

Cheri Shakiban
 Professor of Mathematics
 University of St. Thomas
 St. Paul, Mn. 55433

Chapter 5. Geometric Symmetry

There has always been an interest in symmetry from both artistic and mathematical perspective. In fact the development of the underlying laws of symmetry traces back to the Pythagorean times. It has always played an important role in architectural and engineering designs and particularly in the decorative arts. By the end of the eighteenth century it was realized that it also plays an important role in chemistry and atomic physics, and has a dominant role in crystallography. There are many interesting books available on this subject. For instance see [Lockwood & Macmillan], [Hargittai] and [Stewart & Golubitsky].

Transformations of the points in the coordinate plane play an important role in the study of some geometric symmetry in computer graphics.

5.1. Transformations.

A *transformation* in the plane, \mathbf{R}^2 , is a mapping T of a set A onto another set B such that each element of B is the image of exactly one element of A , i.e. a one-to-one correspondence exists between the sets of elements of A and B .

Two of the earliest mathematicians to write about transformation theory were the English mathematician Arthur Cayley (1821-1895), and James Sylvester (1814-1897).

Translations.

The simplest types of transformations are translations. For instance the mapping T , which maps a point

$\begin{bmatrix} x \\ y \end{bmatrix}$ to $\begin{bmatrix} x+a \\ y+b \end{bmatrix}$, where a and b are real numbers, is an example of a translation.

The following program translates any figure f defined by lines to its image under the translation T , given above, and plots both figures. Here $\mathbf{x0}$ and $\mathbf{y0}$, stand for the initial position of the inputted image.

Program 5.1: Translation

```
Translation[f_,a_,b_,x0_,y0_] :=(
```

```

T[{x_,y_}]= {x+a,y+b};
locations={ {x0,y0},T[{x0,y0}]};

Show[
Graphics[
Line/@Map[f,locations]],
PlotRange->{{0,4},{0,4}},
Axes->True, AspectRatio->Automatic]
*****

```

If the figure, is a flag given by

```

flag[{x_,y_}] :=
{ {x,y}, {x,y+2}, {x+1,y+1}, {x,y+1} },

```

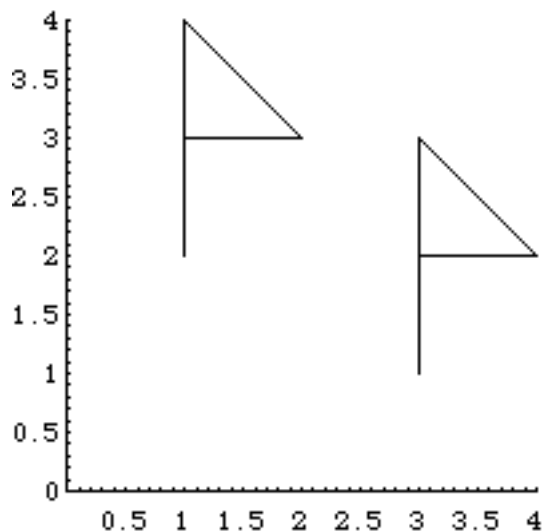
Then Translation can

to plot a flag at (1,2) with its image under the translation $T[{x_,y_}] = \{x+2,y-1\}$.

```

Translation[flag,2,-1,1,2]

```



If T is a transformation from A onto B and S is a transformation from B onto C , then $T \circ S$ given by $T \circ S(P) = T[S(P)]$ for each point of A , is also a transformation. In particular if T and S are translations so is $T \circ S$. (Show this!)

If T is a translation, we can form a list of translations $T, T^2 = T \circ T, T^3, \dots, T^n$. This list when applied to a figure produces n images of that figure.

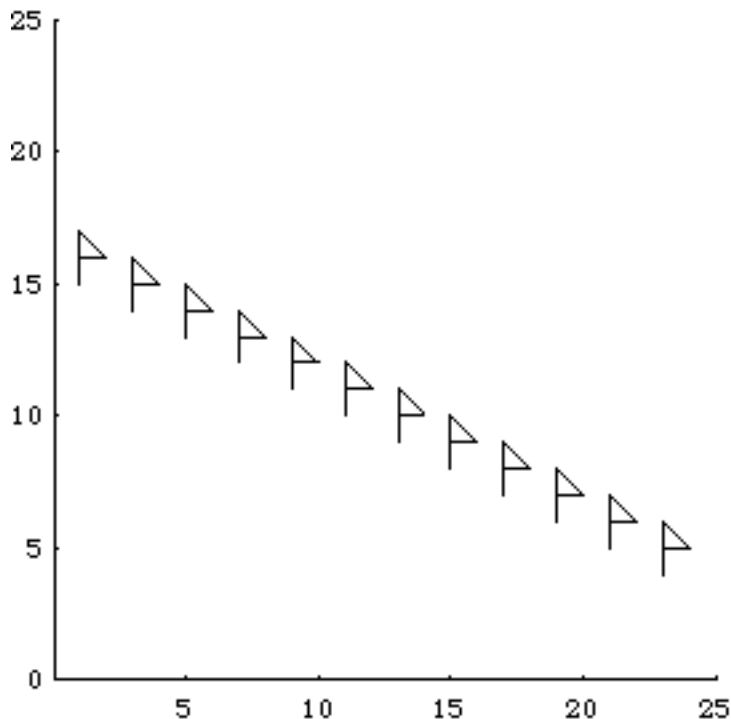
A simple **Translation** modification using the command **NestList** can draw n images of a figure with respect to a given translation $T[\{x_, y_-\}] = \{x+a, y+b\}$ with the initial position at (x_0, y_0) .

Program 5.2: TranslationSeveral

```
TranslationSeveral[f_, a_, b_, x0_, y0_, n_] := (
  T[{x_, y_}] = {x+a, y+b};
  locations = NestList[T, {x0, y0}, n-1];
```

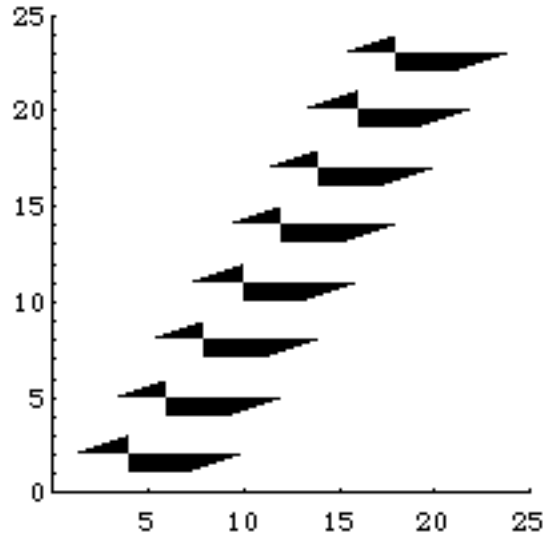
```
  Show[
    Graphics[
      Line/@Map[f, locations]],
      PlotRange -> {{0, 25}, {0, 25}},
      Axes -> True, AspectRatio -> Automatic]
```

```
TranslationSeveral[flag, 2, -1, 1, 15, 12]
```



If we use the `TranslationSeveral` command in `Table` and use the following function as our input for f , we get something completely different

```
dock[{x_,y_}]=
  Table[{x+3 Mod[n,5],y+Floor[n/3]},{n,6}];
TranslationSeveral[dock,2,3,1,1,8]
```



The concept of transformation is important because it allows us to investigate a close relationship between abstract algebra and modern geometry.

In geometry we are often interested in a set of transformations rather than just one single transformation. Such sets of transformations usually form a *group* which is the fundamental topic in abstract algebra.

Definition. A *group of transformations* of a set A onto itself is a nonempty set of transformations such that:

1. If T and S are in S , then $T \circ S$ is in S . (Closure)
2. If T and S and R are in S , then $(T \circ S) \circ R = T \circ (S \circ R)$. (Associativity)
3. $I \in S$ satisfying $I \circ T = T \circ I = T$, for all T in S . (Identity)
4. Given T in S , there exists a unique element T^{-1} satisfying $T \circ T^{-1} = T^{-1} \circ T = I$. (Inverse)

For S , the set of all translations of the form given by T

Error!

Many of the groups of transformations in geometry are infinite groups such as the group of translations discussed above. However, many examples of symmetries are based on finite groups of transformations.

The simplest of the transformations used in symmetries are linear transformations.

5.2. Linear Transformations.

A linear transformation in the plane, \mathbf{R}^2 , is a transformation L , which maps a point

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$$

in the plane to another point $L(\mathbf{x})$ in the plane such that the following properties hold:

- (a) $L(\mathbf{x} + \mathbf{y}) = L(\mathbf{x}) + L(\mathbf{y})$, for every \mathbf{x} and \mathbf{y} in \mathbf{R}^2 .
- (b) $L(c \mathbf{x}) = c L(\mathbf{x})$, for every \mathbf{x} in \mathbf{R}^2 and every scalar c .

For example reflection with respect to the x -axis given by L

$$\begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} x \\ -y \end{bmatrix}$$

is a linear transformation. (Check this!)

The easiest way of representing linear transformations L , is by using matrix representation.

Theorem 5.1. If L is a linear transformation from \mathbf{R}^2 to \mathbf{R}^2 , then there is a matrix of the form $A =$

Error!

The proof of this theorem is a constructive one and is left as an exercise.

Example. The reflection with respect to the x -axis is given by $L \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$

What is the matrix of the linear transformation for the reflection with respect to the y -axis?

Rotation.

The second example of a linear transformation is an anticlock-wise rotation about the origin through the angle θ . It is easy to see from Fig. 1., that

$$x' = \cos(\theta) r - \sin(\theta) y = \cos(\theta) x - \sin(\theta) y$$

$$y' = \sin(\theta) r + \cos(\theta) x = \sin(\theta) x + \cos(\theta) y$$

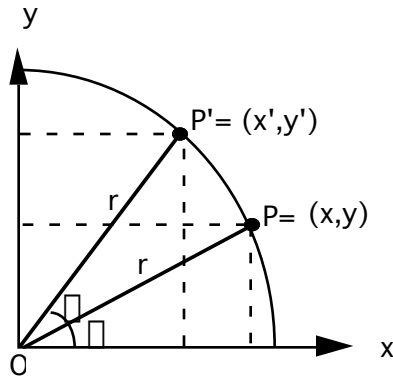


Fig. 1

Therefore the linear transformation of the rotation about the origin through the angle θ is given by

$$L \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

What is the matrix of linear transformation of the clock-wise rotation about the origin?

Let us investigate how this theory applies to graphics with an example.

