Problem 1: Lists and Sets in Maple

a)

```
restart;
a := proc(n)
 local L, i :
 L := NULL :
 i := 1 :
 while i! < n do
  L := L, i!:
  i := i + 1 :
 od:
 return [L] :
end proc:

a(7);
```

$$[1, 2, 6] \qquad \textbf{(1.1.1)}$$

b) Alternative 1

```
restart;
with(combinat) :
b := proc(M, k)
 local S, T, R :
 R := NULL :
 S := subsets(M) :
 while not S_finished do
  T := S_nextvalue( ) :
  if nops(T) ≠ k then
   R := R, T :
  fi:
 od:
 return {R} :
end proc:

b({1, 2, 3, 4}, 2);
```

$$\{\{\}, \{1\}, \{2\}, \{3\}, \{4\}, \{1, 2, 3\}, \{1, 2, 4\}, \{1, 3, 4\}, \{2, 3, 4\}, \{1, 2, 3, 4\}\} \qquad \textbf{(1.2.1)}$$

b) Alternative 2

```
restart;
with(combinat) :
b2 := proc(M, k)
```

```
local S, T, R :
R := NULL :
S := powerset(M) :
for T in S do
 if nops(T) ≠ k then
  R := R, T :
 fi:
od:
return {R} :
end proc:

b2({1, 2, 3, 4}, 2);
```

$$\{\{\}, \{1\}, \{2\}, \{3\}, \{4\}, \{1, 2, 3\}, \{1, 2, 4\}, \{1, 3, 4\}, \{2, 3, 4\}, \{1, 2, 3, 4\}\} \quad \textbf{(1.3.1)}$$

b) Alternative 3

```
restart;
with(combinat) :
b3 :=proc(M, k)
 return powerset(M)\choose(M, k) :
end proc:

b3({1, 2, 3, 4}, 2);
```

$$\{\{\}, \{1\}, \{2\}, \{3\}, \{4\}, \{1, 2, 3\}, \{1, 2, 4\}, \{1, 3, 4\}, \{2, 3, 4\}, \{1, 2, 3, 4\}\} \quad \textbf{(1.4.1)}$$

Problem 2: Decimal Expansion of Rational Numbers

```
restart;
DecimalExpansion :=proc(aa, bb)
 local a, b, rem, g, i, j, k, dd, decstring, remarray, decarray, decstringtmp;
 g := 10;
 decstring := "";
 decstringtmp := "";

 if (bb = 0) then return undefined; end if;

 if (aa < 0 xor bb < 0) then
  decstring := "-";
 end if;
 a := abs(aa);
 b := abs(bb);
 remarray := Array(1 ..b);
 decarray := Array(1 ..b);

 if (a ≥ b) then
  rem := irem(a, b, 'dec');
  decstring := cat(decstring, dec, ".");
 else
```

```
  rem := a;
  decstring := cat(decstring, "0.");
end if;

# print("remainder:", rem);

rem := 10·rem;
for i from 1 to infinity do
  rem := irem( rem, b,'dd');

  # print("remainder:", rem);
  # print(dd);

  # check for period
  for j from 1 to i − 1 do
    if (rem = remarray[j]) then
      # print("found period");
      if (decarray[j] ≠ dd) then
        # index j does not belong to the period
        decarray[i] := dd;
        for k from 1 to j do
          decstringtmp := cat(decstringtmp, decarray[k]);
        end do;
        decstringtmp := cat(decstringtmp,'p');
        for k from j + 1 to i do
          decstringtmp := cat(decstringtmp, decarray[k]);
        end do;
        return cat(decstring, decstringtmp);
      else
        # index j belongs to period
        for k from 1 to j − 1 do
          decstringtmp := cat(decstringtmp, decarray[k]);
        end do;
        decstringtmp := cat(decstringtmp, "p");
        for k from j to i − 1 do
          decstringtmp := cat(decstringtmp, decarray[k]);
        end do;
        return cat(decstring, decstringtmp);
      end if;
    end if;
  end do;

  remarray[i] := rem;
  decarray[i] := dd;

  # check if the expansion is finite
  if (rem = 0) then
    for j from 1 to i do
      decstringtmp := cat(decstringtmp, decarray[j]);
    end do;
    return cat(decstring, decstringtmp);
  end if;
```

```
    rem := 10·rem;
  end do;

  return undefined;
end proc:
```

```
DecimalExpansion(1, 1); DecimalExpansion(-3, 4); DecimalExpansion(1, 3);
    DecimalExpansion(1, 6); DecimalExpansion(-1, 700);
```

"1.0"

"-0.75"

"0.p3"

"0.1p6"

"-0.00p142857" **(2.1)**

Problem 3: Design of a Beer Glass

```
restart;
```

a)

```
p := x→a x^5 + b x^4 + c x^3 + d x^2 + e x + f;
```

$$x \to a\,x^5 + b\,x^4 + c\,x^3 + d\,x^2 + e\,x + f \quad \textbf{(3.1.1)}$$

```
erg := solve({p(0) = 11/4, p(6) = 19/10, p(18) = 9/2, p(20) = 4, p'(6) = 0, p'(18) = 0}, {a, b, c,
    d, e, f});
```

