The operation \( R_1 \) which takes the point \((x, y)\) into the point \((x, -y)\) is usually referred to as a reflection across the \(x\)-axis. Similarly, the operation \( R_2 \) which takes the point \((x, y)\) into the point \((-x, y)\) is usually referred to as a reflection across the \(y\)-axis. Similarly, the operation \( R_3(x, y) = (y, x) \) is a reflection across the line \(x = y\). If we take an arbitrary curve which starts on the \(x\)-axis, ends on the line \(x = y\) and is otherwise contained in the wedge between these two lines, then apply to it in succession \( R_1, R_2, R_1, R_2 \), then apply \( R_3 \) to each of the four curves we obtain a picture consisting of eight curves which, no matter how ugly the original curve was, turns out to be pleasing to the eye. The reason for this phenomenon is due to the fact that our brain likes symmetry. We can thus take advantage of this fact in creating nice pictures out of ugly ones. In fact, this is the principle behind the kaleidoscope, which transforms garbage into jewels.

For this assignment, you must use four procedures \texttt{create}, \texttt{bezier}, \texttt{reflect} and \texttt{disp}. The first should have heading \texttt{create[a_, n_]} and its output should be a sequence of the form

\[
\{\{x_1, y_1\}, \{x_2, y_2\}, \ldots , \{x_n, y_n\}\}
\]

where all \(x_i, y_i\) should be random integers in the interval \([0,2a]\) with \(x_i \leq y_i\).

Alternatively you may use the David Little applet to create your curve, or even use the random number generator \texttt{ra[] := Random[]}. The sequence pairs you have constructed are to be used as control points, to be fed as input to the procedure \texttt{bezier}, that we have seen in class. If you forgot it, here it is again

\[
\texttt{bezier[sec_]} := (n = \texttt{Length[sec]} - 1 ;
\texttt{cur = Sum[sec[[k + 1]] Binomial[n, k] t^k(1 - t)^(n - k), \{k, 0, n\}]} ;
\texttt{Return[cur]})
\]

The procedure \texttt{reflect} should have heading \texttt{reflect[curve_] and a call of \texttt{reflect}[[f, g]] should return the eight pairs}

\[
\{\{f, g\}, \{f, -g\}, \{-f, g\}, \{-f, -g\}, \{g, f\}, \{g, -f\}, \{-g, f\}, \{-g, -f\}\}
\]

This done you are ready to produce a variety of snowflakes using the principle of the Kaleidoscope. Construct several random curves by feeding into \texttt{bezier} the outputs of \texttt{create} called with different values of \(a\) and \(n\). This done feed each of these curves to \texttt{reflect} (that is the kaleidoscope) and then using the MATHEMATICA “Join” command collect all the resulting sequences of curves into a single sequence and display the whole shebang by means of the procedure given below

\[
\texttt{disp[curve_]} := \texttt{ParametricPlot[}
\texttt{Evaluate[curve], \{t, 0, 1\},}
\texttt{AspectRatio -> Automatic, Axes -> True, PlotRange -> All]}
\]
This done, post on your website your procedures and a few of the best snowflakes you have been able to create.

Note that the two displays below have been obtained from a single bezier curve, constructed from the David Little Applet. If you observe the picture on the right appears to have a shadow. This is simply obtained by displaying two copies of the same curve with the second copy shifted with respect to the other and with the color reduced to gray.

Have fun.