Program 5.3: Airplane

(* the parametric equation of a curve in the shape of an airplane)

```
r[t_] := Sin[7t] + Cos[4t] + 3;
```

```
Airplane[t_] := { r[t] Cos[t], r[t] Sin[t]};
```

(* rotating the airplane through an angle s*)

```
rotate[t_, s_] :=
```

```
{ {Cos[s], Sin[s]}, {-Sin[s], Cos[s]} }.Airplane[t];
```

(* plotting a sequence of images*)

```
motion[s_] :=
```

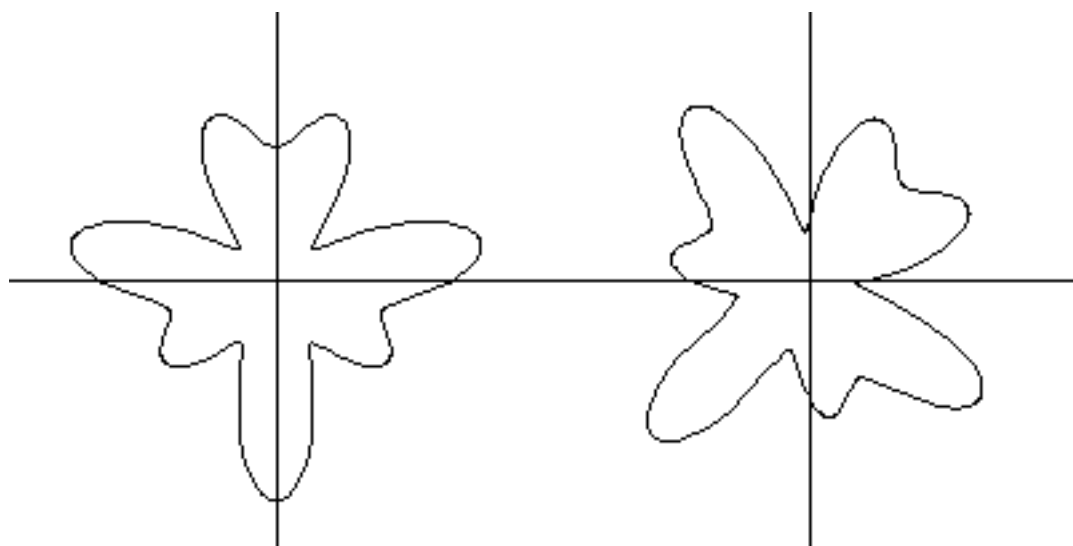
```
ParametricPlot[rotate[t, s], {t, 0, 2Pi},
```

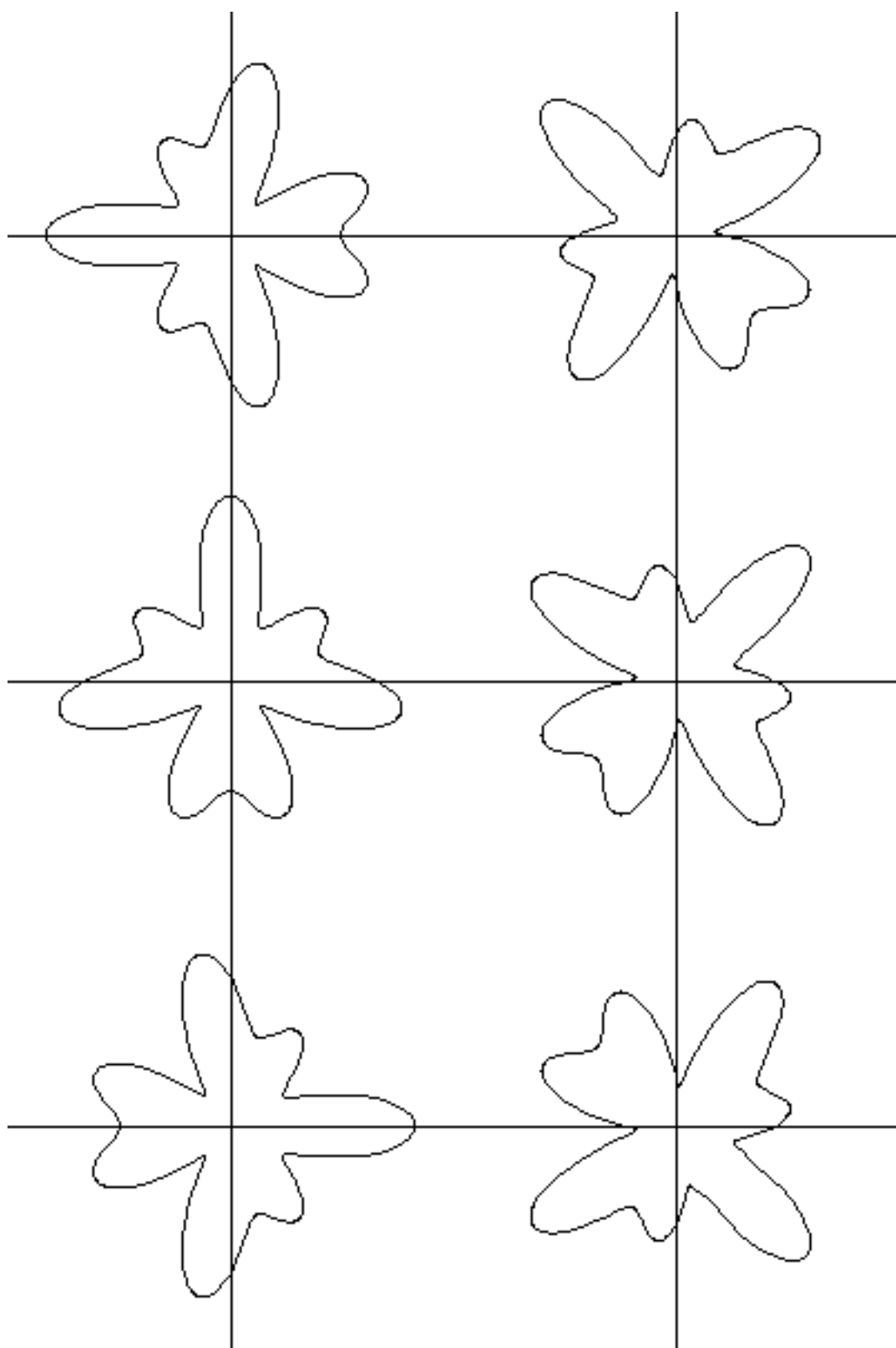
```
AspectRatio->Automatic, Ticks->None,
```

```
PlotRange->{{-6, 6}, {-6, 6}}]
```

```
motion/@ Range[0, 2 Pi, 7 Pi/4]
```

The output of the program which is displayed below when animated produces the effect of an airplane turning around itself.





Reflection.

Composition of two linear transformations is also another linear transformation. In fact if L_A

Error!

In order to reflect a point P in the line through the origin at angle θ to a point P' , (Fig. 2.), we can compose the linear transformation for a clockwise rotation of θ about the origin with a reflection in the x -axis and finally with an anticlockwise rotation of θ about the origin.

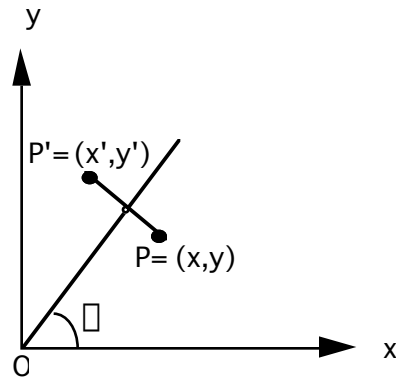


Fig.2.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

This matrix of reflection can be simplified using the matrix multiplication routine available in *Mathematica*.

In[1]:

```
M1={Cos[t],-Sin[t]},{Sin[t], Cos[t]};
M2={1,0},{0,-1};
M3 ={{Cos[t],Sin[t]},{-Sin[t],Cos[t]};
MatrixForm[M1.M2.M3]
```

Out[1]:

$$\begin{aligned} \cos^2 t - \sin^2 t &= \cos 2t \\ 2 \cos t \sin t &= \sin 2t \end{aligned}$$

Using the identities $\cos 2\theta = \cos^2 \theta - \sin^2 \theta$, and $\sin 2\theta = 2 \cos \theta \sin \theta$, we get

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos 2\theta & \sin 2\theta \\ \sin 2\theta & -\cos 2\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

The following program reflects an image given in the shape of an spiral through a line going through the origin with angle s .

Program 5.4: Reflect

(* the parametric equation of a curve in the shape of an spiral*)

S[t_]:= {t/20 Cos[t]+3, t/20 Sin[t]+3 Tan[Pi/8]};

(* the matrix of reflection through a line with angle s*)

mat[s_]:= {{Cos[2s], Sin[2s]}, {Sin[2s], -Cos[2s]}};

(* the parametric equation of the image of the spiral through the line $y=x$ *)

s= Pi/4; R[t_]:= mat[s].S[t];

(* the graphics routine*)

**graph[f_]:=ParametricPlot[f[t], {t,0,6Pi},
AspectRatio->Automatic, Ticks->None,
DisplayFunction->Identity,
PlotRange->{{0,6},{0,6}}];**

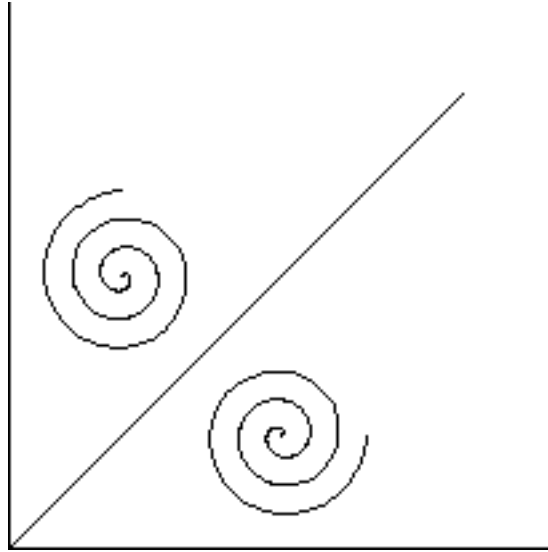
(* the line through the origin with angle s*)

**mirror = Plot[Tan[s] x, {x, 0, 5},
AspectRatio->Automatic, DisplayFunction->Identity];**

(* displaying the graphs of S and R*)

Show[{graph/@{S,R}, mirror},

DisplayFunction->\$DisplayFunction]



The following program consists of a sequence of consecutive reflections of the spiral given in the program **Reflect**.

Program 5.5: ReflectSeveral

```
S[t_]:= {t/20 Cos[t]+3, t/20 Sin[t]+ 3 Tan[Pi/8]};
```

```
mat[s_]:= {{Cos[2s], Sin[2s]}, {Sin[2s], -Cos[2s]}};
```

```
gg={};
```

```
Do[
```

```
h[s_]:=ParametricPlot[mat[s].S[t], {t, 0, 6Pi},
```

```
    AspectRatio->Automatic, Ticks->None,
```

```
    DisplayFunction->Identity,
```

```
    PlotRange->{{-6, 6}, {-6, 6}}];
```

```
AppendTo[gg, h[s]];
```

```
S[t_]= mat[s].S[t], {s, Pi/4, 2 Pi, Pi/4}];
```

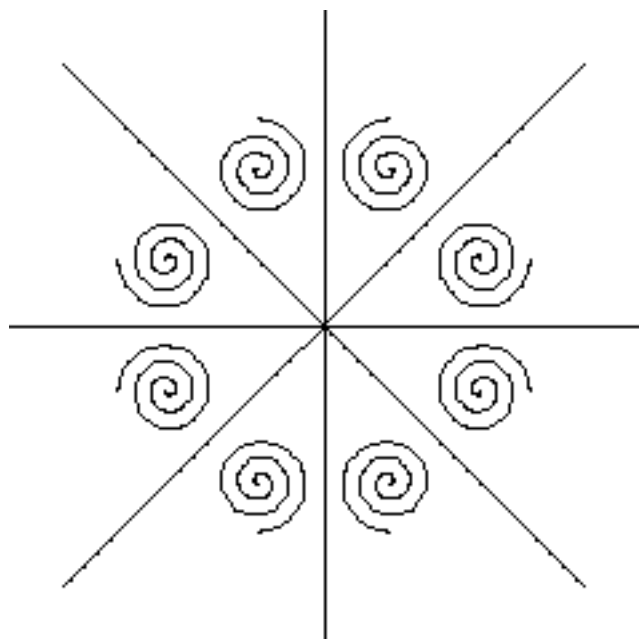
```
Mirror[j_]:=Plot[ j x, {x, -5, 5},
```

```

        AspectRatio->Automatic,
        DisplayFunction->Identity];

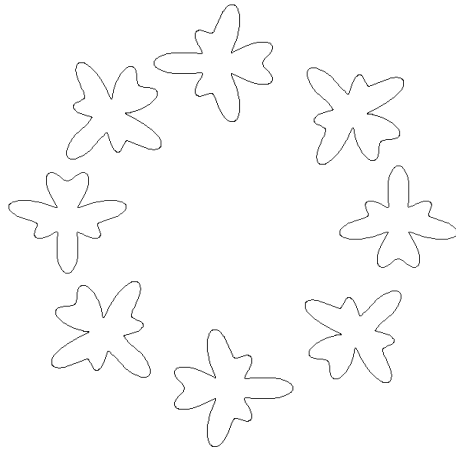
Show[{gg, Mirror/@{-1,1}},
      DisplayFunction->$DisplayFunction]
*****