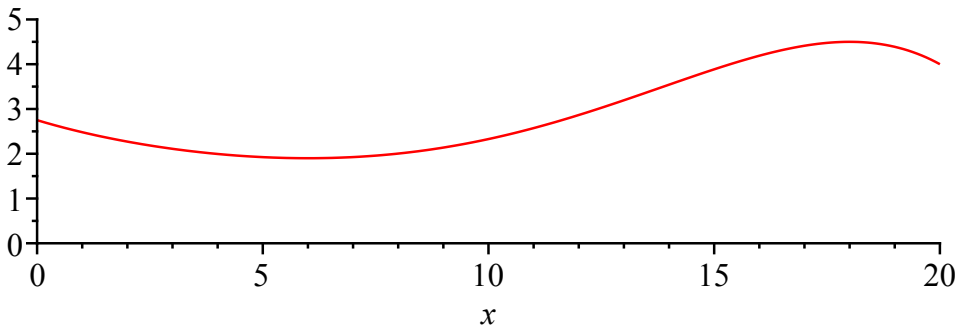
$$\left\{a = -\frac{59}{6531840},\ b = \frac{463}{1632960},\ c = -\frac{403}{136080},\ d = \frac{103}{2835},\ e = -\frac{305}{1008},\ f = \frac{11}{4}\right\} \quad \textbf{(3.1.2)}$$

```
assign(erg);
p(x);
```

$$-\frac{59}{6531840}\,x^5 + \frac{463}{1632960}\,x^4 - \frac{403}{136080}\,x^3 + \frac{103}{2835}\,x^2 - \frac{305}{1008}\,x + \frac{11}{4} \quad \textbf{(3.1.3)}$$

b)

```
plot(p(x), x = 0 .. 20, scaling = constrained, view = [DEFAULT, 0 .. 5]);
```



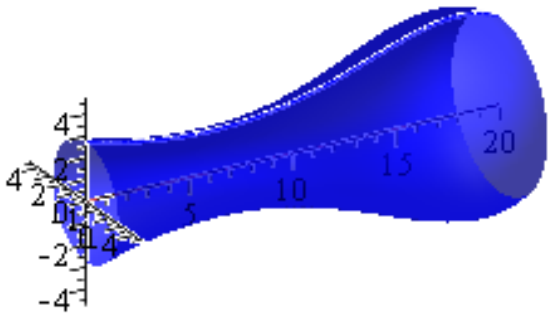
c)

```
with(Student[Calculus1]):
with(plots):
glas := VolumeOfRevolution(p(x), x = 0..20, output = plot, scaling = constrained,
        volumeoptions = [color = blue, transparency = 0.5]);
```

$$PLOT3D(\ldots) \qquad \textbf{(3.3.1)}$$

```
display(glas);
```

The Volume of Revolution Around the Horizontal Axis of
f(x) = -59/6531840*x^5+463/1632960*x^4-403/136080*x^3+103/2835*
x^2-305/1008*x+11/4
on the Interval [0, 20]



d)

$eichstrich := fsolve(VolumeOfRevolution(p(x), x = 0..es) = 500, es);$

18.54183961 **(3.4.1)**

*)

$strich := spacecurve\left([eichstrich, p(eichstrich)\cdot\cos(t), p(eichstrich)\cdot\sin(t)], t = \frac{3}{4}\pi..\pi\right);$

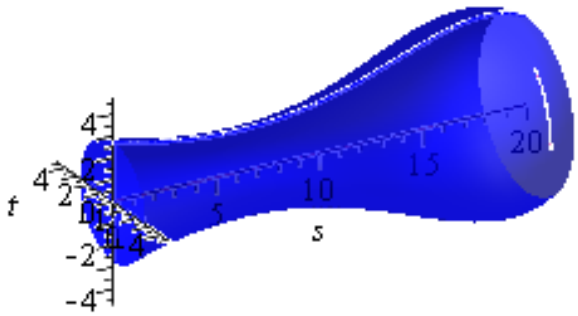
$PLOT3D(...)$ **(3.5.1)**

$boden := plot3d\left(p(0)\cdot[0, s, t], s = -1..1, t = -\sqrt{1-s^2}..\sqrt{1-s^2}, color = blue, transparency = 0.5, style = surface\right);$

$PLOT3D(...)$ **(3.5.2)**

$display([glas, strich, boden]);$

The Volume of Revolution Around the Horizontal Axis of
f(x) = -59/6531840*x^5+463/1632960*x^4-403/136080*x^3+103/2835*
x^2-305/1008*x+11/4
on the Interval [0, 20]



