
> \text{S1:=map(f1,S);}
\]
\[
> \text{S2:=map(f2,S);}
\]
\[
> \text{S3:=map(f3,S);}
\]
\[
> \text{S:=\‘union\’(S1,S2,S3);}
\]
\[
> \text{pointplot(S,symbol=point,scaling=constrained,}
\]
\[
> \text{title=\‘Sierpinski Triangle\’);}
\]

Example 2:
To get this example to run correctly, you must use "restart" (at the top of the file) to clear out the memory, and then re-enter the various procedures you want to use again. To do this, place the cursor at the end of each command sequence that you want to run and hit "return". (You will need to enter everything up to the start of Example 1. Then move your cursor to the end of the first command line in Example 2 and enter the affine maps for the twig.)
> f1:=P->AFFINE1(P,.4,.4,.4,-.4,.24,.55);
f2:=P->AFFINE1(P,.45,.01,.09,.01,-.05,.31);
f3:=P->AFFINE1(P,.45,-.1,0,-.3,.4,.48);

> TESTMAP([f1,f2,f3]);

> S:={[0,0]};
> for i from 1 to 9 do
> S1:=map(f1,S):
> S2:=map(f2,S):
> S3:=map(f3,S):
> S:=`union`(S1,S2,S3);
> od:

> pointplot(S,scaling=constrained,symbol=point,
> title=`twig`,axes=none);