```

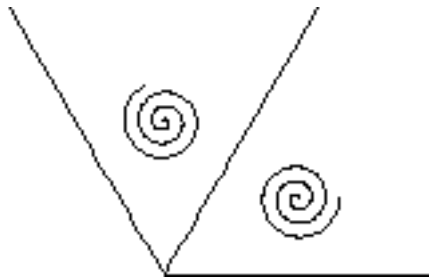


Exercise Set 5.1

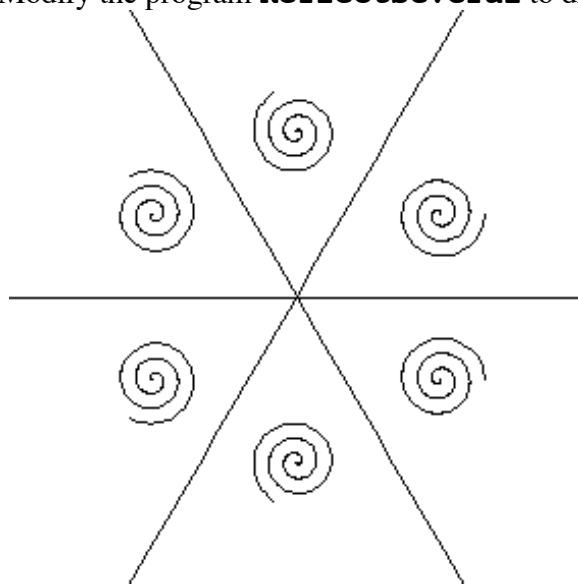
- 1 . **Translation**
to reflect a flag with respect to the x-axis and draw both flags.
2. Modify the program **TranslationSeveral** to draw several horizontal flags.
3. Create any image and use any translation to draw several of them both with using the command **Line** and **Polygon** in the program **TranslationSeveral**.
4. Modify the program **Airplane** to rotate an airplane around a circle.



5. Modify the program **Reflect** to draw an spiral and its image reflected through the line which makes an angle of $\pi/3$ with the x-axis.



6. Modify the program **Reflect** to create different images and reflect them through the line $y=x$.
7. Modify the program **ReflectSeveral** to draw the following figure.



8. Modify the program **ReflectSeveral** to draw an spiral and its image rotated through the lines which intersect at the origin with angle $\frac{2\pi}{n}$. Test your program for $n = 4$.

5.3. Point Symmetry.

A transformation that changes a figure into a congruent figure is called an *isometry*. Rotation is a *direct isometry* as each part of the congruent figure is an exact reproduction of the original figure, though in a different place. Reflection is an *indirect isometry*. (Why?)

By *symmetry* we mean not only that the different parts of a figure are congruent but also related by an isometry, e.g. reflection or rotation, in such a way that the whole figure is self-coincident under that isometry, i.e. when we apply an isometry to this figure, it changes every part of the figure into another part, leaving the figure as a whole unchanged.

First we consider rotations.

Consider Fig. 3., generated by the following program **symmetric**.

In this program, we will draw circles at different positions on a curve given by the polar equation $r[t] = b \cdot t$, which is the equation of a spiral and then rotate the spiral through an angle of

Error!

Program 5.6: symmetric.

```
symmetric[a_,b_,n_,m_] := (
  r[t_] := b t;
  (* rotation applied to a the polar function*)
  mat[s_] :=
    {{Cos[s], Sin[s]}, {-Sin[s], Cos[s]}};
  vp[t_, s_] := mat[s].{r[t] Cos[t], r[t] Sin[t]};

  (* drawing circles*)
  cir[s_, t_] := Graphics[
    Circle[vp[t, s], a],
    AspectRatio->Automatic];
```

```
(* drawing the circles at the polar curve*)
f[s_]:=cir[s,#]& /@ Range[0, Pi, Pi/m//N];
Show[{f /@ Range[0, 2 Pi, 2 Pi/n//N]})
*****

symmetric[1,0.2,4,50]
```

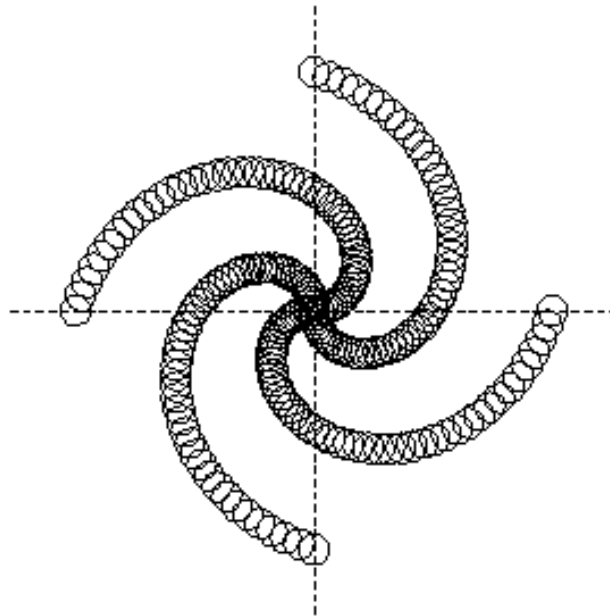


Fig. 3.

Looking at the symmetrical pattern in Fig. 3, we notice that the only transformations that leave this figure unchanged are rotations. In fact the minimum angle for a rotation transformation that leaves this figure unchanged is 90° and the figure is also unchanged through angles 180° , 270° , and 360° (which brings back the figure to the original position). In this case we can see that the center point of this pattern remains fixed. Thus the symmetry of such patterns is called *point symmetry*.

In general in the patterns with just rotational symmetry, the minimum angle of rotation is

Error!

If we denote the transformations corresponding to the rotation of d degrees, by r_d . We can see that the set of rotations r_0 , r_{90} , r_{180} , and r_{270} , form a group, C_4 . The following table shows how these rotations combine as elements of the group C_4 . Note that r_0 , the rotation through angle 0, which is also equal to r_{360} , acts just as the identity of this group.

	r_0	r_{90}	r_{180}	r_{270}
r_0	r_0	r_{90}	r_{180}	r_{270}
r_{90}	r_{90}	r_{180}	r_{270}	r_0
r_{180}	r_{180}	r_{270}	r_0	r_{90}
r_{270}	r_{270}	r_0	r_{90}	r_{180}

Group C_4

The following commands generate the images of Fig 4.

`symmetric[1,3,2,30]`

`symmetric[2,3,3,30]`

`symmetric[1,2,5,30]`

`symmetric[1,0.3,7,50]`

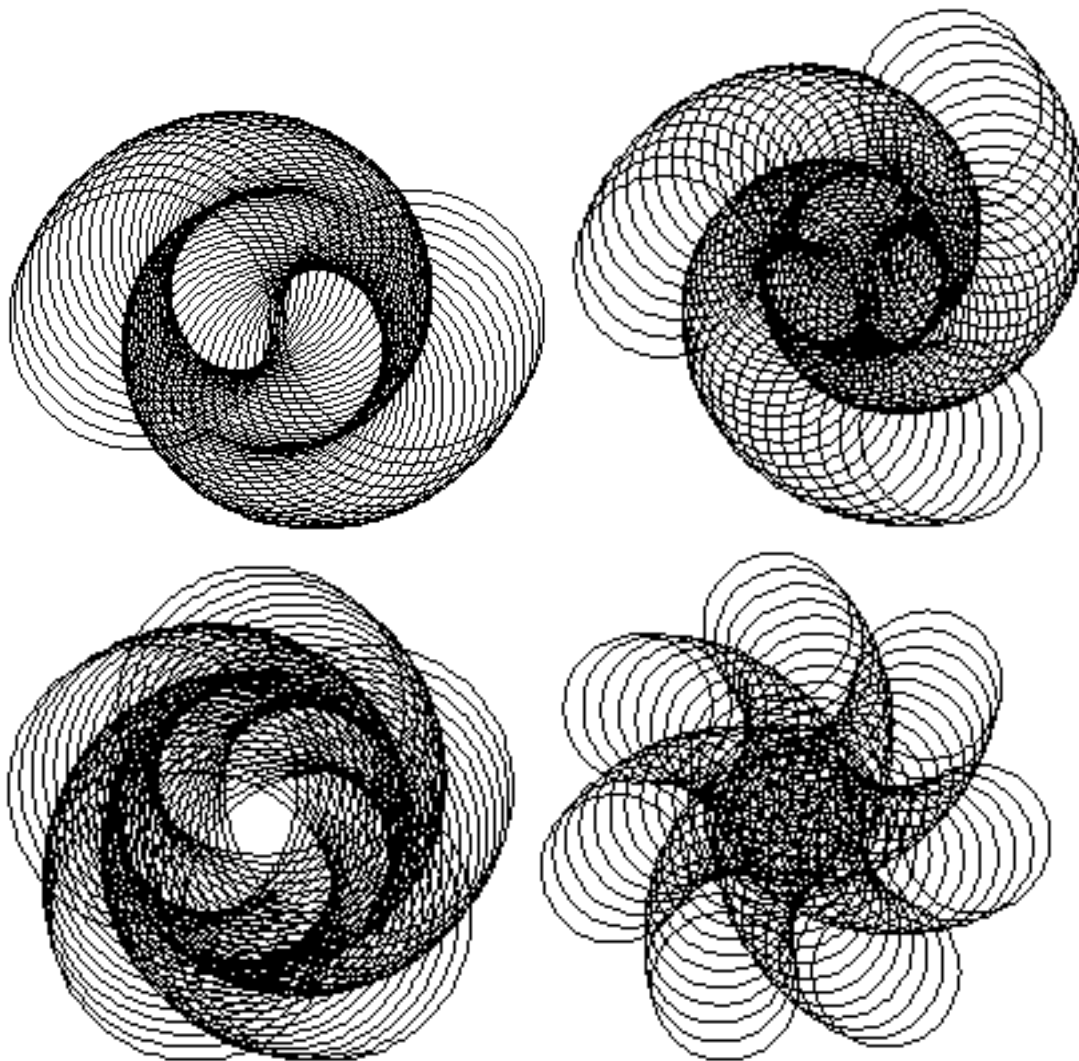


Fig. 4.

Next we consider groups which have reflections as their symmetries.

Consider Fig. 5., generated by the following simple routine in *Mathematica*.

Program 5.7: Butterfly.

```
*****
Butterfly[a_, b_, c_] := (
ParametricPlot[
  {Sin[a t] Cos[ b t], Sin[a t] Sin[ c t]},
  {t, 0, 2 Pi}, AspectRatio->Automatic, Axes-> None]
*****
```

Butterfly [23, 13, 3]

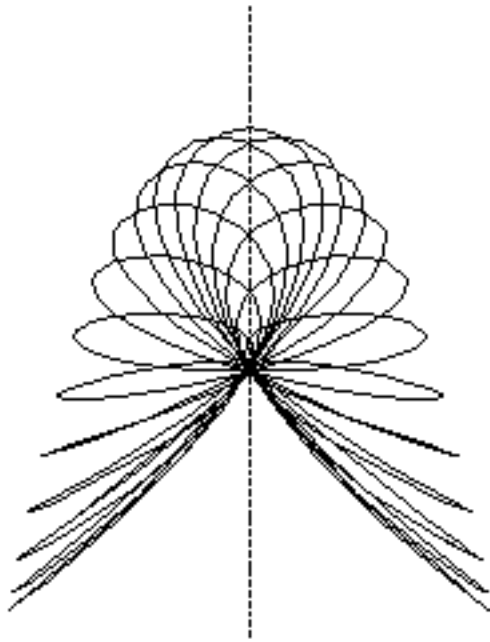


Fig. 5.

As we can see there is a single reflection line for this symmetrical pattern. This is perhaps the simplest of all symmetry types. If, however, there is more than one, reflection line, there must also be a rotation center. For instance consider the following image generated by **Butterfly**[12,5,7].