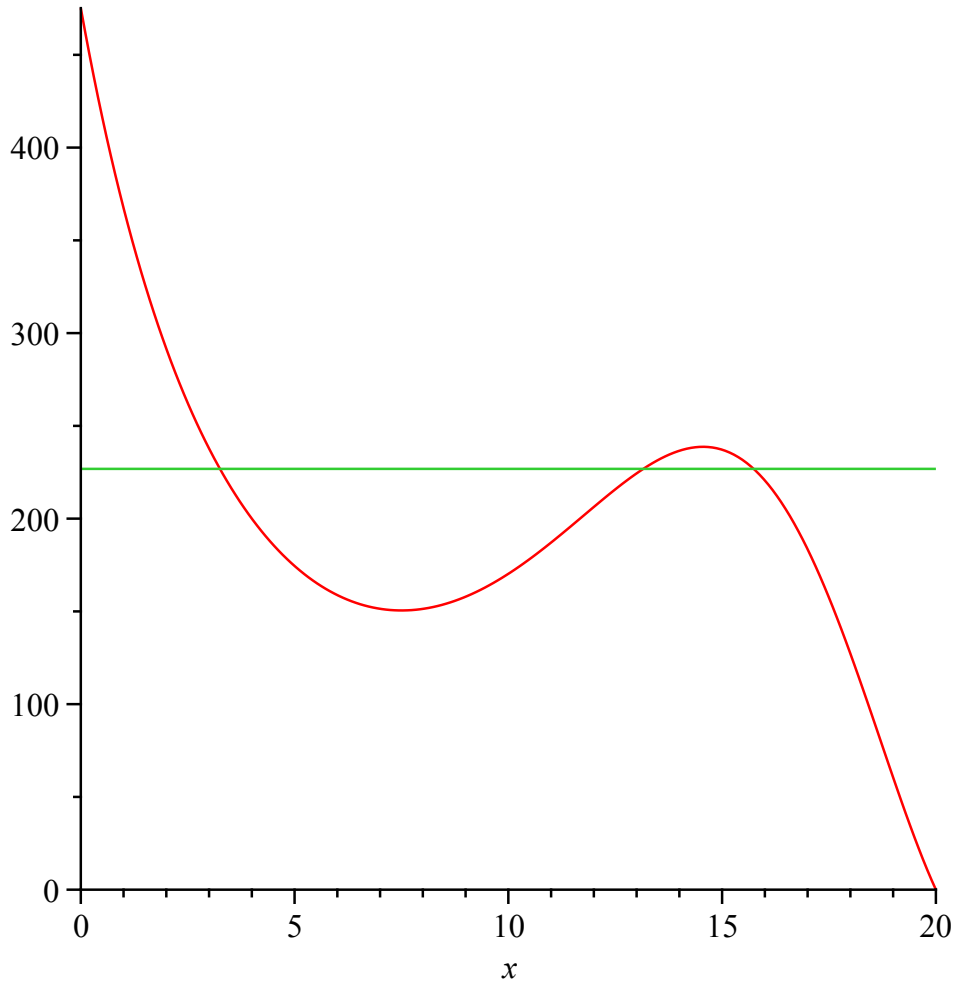
e)

For every L, p(L)·p(L)·π·(20-L) is an upper bound for the volume of the cylinder on the interval [L,20]. Furthermore 19/10·19/10·π·20 is an upper bound for L <= 6, because otherwise, the cylinder would not fit in the glass. Let's have a look at this:

$vtmp := x \to (p(x))^2 \cdot \mathrm{Pi} \cdot (20 - x);$

$$x \to p(x)^2\, \pi\, (20 - x) \qquad \textbf{(3.6.1)}$$

$plot\left(\left[vtmp(x),\ \frac{19}{10} \cdot \frac{19}{10} \cdot \pi \cdot 20\right], x = 0..20\right);$



It might be possible to put the glass over a cylinder of height ≈ 20-15 = 5, whose volume is bigger than that of a cylinder that has height 20. Of course, we have to check whether this is possible.
First of all, let's find this L:

possibleLs := fsolve(diff(vtmp(x), x));

7.502245815, 14.55478011, 21.30252472, 23.08387727 **(3.6.2)**

Obviously, this can only be the second value:

L := possibleLs[2];

14.55478011 **(3.6.3)**

Let's compare p(L) with p(20) to see whether we can put the glass over that cylinder:

p(L);

3.735593423 **(3.6.4)**

p(20);

4 **(3.6.5)**

⇒The glass can be put over the cylinder and no other cylinder can have a bigger volume.

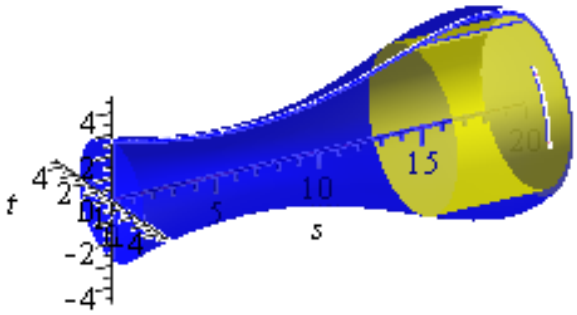
f)

cylinder := VolumeOfRevolution(p(L), x = L..20, output = plot, scaling = constrained,
volumeoptions = [color = yellow, transparency = 0.5]);

PLOT3D(...) **(3.7.1)**

display([*glas*, *strich*, *boden*, *cylinder*]);

The Volume of Revolution Around the Horizontal Axis of
f(x) = -59/6531840*x^5+463/1632960*x^4-403/136080*x^3+103/2835*
x^2-305/1008*x+11/4
on the Interval [0, 20]



g)

CorrectVolume := *VolumeOfRevolution*(*p*(L), x = L..20);

238.7176327 **(3.8.1)**

$$WrongVolume := evalf\left(\frac{19}{10}\cdot\frac{19}{10}\cdot\pi\cdot 20\right);$$

226.8229896 **(3.8.2)**

Problem 4: An Application: Image Processing

restart;
with(*ImageTools*);
[*Checkerboard*, *Clip*, *ColorAImage*, *ColorImage*, *ColorTransform*, *CombineLayers*, **(4.1)**