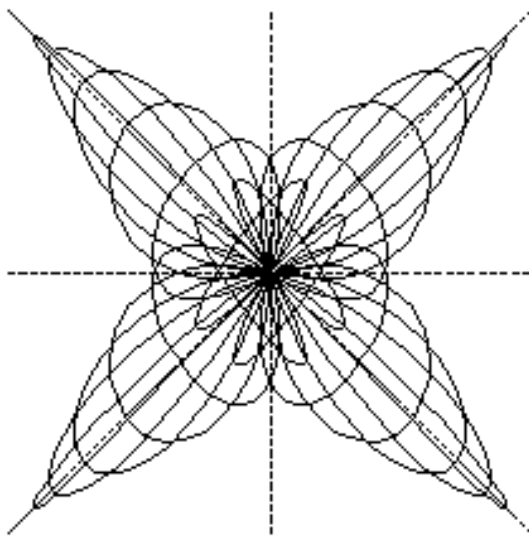


Fig. 6.

The pattern given in Fig. 6. admits reflections in four lines intersecting at the center of the pattern making angles 45° . It also admits rotation about the center through the angles 90° , 180° , 270° , and 360° .

In general patterns with reflection lines intersecting at the center of the pattern at an angle θ , and with rotation symmetries about the center through the angle 2θ , are called *dihedral groups* and are denoted by D_n , where n is the value given in the angle $\theta =$

Error!

If we denote the transformations corresponding to the rotation of d degrees, by r_d , and the reflections by H (horizontal reflection line), V (vertical reflection line), D (diagonal reflection line with angle $\pi/4$) and D' (diagonal reflection line with the angle $3\pi/4$), We can see that the set of rotations r_0, r_{90}, r_{180} , and r_{270} , together with the reflections H, V, D and D' form a group, D_4 . The following table shows how these rotations and reflections combine as elements of the group D_4 . Note that r_0 also acts as the identity of this group.

		First movement							
		r_0	r_{90}	r_{180}	r_{270}	H	V	D	D'
Second movement	r_0	r_0	r_{90}	r_{180}	r_{270}	H	V	D	D'
	r_{90}	r_{90}	r_{180}	r_{270}	r_0	D'	D	H	V
	r_{180}	r_{180}	r_{270}	r_0	r_{90}	V	H	D'	D
	r_{270}	r_{270}	r_0	r_{90}	r_{180}	D	D'	V	H
	H	H	D	V	D'	r_0	r_{180}	r_{90}	r_{270}
	V	V	D'	H	D	r_{180}	r_0	r_{270}	r_{90}
	D	D	V	D'	H	r_{270}	r_{90}	r_0	r_{180}
	D'	D'	H	D	V	r_{90}	r_{270}	r_{180}	r_0
Group D_4									

If we symmetrize, by
`r[t_]:=Sin[a`
`t]`, We can produce a number of examples for the dihedral group.

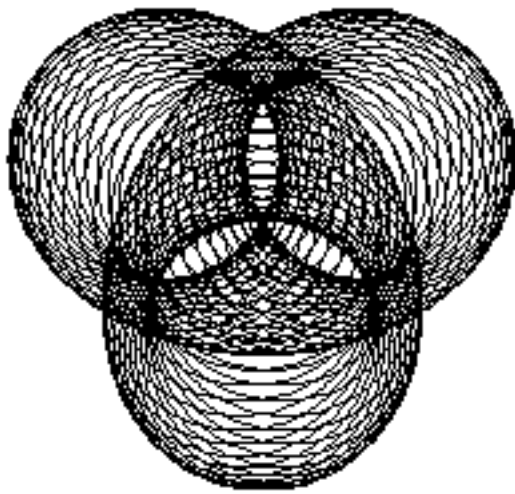
The following commands generate the images of Fig 7.

`symmetric[3,1,3,35]`

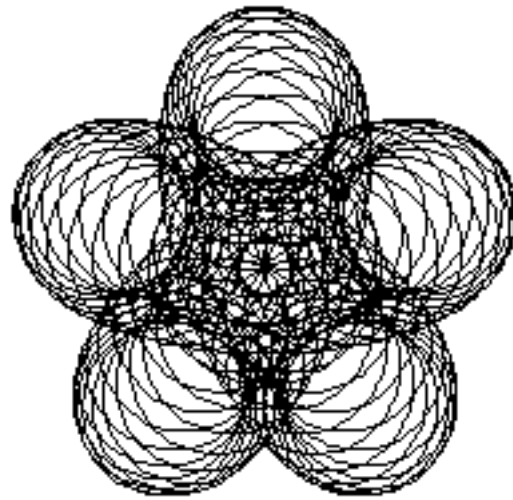
```

symmetric[5,0.5,5,90]
symmetric[6,0.05,6,70]
symmetric[1,.2,6,50]

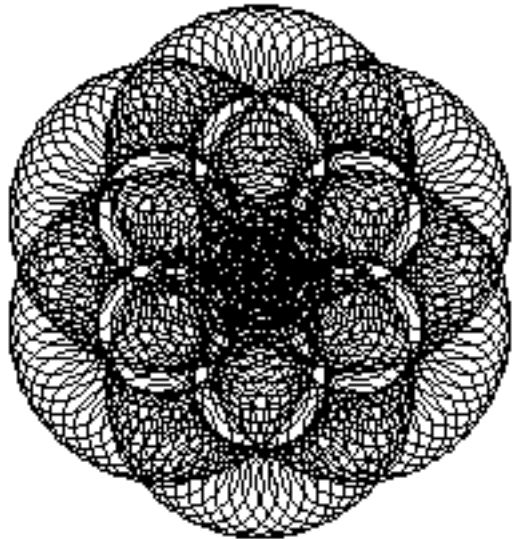
```



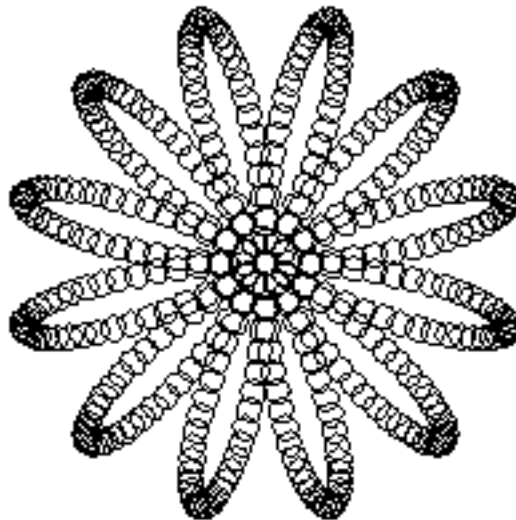
D_3



D_5



D_6



D_{12}

Fig. 7.

As our **Butterfly** algorithm example by connecting lines from the origin to the curve and graphing the curve for two different sets of values.

Program 5.8: Moiré Butterfly.

```

*****
MoireButterfly[a_, b_, c_, n_] := (
    r[t_] := {Sin[a t] Cos[ b t], Sin[a t] Sin[ c t]};

    (* drawing the lines to the polar curve*)
    values = r/@Range[0, 2 Pi, 2 Pi/n//N ];
    Show[Graphics[{Thickness[.0005],
        Map[Line[{0,0}, #]&, values],

    (* drawing the two polar curves*)
    Table[Line[values rad^0.2], {rad, 0.5, 1, 0.5}]],
    AspectRatio->1])
*****

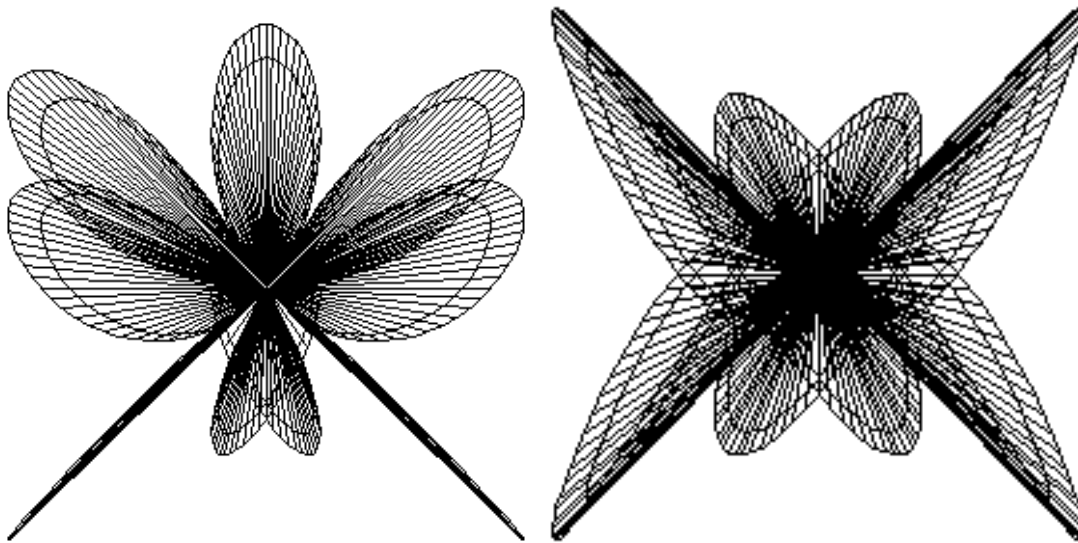
```

The following commands produce the images of the Fig. 8.

```

MoireButterfly[9,3,5,720]
MoireButterfly[2,3,5,360]
MoireButterfly[8,3,3,360]

```



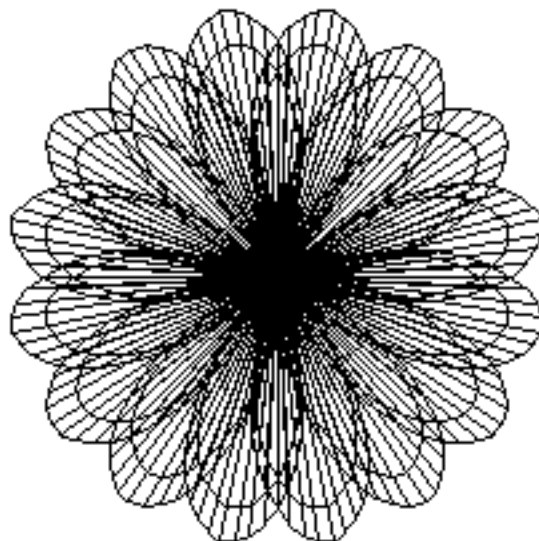


Fig. 8.

Exercise Set 5.2

1. Generate the images given by

MoireButterfly[7,4,3,360]

MoireButterfly[19,9,9,360]

MoireButterfly[17,9,5,360]

MoireButterfly[7,5,2,360]

MoireButterfly[5,2,2,360]

Find the condition on a , b , and c that determines when the image is a butterfly type image and when the image is a flower type image.

Identify the symmetry group of each image.

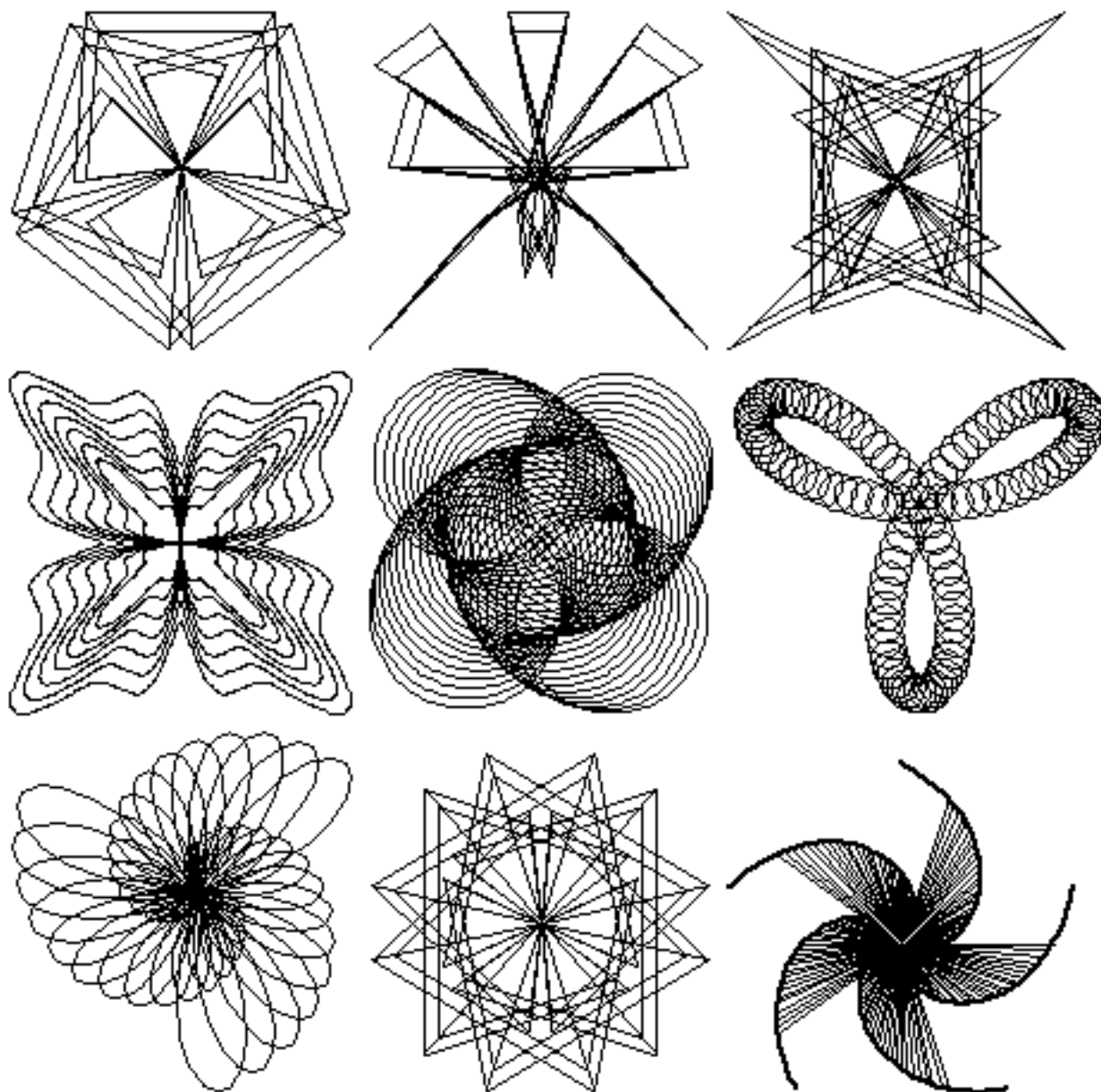
2. Modify the program **MoiréButterfly**, by defining $\mathbf{r}[\mathbf{t_}]$ to be $\mathbf{r}[\mathbf{t_}] :=$

$(2 + \sin[a \cdot t]/2) \cdot \{\cos[t + \sin[b \cdot t]/c], \sin[t + \sin[b \cdot t]/c]\}.$

Generate images for the values $a=8$, $b=16$, $c=4$, and $a=9$, $b=6$, $c=6$, and $a=6$, $b=18$, $c=18$.

Identify the symmetry group of each image.

3. Identify the symmetry group of the images given below.



4. Write a program to produce your own symmetric images for finite patterns in two dimensions, both the cyclic types and the dihedral types.

5.3. Point Symmetry.

A transformation that changes a figure into a congruent figure is called an *isometry*. Rotation is a *direct isometry* as each part of the congruent figure is an exact reproduction of the original figure, though in a different place. Reflection is an *indirect isometry*. (Why?)

By *symmetry* we mean not only that the different parts of a figure are congruent but also related by an isometry, e.g. reflection or rotation, in such a way that the whole figure is

self-coincident under that isometry, i.e. when we apply an isometry to this figure, it changes every part of the figure into another part, leaving the figure as a whole unchanged.

First we consider rotations.

Consider Fig. 3., generated by the following program **symmetric**.

In this program, we will draw circles at different positions on a curve given by the polar equation $r = b e^{at}$, which is the equation of a spiral and then $\frac{2\pi}{n}$ rotate the spiral and repeat the process. Here, a stands for the radius of the circles, n stands for the value in the angle $\frac{2\pi}{n}$, and m is the number of circles desired.

Program 5.6: symmetric.

```
*****
symmetric[a_,b_,n_,m_] := (
  r[t_] := b t;
  (* rotation applied to a the polar function*)
  mat[s_] :=
    {{Cos[s], Sin[s]}, {-Sin[s], Cos[s]}};
  vp[t_, s_] := mat[s].{r[t] Cos[t], r[t] Sin[t]};

  (* drawing circles*)
  cir[s_, t_] := Graphics[
    Circle[vp[t, s], a],
    AspectRatio -> Automatic];

  (* drawing the circles at the polar curve*)
  f[s_] := cir[s, #] & /@ Range[0, Pi, Pi/m/N];
  Show[{f /@ Range[0, 2 Pi, 2 Pi/n/N]})
*****

symmetric[1,0.2,4,50]
```

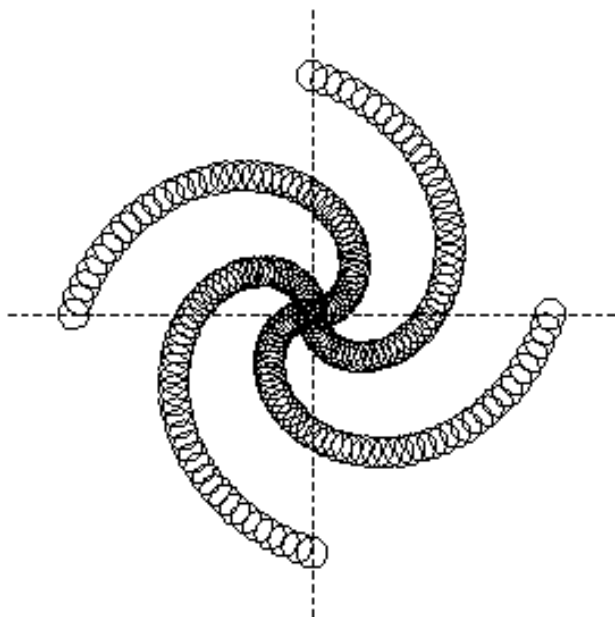



Fig. 3.

Looking at the symmetrical pattern in Fig. 3, we notice that the only transformations that leave this figure unchanged are rotations. In fact the minimum angle for a rotation transformation that leaves this figure unchanged is 90° and the figure is also unchanged through angles 180° , 270° , and 360° (which brings back the figure to the original position). In this case we can see that the center point of this pattern remains fixed. Thus the symmetry of such patterns is called *point symmetry*.

In general in the patterns with just rotational symmetry, the minimum angle of rotation is $\frac{360^\circ}{n}$ where n is an integer.

Patterns of this sort are said to have *cyclic symmetry*, and there are an infinite number of them, according to the different value of n . Fig. 4, shows examples in which $n = 2, 3, 5$ and 7 . These types of symmetries are denoted by C_n , or Z_n .

If we denote the transformations corresponding to the rotation of d degrees, by r_d , we can see that the set of rotations r_0, r_{90}, r_{180} , and r_{270} , form a group, C_4 . The following table shows how these rotations combine as elements of the group C_4 . Note that r_0 , the rotation through angle 0 , which is also equal to r_{360} , acts just as the identity of this group.

	r_0	r_{90}	r_{180}	r_{270}
r_0	r_0	r_{90}	r_{180}	r_{270}
r_{90}	r_{90}	r_{180}	r_{270}	r_0
r_{180}	r_{180}	r_{270}	r_0	r_{90}
r_{270}	r_{270}	r_0	r_{90}	r_{180}

Group C_4

The following commands generate the images of Fig 4.

`symmetric[1,3,2,30]`

`symmetric[2,3,3,30]`

`symmetric[1,2,5,30]`

`symmetric[1,0.3,7,50]`

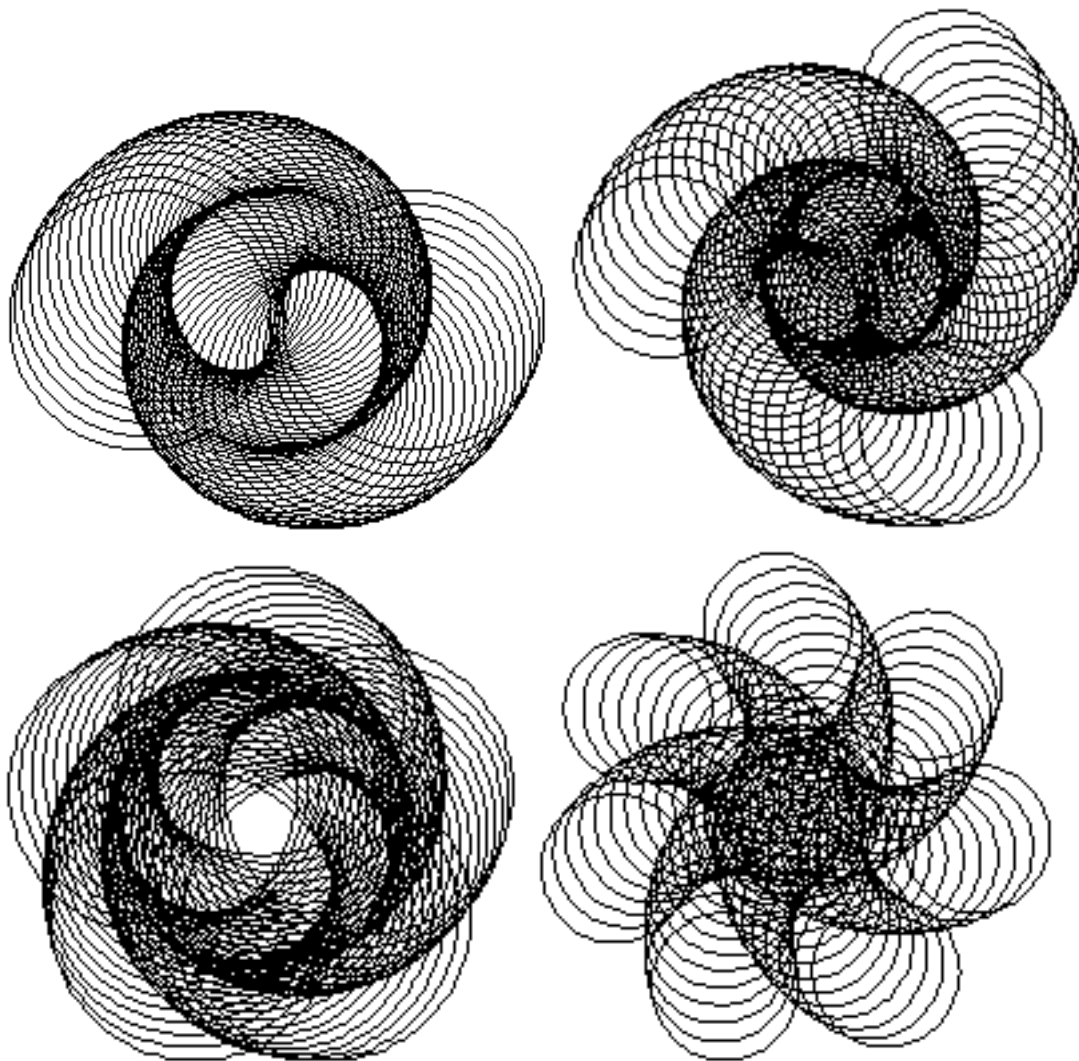


Fig. 4.

Next we consider groups which have reflections as their symmetries.

Consider Fig. 5., generated by the following simple routine in *Mathematica*.