Complement, *Convolution*, *Create*, *Entropy*, *FitIntensity*, *Flip*, *FormatFromName*, *Formats*, Γ, *GetLayer*, *GetSubImage*, *GrayImage*, *HSVtoRGB*, *Height*, *Histogram*, *Image*, *Intensity*, *Layers*, *Mask*, *PadImage*, *PlotHistogram*, *Preview*, *Quality*, *RGBtoGray*, *RGBtoHSV*, *RGBtoYUV*, *Read*, *Rotate*, *Scale*, *ScaleIntensity*, *SetLayer*, *SetSubImage*, *Stack*, *Threshold*, *ToGrayscale*, *ToRGB*, *ToRGBA*, *Transpose*, *View*, *WhatTypeImage*, *Width*, *Write*, *YUVtoRGB*]

b)

org := *Read*("D:/IMS_tmp/Image1.jpg");

$$\begin{bmatrix} 1..324 \times 1..432 \times 1..3\ Array \\ Data\ Type: float_8 \\ Storage: rectangular \\ Order: C\_order \end{bmatrix} \quad \textbf{(4.1.1)}$$

c)

gray := *ToGrayscale*(*org*);

$$\begin{bmatrix} 1..324 \times 1..432\ Array \\ Data\ Type: float_8 \\ Storage: rectangular \\ Order: C\_order \end{bmatrix} \quad \textbf{(4.2.1)}$$

d)

rot := *Rotate*(*gray*, -90);

$$\begin{bmatrix} 1..432 \times 1..324\ Array \\ Data\ Type: float_8 \\ Storage: rectangular \\ Order: C\_order \end{bmatrix} \quad \textbf{(4.3.1)}$$

View(*rot*);

e)

test := *Create*(*Height*(*rot*), *Width*(*rot*));

$$\begin{bmatrix} 1..432\ x\ 1..324\ Array \\ Data\ Type:\ float_8 \\ Storage:\ rectangular \\ Order:\ C\_order \end{bmatrix} \quad \textbf{(4.4.1)}$$

```
for i from 1 to Height(test) do
 for ii from 1 to Width(test) do
  test[i, ii] := 0.5 :
 od:
od:
# View(test);
```

f)

```
filtered := Create(Height(rot), Width(rot) );
```

$$\begin{bmatrix} 1..432\ x\ 1..324\ Array \\ Data\ Type:\ float_8 \\ Storage:\ rectangular \\ Order:\ C\_order \end{bmatrix} \quad \textbf{(4.5.1)}$$

```
for i from 2 to Height( filtered) - 1 do
 for ii from 2 to Width( filtered) - 1 do
  filtered[i, ii] := Statistics[Median]( [rot[i - 1, ii - 1 ], rot[i - 1, ii], rot[i - 1, ii + 1 ], rot[i, ii
     - 1 ], rot[i, ii], rot[i, ii + 1 ], rot[i + 1, ii - 1 ], rot[i + 1, ii], rot[i + 1, ii + 1 ]]) :
 od:
od:
# View( filtered);
```

g)

$$SobelX := \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix};$$

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad \textbf{(4.6.1)}$$

```
SobelY := LinearAlgebra[Transpose](SobelX);
```

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad \textbf{(4.6.2)}$$

$GX := Convolution(filtered, SobelX);$

$$\begin{bmatrix} 1..432 \text{ x } 1..324 \text{ Array} \\ \text{Data Type: } float_8 \\ \text{Storage: rectangular} \\ \text{Order: C\_order} \end{bmatrix} \quad \textbf{(4.6.3)}$$

$GY := Convolution(filtered, SobelY);$

$$\begin{bmatrix} 1..432 \text{ x } 1..324 \text{ Array} \\ \text{Data Type: } float_8 \\ \text{Storage: rectangular} \\ \text{Order: C\_order} \end{bmatrix} \quad \textbf{(4.6.4)}$$

h)

Maple 14

$GS := \text{sqrt}\sim(GX^2 + GY^2);$

$$\begin{bmatrix} 1..432 \text{ x } 1..324 \text{ Array} \\ \text{Data Type: } float_8 \\ \text{Storage: rectangular} \\ \text{Order: C\_order} \end{bmatrix} \quad \textbf{(4.7.1.1)}$$

Maple 11 (should work with any recent version)

$GS := convert(map(\text{sqrt}, GX^2 + GY^2), Array, order = C\_order, datatype = float_8);$

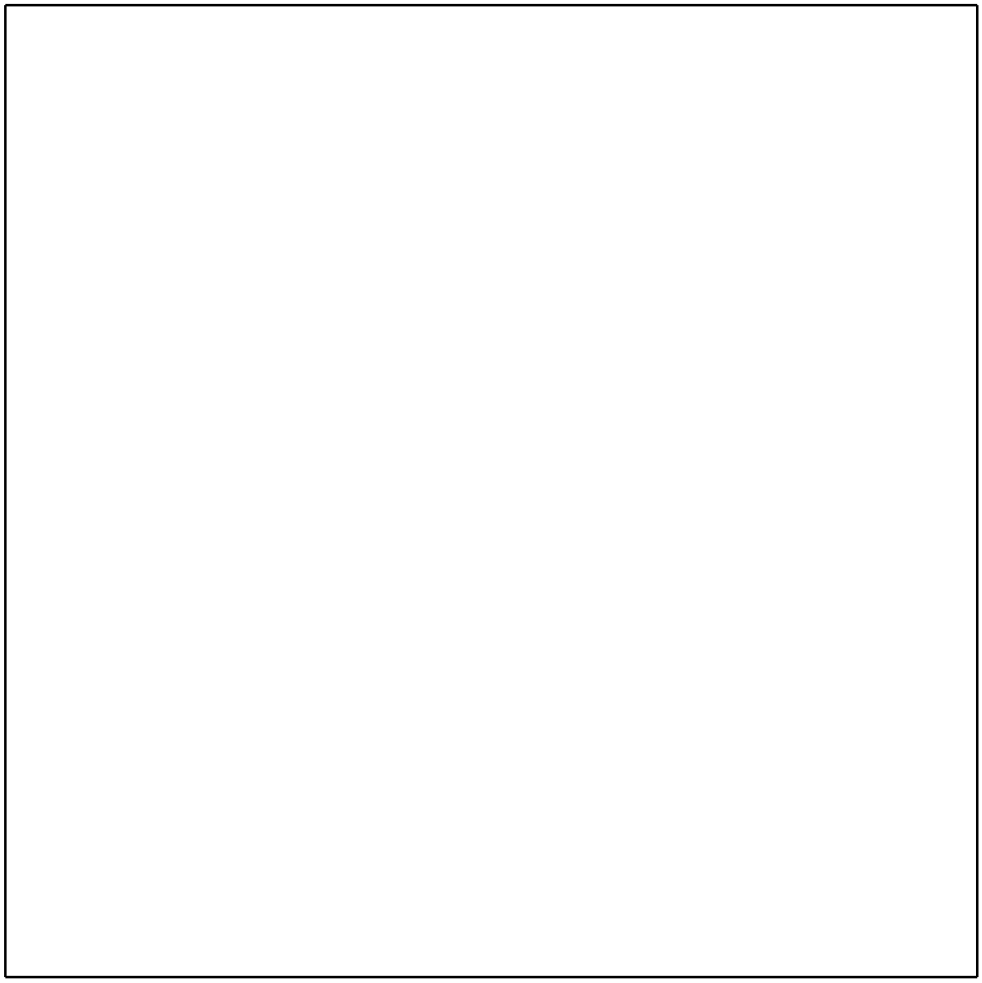
$$\begin{bmatrix} 1..432 \text{ x } 1..324 \text{ Array} \\ \text{Data Type: } float_8 \\ \text{Storage: rectangular} \\ \text{Order: C\_order} \end{bmatrix} \quad \textbf{(4.7.2.1)}$$