Program 5.7: Butterfly.

```
*****
Butterfly[a_, b_, c_] := (
ParametricPlot[
  {Sin[a t] Cos[ b t], Sin[a t] Sin[ c t]},
  {t, 0, 2 Pi}, AspectRatio->Automatic, Axes-> None]
*****
```

Butterfly [23, 13, 3]

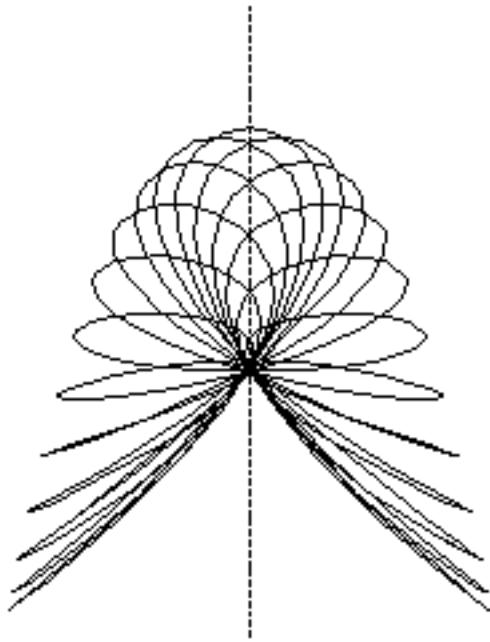


Fig. 5.

As we can see there is a single reflection line for this symmetrical pattern. This is perhaps the simplest of all symmetry types. If, however, there is more than one, reflection line, there must also be a rotation center. For instance consider the following image generated by **Butterfly**[12,5,7].

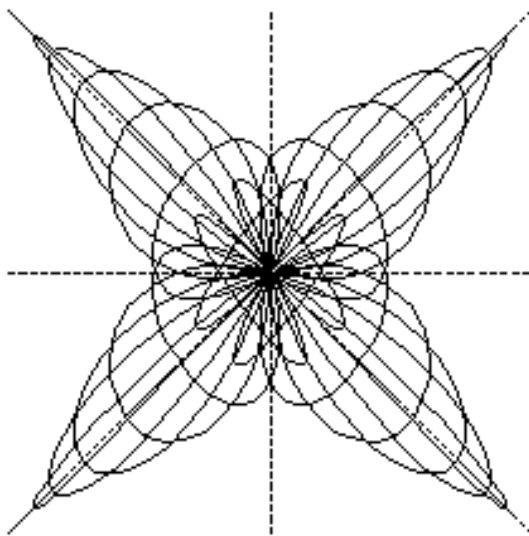


Fig. 6.

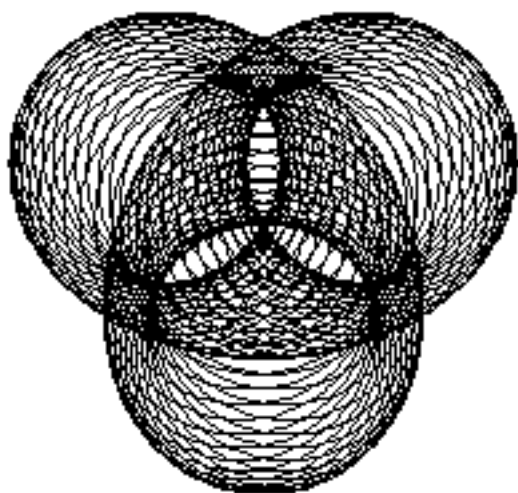
The following commands generate the images of Fig 7.

`symmetric[3,1,3,35]`

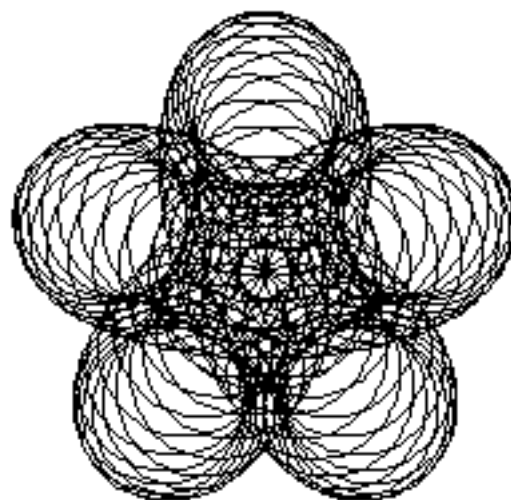
`symmetric[5,0.5,5,90]`

`symmetric[6,0.05,6,70]`

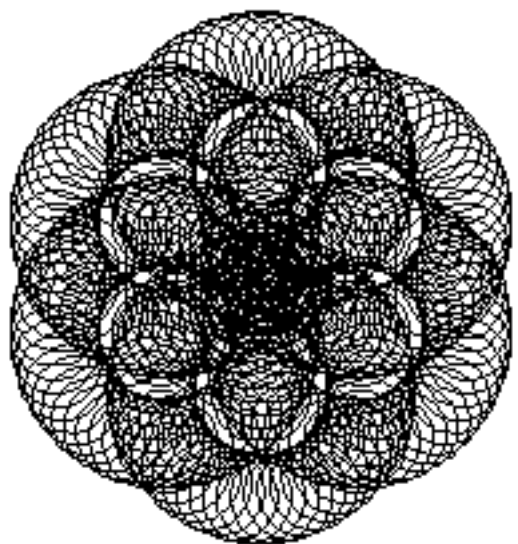
`symmetric[1,.2,6,50]`



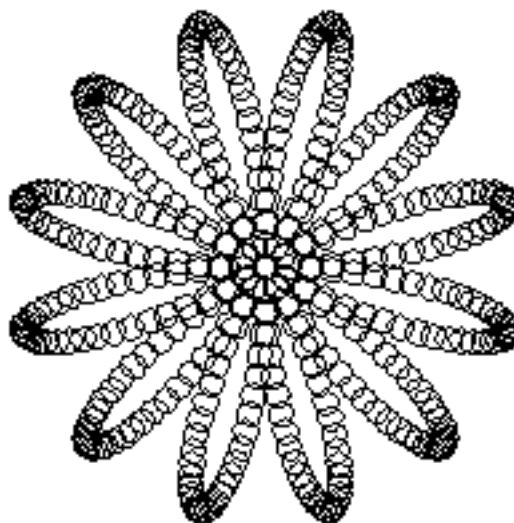
D_3



D_5



D_6



D_{12}

Fig. 7.

As our **Butterfly** algorithm example by connecting lines from the origin to the curve and graphing the curve for two different sets of values.

Program 5.8: Moiré Butterfly.

```
MoireButterfly[a_, b_, c_, n_] := (
    r[t_] := {Sin[a t] Cos[ b t], Sin[a t] Sin[ c t]};
```

(* drawing the lines to the polar curve*)

```
values = r/@Range[0, 2 Pi, 2 Pi/n//N ];
    Show[Graphics[{Thickness[.0005],
        Map[Line[{0,0}, #]&, values],
```

(* drawing the two polar curves*)

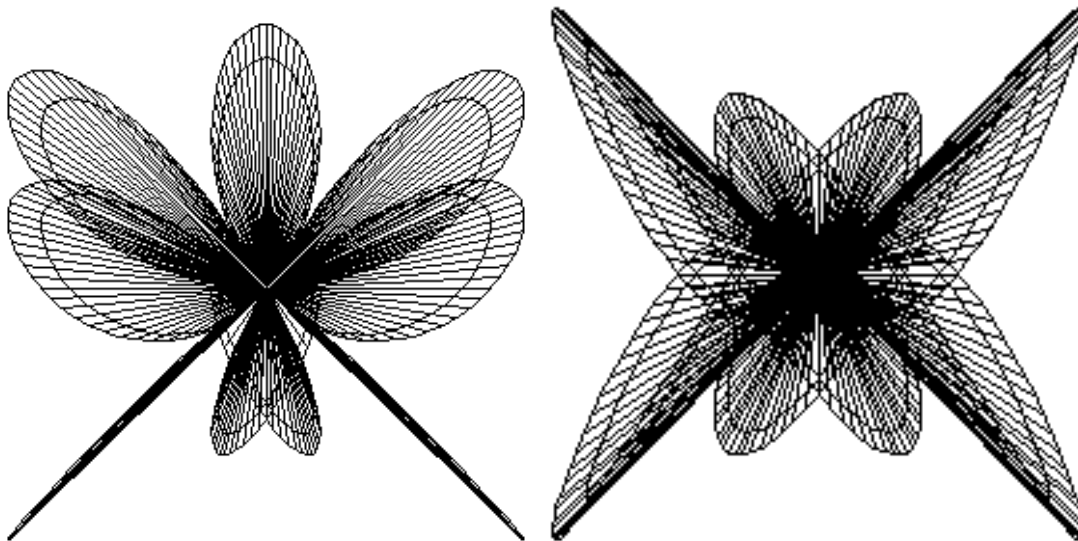
```
Table[Line[values rad^0.2], {rad, 0.5, 1, 0.5}]],
    AspectRatio->1])
```

The following commands produce the images of the Fig. 8.

```
MoireButterfly[9,3,5,720]
```

```
MoireButterfly[2,3,5,360]
```

```
MoireButterfly[8,3,3,360]
```



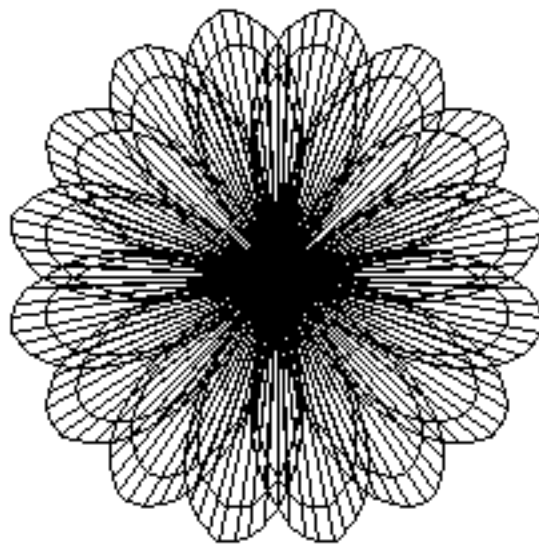


Fig. 8.

Exercise Set 5.2

1. Generate the images given by

MoireButterfly[7,4,3,360]

MoireButterfly[19,9,9,360]

MoireButterfly[17,9,5,360]

MoireButterfly[7,5,2,360]

MoireButterfly[5,2,2,360]

Find the condition on a , b , and c that determines when the image is a butterfly type image and when the image is a flower type image.

Identify the symmetry group of each image.