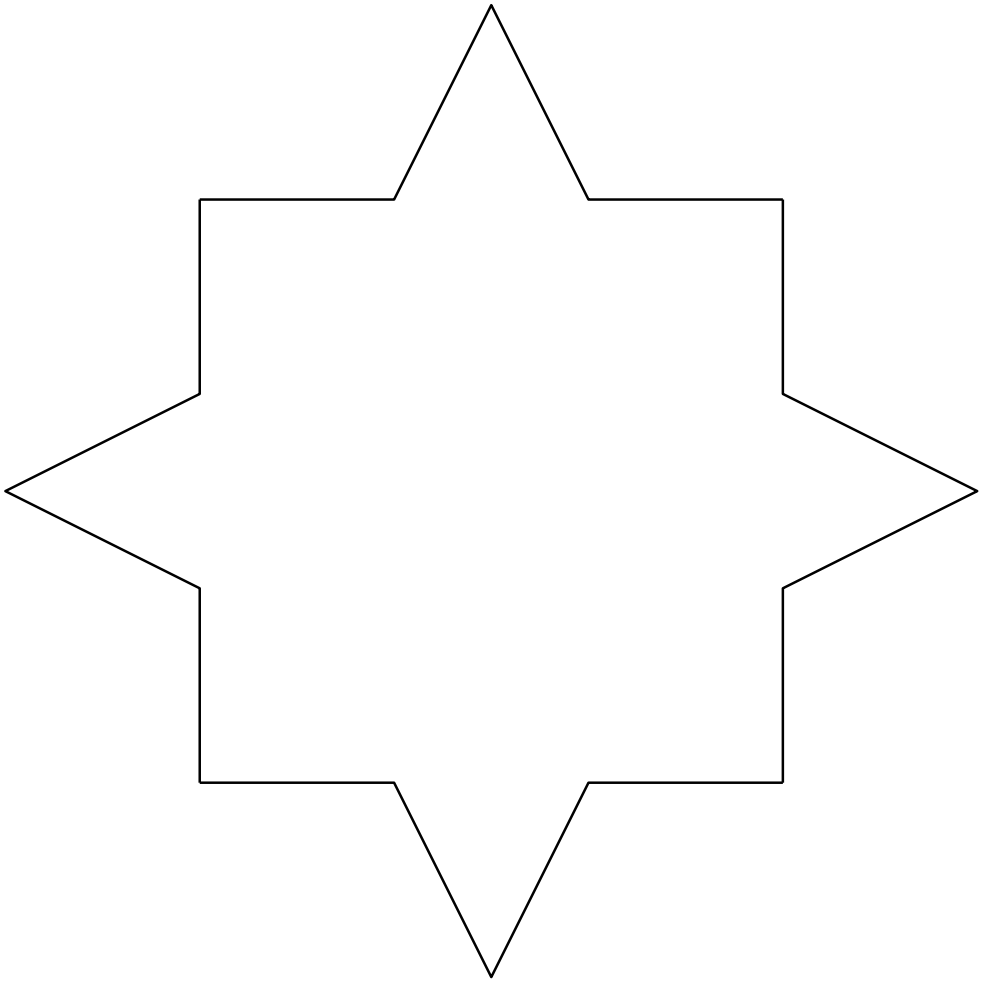
$\#\ View(GS);$

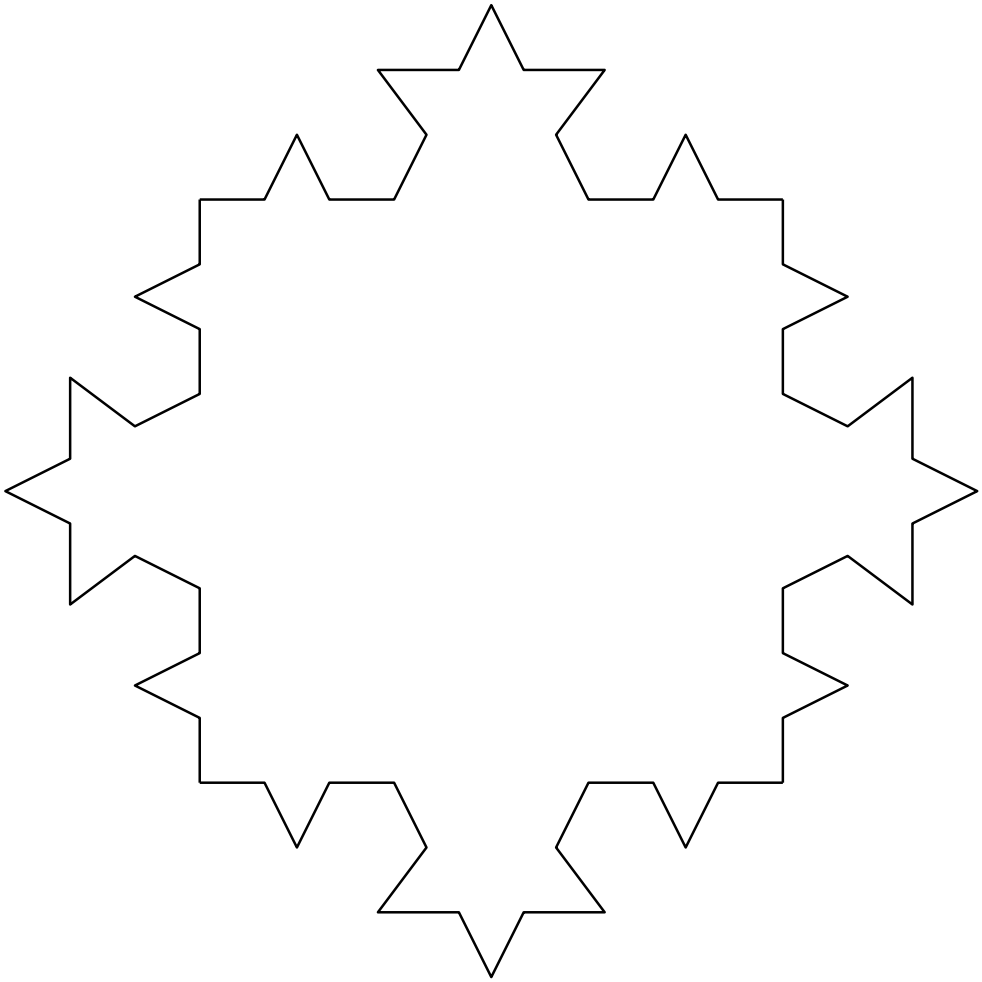
Problem 5: Koch Curve

b)

```
restart;
with(plots) :
with(plottools) :

KochFunction := proc(A, B)
  return A, [(2·A1 + B1)/3, (2·A2 + B2)/3], [(3·B1 + 3·A1 + 2·A2 − 2·B2)/6,
    (3·B2 + 3·A2 + 2·B1 − 2·A1)/6], [(A1 + 2·B1)/3, (A2 + 2·B2)/3], B;
end proc:

KochPoints := proc(f, k)
  local i, points, C;
  points := [[0, 0], [1, 0]];
  for i to k do
    C := zip((x, y) → [x, y], points[1 .. -2], points[2 .. -1]);
    points := map(x→f(op(x))[1 .. -2], C);
    points := [op(points), [1, 0]];
  end do;
  return points;
end proc:

KochPlot := proc(n)
  local points, p1, p2, p3, p4, p5;
  points := KochPoints(KochFunction, n);
  p1 := pointplot(points, connect = true);
  p2 := rotate(pointplot(points, connect = true), π);
  p3 := translate(p2, 1, -1);
  p4 := translate(rotate(pointplot(points, connect = true), π/2), 0, -1);
  p5 := translate(reflect(p4, [1/2, 0], [1, 0]), 0, -1);
  display(p1, p3, p4, p5, scaling = constrained, axes = none);
end proc:

KochPlot(0); KochPlot(1); KochPlot(2);
```







c)

KochFunction

KochFunction takes two parameters, which should be two-dimensional points as list of x- and y-coordinate, e.g.
KochFunction([0.5, 1], [1.5, 1]);
It interpolates these points by adding three additional points inbetween, as seen in
pointplot([%], view = [0 .. 2, 0 .. 2]);
Imagine a linear interpolation between neighboring points. The KochFunction distorted the original line by inserting a peak to the left. By repeating that many times on different edges, the snow flake becomes more detailed.

KochPoints

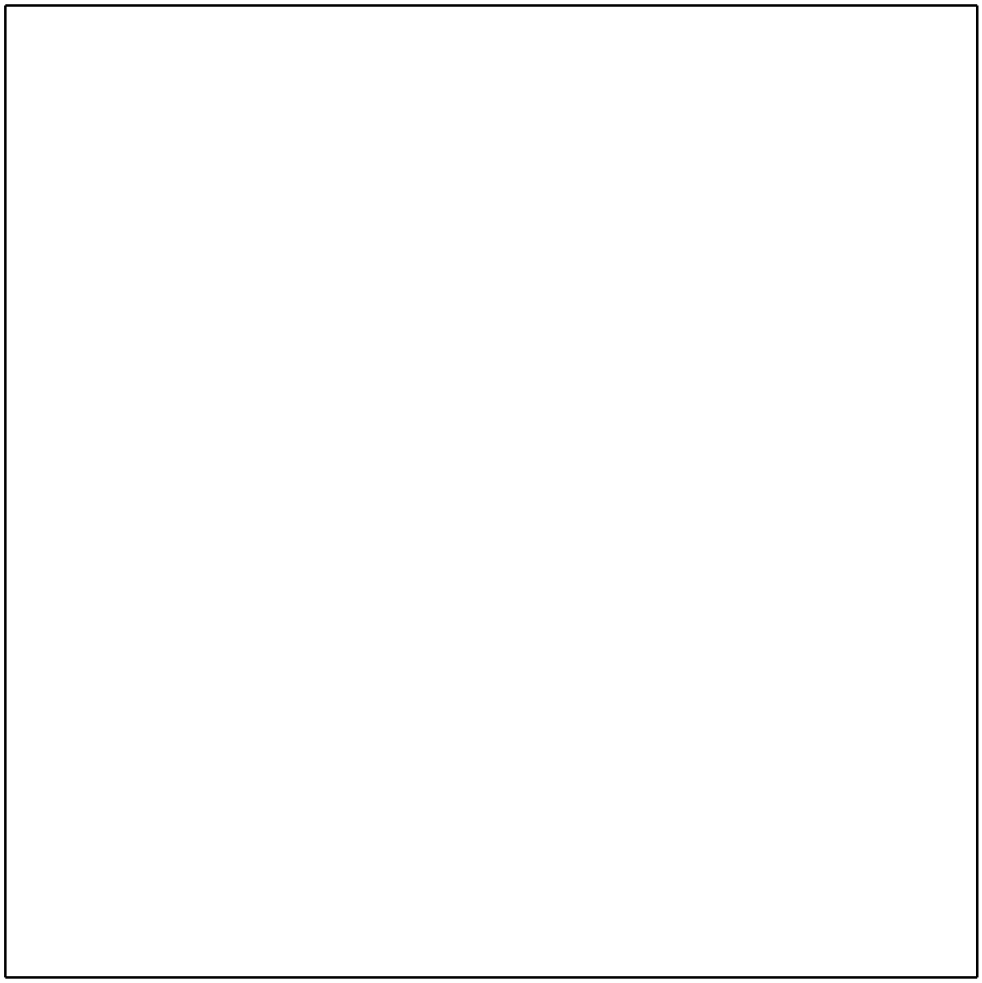
KochPoints generates a segment of the snow flake by repeatedly distorting the initial point list [[0, 0], [1, 0]];
Note, how the loop interates multiple times without using the variable 'i'. instead, the list 'points' is defined in terms of different original 'points' from the last iteration.

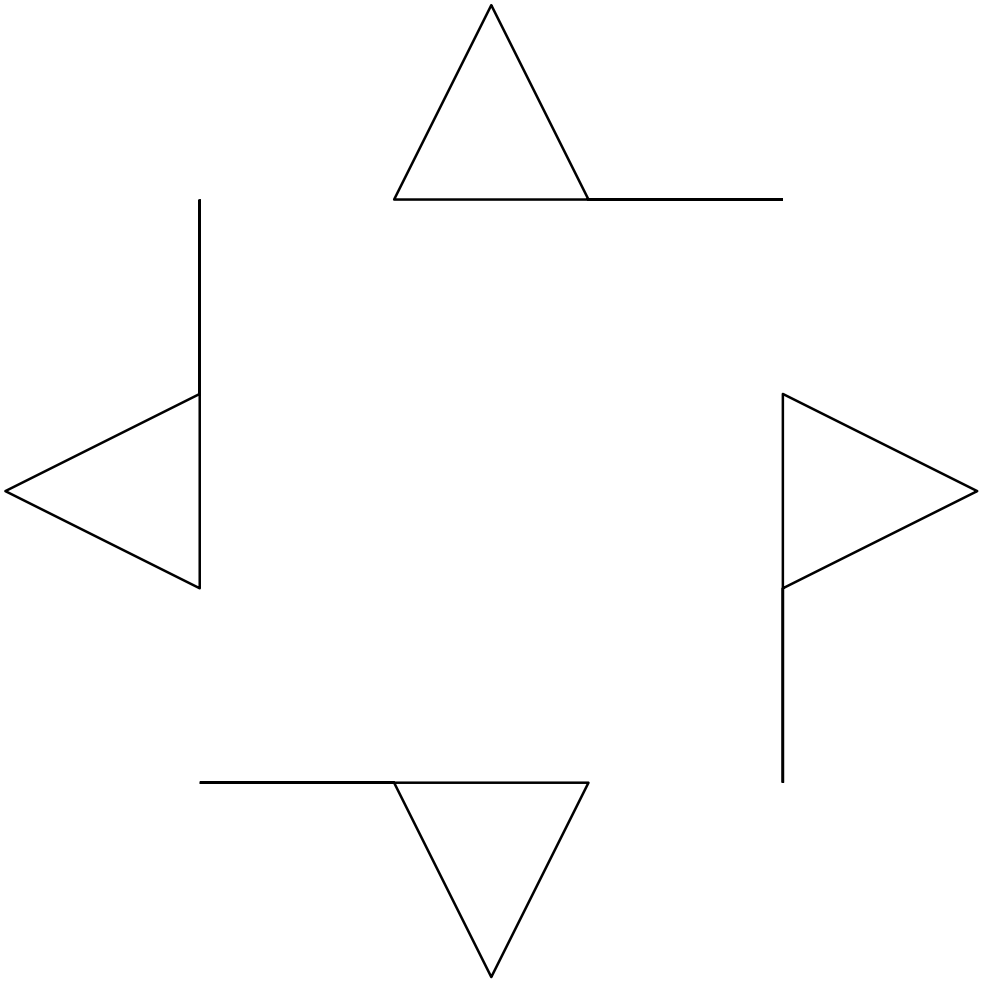
You can visualize the segment of the snow flake by the following commands:
points := KochPoints(KochFunction, 3);
pointplot(points, connect = true);