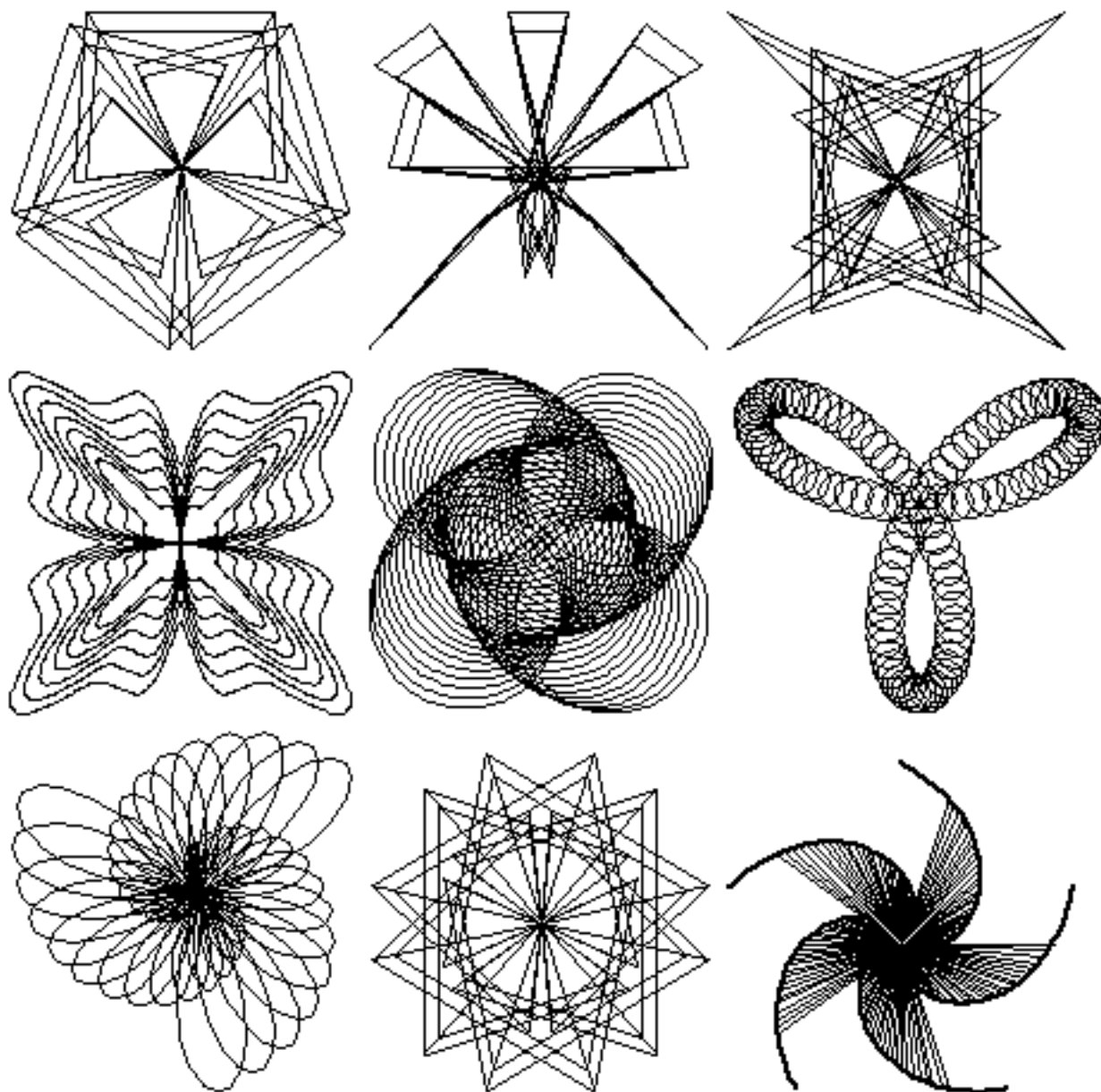
2. Modify the program **MoiréButterfly**, by defining $\mathbf{r}[\mathbf{t_}]$ to be $\mathbf{r}[\mathbf{t_}] :=$

$(2 + \sin[a \cdot t]/2) \cdot \{\cos[t + \sin[b \cdot t]/c], \sin[t + \sin[b \cdot t]/c]\}$.

Generate images for the values $a=8$, $b=16$, $c=4$, and $a=9$, $b=6$, $c=6$, and $a=6$, $b=18$, $c=18$.

Identify the symmetry group of each image.

3. Identify the symmetry group of the images given below.



4. Write a program to produce your own symmetric images for finite patterns in two dimensions, both the cyclic types and the dihedral types.

5.4. Frieze Patterns.

A *frieze pattern*, consists of a *motif*, repeated at regular intervals along a line. In practice, frieze patterns are not infinite in length, but in theory since we can compose a translation on with itself any number of times, it is convenient to suppose that patterns are extended indefinitely. Therefore, when a translation is applied to the pattern, it would leave the pattern unchanged as a whole.



Fig. 9

In addition to translations, for some frieze patterns rotations, also leave the pattern unchanged. For a frieze pattern to have rotation or reflection symmetry each motif in the pattern must have that symmetry and so must the row of the motifs. In the Fig. 10., each motif has symmetry group D_8 , but the frieze as a whole would not remain unchanged if turned through the angle 45° , so we cannot say that this frieze pattern has D_8 as its symmetry group. It does however have symmetry group D_2 .



Fig. 10

In fact the frieze patterns can have only, rotation symmetry through the angle 180° , i.e., half turns. Note that the rotation through the angle 360° , brings the pattern back to the original position. They can also only have horizontal or vertical reflection symmetry. There is furthermore a transformation which leaves certain types of frieze patterns as a whole unchanged when applied to them and that is *glide reflection*. In glide reflection, the motif is first reflected with respect to a horizontal line and then translated.

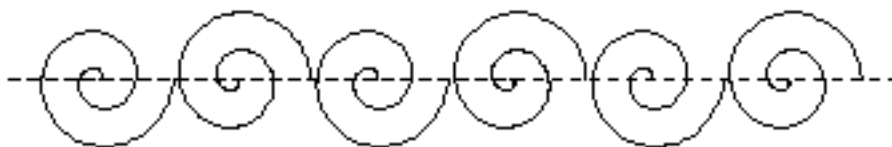


Fig. 11

The group associated with a frieze pattern is referred to as a *line group*. The simplest line group consists solely of translations T^n , where n is an integer and T is the shortest translation. (See Fig. 9.) In general a line group could be generated by a rotation, reflection, or glide reflection combined with translation. It is an infinite group.

There are seven different types of line groups associated with frieze patterns. The international notation used for describing these groups is given in the form $rijk$, where r is to indicate translation as represented by the row. i can take the value 1 to indicate there is n

o rotational symmetry or 2 to indicate there is a half-turn symmetry. m is used for the reflection symmetries, j can take the value m to indicate there is a vertical symmetry otherwise we leave j out and k can take the value m to indicate there is a horizontal symmetry otherwise we leave k out.

These groups can be described with a simple diagram using the letters of alphabet.

No rotations or reflections	L L L L L L L L L L L L L L	r1
Half-turns	N N N N N N N N N N N N N N	r2
Vertical reflections	V V V V V V V V V V V V V V	r1m
Horizontal reflections	D D D D D D D D D D D D D D	r11m
Vertical and horizontal reflections with half-turns	H H H H H H H H H H H H H H	r2mm
Glide reflections	L Γ L Γ L Γ L Γ L Γ L Γ L Γ L	r11g
Vertical reflections and glide reflections with half-turns	$\nabla \Delta \nabla \Delta \nabla \Delta \nabla \Delta \nabla \Delta \nabla \Delta \nabla \Delta \nabla \Delta$	r2mg

The **TranslationSeveral** program of sec 5.1, could be modified to produce different types of the frieze groups.

In the **f** is defined to be a set of data points forming the motif used in creating a frieze pattern. The constant a is to determine the horizontal translation $T[\{x_, y_ \}] = \{x+a, y\}$. It should be defined according to the width of the motif used. The value n represents the number of motifs desired in the frieze pattern.

Program 5.9: FriezePolygon.

```

Frieze[f_,a_,n_] := (
  T[{x_,y_}]={x+a,y};

  (* the location of the motifs starting at the origin*)
  locations=NestList[T,{0,0},n-1];
  Show[

```

```

Graphics[
    Polygon/@Map[f,locations]],
    PlotRange->All,
    Axes->None, AspectRatio->Automatic])
*****

```

Let us try this program with the following function, **bird**.

```

bird[{x_,y_}]=
    Table[{x+2 Mod[n,2],y+Floor[n/3]},{n,6}];
Frieze[bird,2,10]

```



Note that the result is a frieze pattern with the line group rl .

We **verticaln** d e f i n e
 which creates frieze patterns with the line group rlm .

```

t1=Table[{x+2-Mod[n,2],y+Floor[n/2]},{n,6}];
t2=Table[{x+Mod[n,2]-2,y+Floor[n/2]},{n,6}];
vertical[{x_,y_}]=Join[t1,Reverse[t2]];
FriezePolygon[vertical,5,10]

```



Further, the program **FriezePolygon**, can be modified to create frieze patterns with *glides*.

Program 5.10. FriezeGlide.

```

*****

```

```

FriezeGlide[f_,a_, n_] :=(

```

(* reflecting the motif with respect to the x-axis*)

```

g[{x_,y_}]=f[{x,y}].{{1,0},{0,-1}};

```

(* defining the location of each motif*)

```

T[{x_,y_}]= {x+a,y};
locations=NestList[T,{0,0},n-1];

```

(* implementing the two motifs on alternating points *)

```
mapp[t_,f1_,f2_] := (r = Range[Length[locations]]);
                    re = List /@ Select[r,EvenQ];
                    ro = List /@ Select[r,OddQ];
                    MapAt[f1,MapAt[f2,locations,re],ro]);
```

```
Show[
  Graphics[Polygon/@ mapp[t,f,g]
    ],
  PlotRange->All,
  Axes->None, AspectRatio->Automatic]
```

Using the function **vertical** defined above, with the program **FriezeGlide**, we can produce frieze patterns with line group $r2mg$.

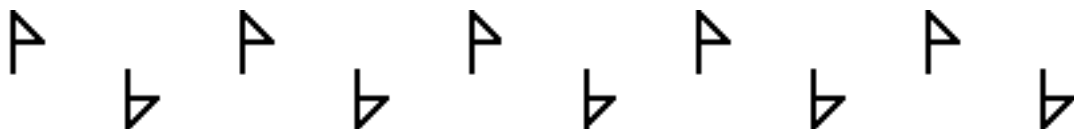
```
FriezeGlide[vertical,4, 10]
```



If we modify the program **FriezeGlide**, by replacing the command **Polygon** in the program with the command **Line**, we can draw motifs to form a $r1lg$ group.

```
flag[{x_,y_}] :=
  {{x,y},{x,y+2},{x+1,y+1},{x,y+1}}
```

```
FriezeGlide[flag,4, 10]
```



Let us investigate a other types of *Mathematica* programs to produce differnt frieze patterns.

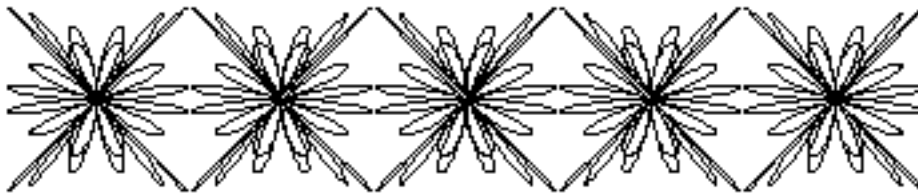
In the following **Buntgerfly** program of the section 5.3, to create frieze patterns with the line groups $r2mm$ and $r11m$.

Program 5.11. FriezePattern.

```
FriezePattern[a_, b_, c_] := (
bas[x_] :=
ParametricPlot
  [{Sin[a t] Cos[ b t]+ x, Sin[a t] Sin[ c t]},
   {t, 0, 2 Pi}, AspectRatio->Automatic,
   Axes-> None, DisplayFunction->Identity];
Show[bas/@Range[0,8,2],
      DisplayFunction->$DisplayFunction])
```

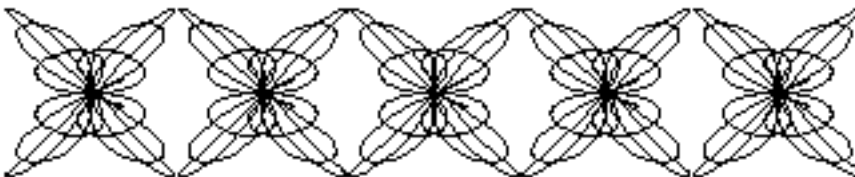
To create a frieze pattern with the line group $r2mm$, we use:

FriezePattern[12, 2, 1]



To create a frieze pattern with the line group $r11m$, we use:

FriezePattern[7, 5, 2]



In the following program we use the equation of an spiral to create a frieze pattern with the line group $r2$.

Program 5.12. HalfTurn.

```
v[t_,d_]:= {t Cos[t]+ d,t Sin[t]};
```

(* plotting an spiral*)

```
sp1[d_]:=ParametricPlot[v[t,d],
{t, 0, 4 Pi}, AspectRatio->Automatic,
Axes-> None, DisplayFunction->Identity];
```

(* plotting a reflected spiral*)

```
sp2[d_]:=ParametricPlot[-v[t,d],
{t, 0, 4 Pi}, AspectRatio->Automatic,
Axes-> None, DisplayFunction->Identity];
```

(* connecting the spirals at different intervals*)

```
Show[{sp1/@Range[0,161,46],sp2/@Range[-25,-163,-46]},
      DisplayFunction->$DisplayFunction]
```



As our final example, we use a nice feature of *Mathematica*, called **Interpolation** function to draw motifs.

Cubic Splines Interpolation.

Many times we are interested in graphing a shape for which there is no mathematical equation. Say we want to graph a swan.