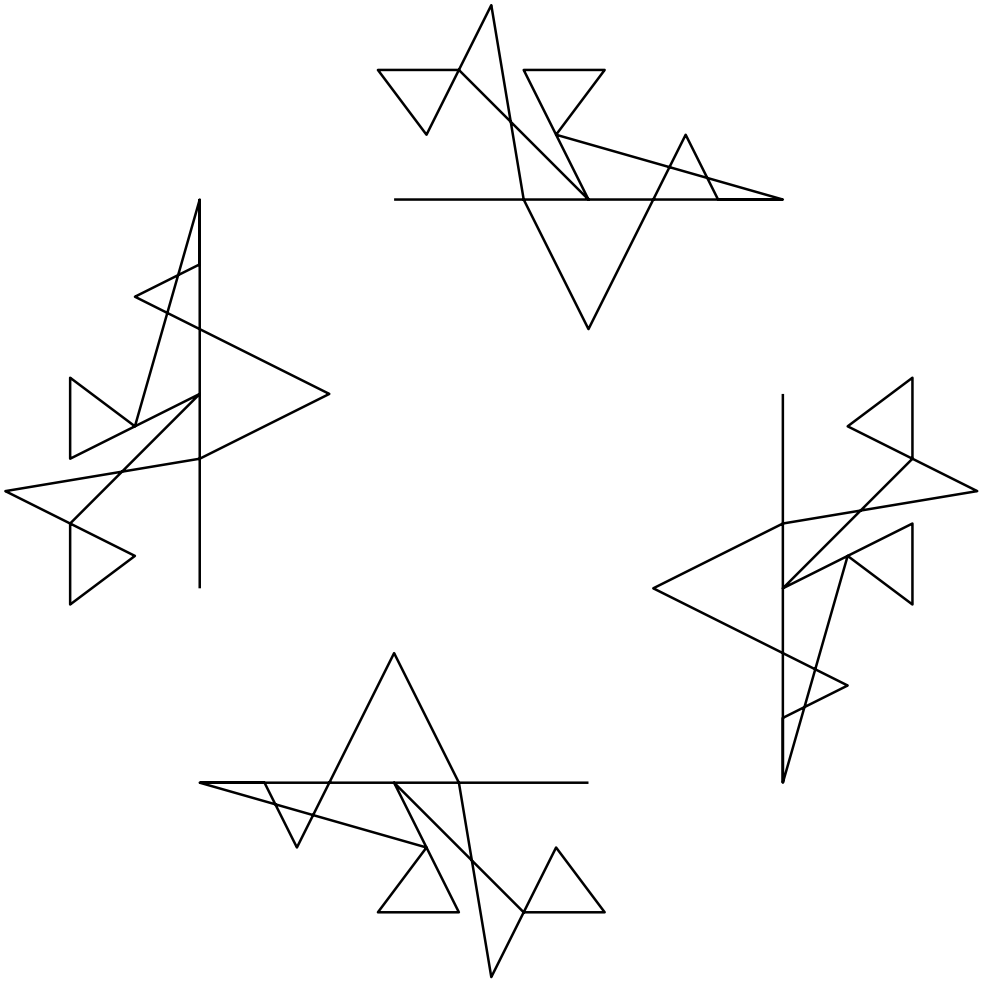
d)

$$KochFunction := \mathbf{proc}(A, B)$$
$$\mathbf{return}\ B, \left[\frac{2\cdot A_1 + B_1}{3}, \frac{2\cdot A_2 + B_2}{3}\right], \left[\frac{3\cdot B_1 + 3\cdot A_1 + 2\cdot A_2 - 2\cdot B_2}{6}, \frac{3\cdot B_2 + 3\cdot A_2 + 2\cdot B_1 - 2\cdot A_1}{6}\right], \left[\frac{A_1 + 2\cdot B_1}{3}, \frac{A_2 + 2\cdot B_2}{3}\right], A;$$
$$\mathbf{end\ proc}:$$

$$KochPlot(0); KochPlot(1); KochPlot(2);$$







```
KochFunction := proc(A, B)
  return A, [(2·A1 + B1)/3, (2·A2 + B2)/3], [(3·B1 + 3·A1 + 2·A2 − 2·B2)/6,
    (3·B2 + 3·A2 + 2·B1 − 2·A1)/6], [(A1 + 2·B1)/3, (A2 + 2·B2)/3], B;
end proc:
```

e)

```
KochPlot := proc(n)
  local points, p1, p2, p3, p4, p5;
  points := KochPoints(KochFunction, n);
  points := map(x → [x[1], -1/(1 + x[2])], points);
  p1 := pointplot(points, connect = true);
  p2 := rotate(pointplot(points, connect = true), π);
  p3 := translate(p2, 1, -1);
```

```
    p4 := translate(rotate(pointplot(points, connect = true), π/2), 0, -1);
    p5 := translate(reflect(p4, [1/2, 0], [1, 0]), 0, -1);
    display(p1, p3, p4, p5, scaling = constrained, axes = none);
end proc:

KochPlot(0); KochPlot(1); KochPlot(2);
```