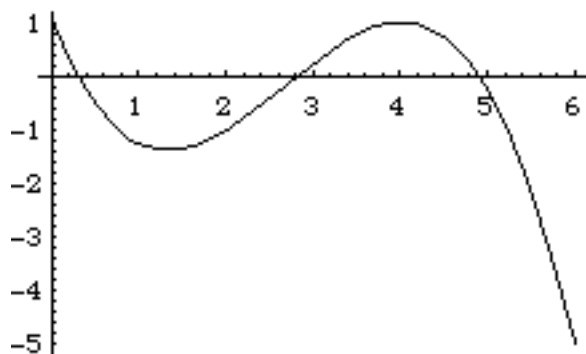
Fig 12.

A primitive way of drawing such a figure is to make a large table consisting of many point and connecting them using the command **Line**. Such an approximation even with a large amount of data points does not produce the satisfactory smoothness. A much better way of drawing such a figure is the interpolatory approach, most commonly based on *cubic splines*.

Suppose that we have $m+1$ data points P_0, \dots, P_m through which we wish to draw a curve such that two successive pair of data points P_i and P_{i+1} , are connected by a curve given by $y_i(u)$ forming a smooth curve for $i=0, \dots, m$. As it turns out the best possibility for such a curve $y_i(u)$ is a cubic polynomial in the parameter u of the form $y_i(u) = a_i + b_i u + c_i u^2 + d_i u^3$, with $y_i(0) = y_i$ and $y_i(1) = y_{i+1}$. The coefficients a_i, b_i, c_i , and d_i are determined so that each $y_i(u)$ connects to $y_{i+1}(u)$ in a smooth way. The smooth curve resulting by connecting the cubic polynomial curves is called a cubic spline. For a detailed description on how to determine these coefficients see [Bartels, Beatty & Barsky].

Mathematica has a built-in function called **Interpolation** based on the theory of cubic splines. The way this function works is by fitting cubic polynomials curves between the points you specify. For instance:

```
points={{0,1},{2,-1},{4,1},{6,-5}};
Plot[Interpolation[points][x],{x,0,6}]
```



Interpolation T h e

f u

can only produce open curves that is to say a curve for which the endpoints do not coincide.

It is however possible to write a simple routine to fit a smooth curve through a set of points to form a closed curve.

Program 5.13. ClosedCurve.

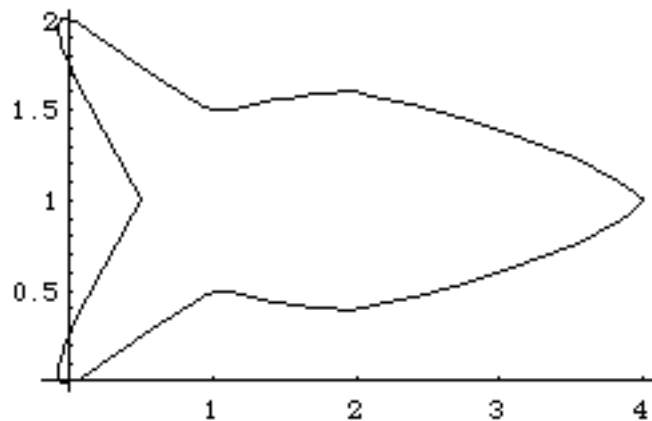
```
*****
```



```

ClosedCurve[s_List]:=
    Transpose[s];
inter[l_,t_] := Interpolation[#][t]& /@ Transpose[l];
F[x_] = inter[s,x];
ParametricPlot[Evaluate[F[t]],{t,1,Length[s]})
*****
fish ={{0.5, 1}, {0, 2},{1, 1.5}, {2,1.6},{4, 1},
        {2,.4},{1,.5} ,{0,0},{0.5,1}};
ClosedCurve[fish]

```



Applying **closed** to another set of points produces the following shark. The teeth and the eye were added.



We can now modify the program **ClosedCurve** to a program which can draw several motifs given by a list of data points to form a frieze pattern. The constant a , should be larger than the width of each motif used in the pattern.

Program 5.14. SequenceMotif.

```

*****
SequenceMotif[s_List,a]:=

```

(* plotting the motif *)

```

Motif[y_] := (
    Transpose[y];
inter[l_,t_] := Interpolation[#][t]& /@ Transpose[l];
F[x_] = inter[y,x];
    ParametricPlot[Evaluate[F[t]],{t,1,Length[y]},
        DisplayFunction->Identity]);

```

(* implementing the motifs on a sequence of points *)

```

ta= Table[{a,0}, {i,Length[s]}];
seq=Table[s+i ta, {i,0,5}];
Show[{Motif/@seq}, AspectRatio->Automatic,
    Axes->None, DisplayFunction->$DisplayFunction,
    PlotRange->All])

```

The following data points will produce a frieze pattern consisting of swans with the line group rl .

```

swan={ {0,3}, {0.5,3}, {0.05,1.6}, {0,0}, {4,0}, {5,1},
{4.5,0.5}, {2.5,1}, {0.25,0.75}, {1, 3}, {0.7, 3.7},
{0.5, 3.5}, {0.5, 3.5}, {0.4, 3.5}, {0.5, 3.5}, {0., 3}};

```

```

SequenceMotif[swan,7]

```



Exercise Set 5.4

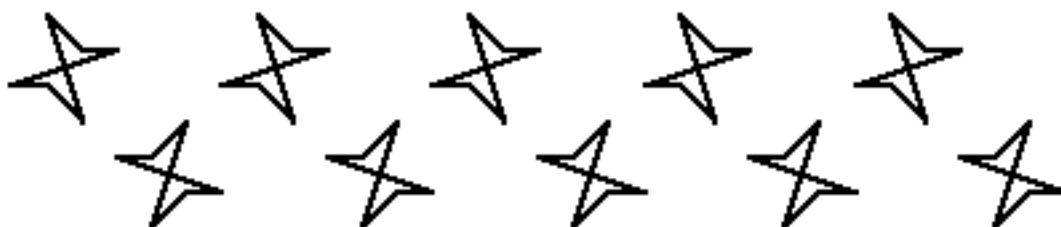
- In a line group show that the following statements are true.
 - If s is a rotation then sTs^{-1} is a translation.
 - If m is a reflection then mTm^{-1} is a translation.
 - If g is a glide reflection, g^2 is a translation.
 - What symmetries are described by gs , sm , mgm^{-1} , and sgs^{-1} ?

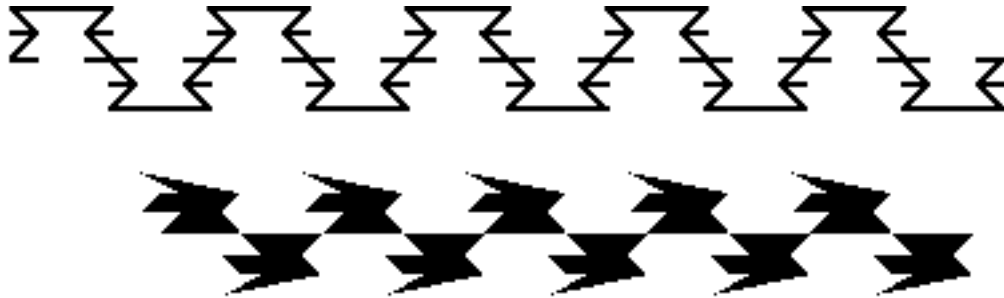
2. Use the program **FriezePolygon** to produce designs with the line groups $r1$, $r2$, $r1m$, $r11m$, and $r2mm$.

3. **vertical** M o d i
to produce a frieze pattern similar to the following frieze pattern.



4. Identify the line group of the following frieze patterns.





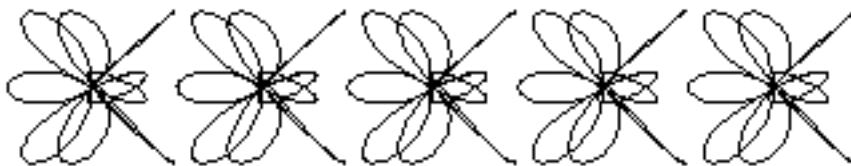
All of these patterns were produced by the program **FriezeGlide** using the function **vertical** with different **table-f** functions (Sometimes **t2** was eliminated). Experiment by using different constants and other table-functions to produce other frieze patterns.

5. **FriezeLine** T
 can be formed by replacing the command **Polygon** in the program **FriezePolygon** with the command **Line**.
 Experiment with **vertical** is program with different constants or any other similar table-functions to produce different frieze patterns such as this one.

6. Identify the line group belonging to the following frieze patterns.

FriezePattern[9,5,9],
FriezePattern[11,3,6],
FriezePattern[13,14,15].

7. One easy way of creating frieze patterns with the line group rlm , is by rotating the function $\{\sin[at] \cos[bt], \sin[at] \sin[ct]\}$, by 90° , and then substituting the result in the program **FriezePattern**. Here is the frieze pattern resulted from this modification for the values $a = 9, b = 3, c = 5$.



Try this modification with the following values: $a = 9$, $b = 5$, $c = 9$ and $a = 13$, $b = 14$, $c = 15$ and $a = 27$, $b = 3$, $c = 5$.

8 .

HalfTurn to reproduce other patterns with the line group $r2$.

9 .

HalfTurn to reproduce the pattern given in the Fig. 9 and Fig. 11.

10. Use the program **FriezePattern** to reproduce the pattern given in the Fig. 10.

11. Use the program **ClosedCurve** to reproduce a similar swan to the one given in the Fig. 12. How about a shark!

12. Use the program **SequenceMotif** together with the function **fish** to reproduce a frieze pattern. What is the line group of this frieze pattern?.

13. Use the program **SequenceMotif** to reproduce two frieze patterns with different line groups.

5.5. Wallpaper Patterns.

A *wallpaper pattern* consists of a *motif*, repeated at regular intervals in more than one direction. The framework on which any wallpaper pattern is built is called a *net*. A net is an array of points determined by an origin O and two translation vectors T_1, T_2 , not in the same direction. All points in the net can be obtained by the translations

T

Error!

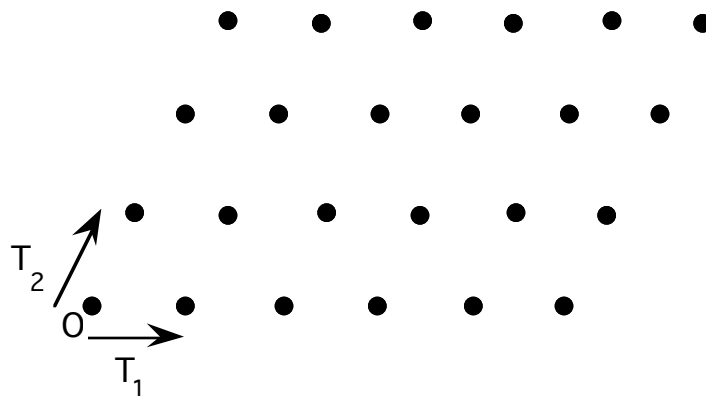


Fig. 13.

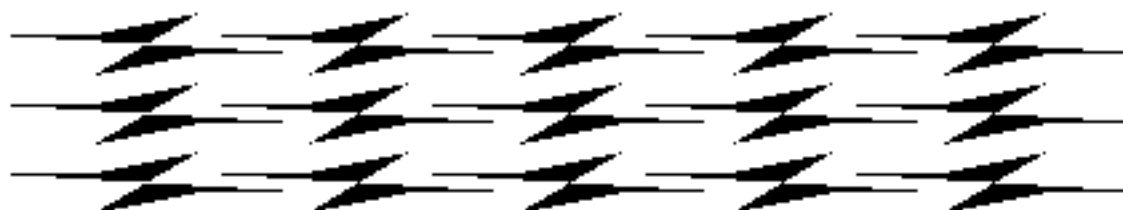
The net represents the mode of repetition. Any motif can be used, provided it is repeated in the way indicated by the net. Our illustrations are necessarily finite but the patterns are assumed to extend to infinity in all directions.

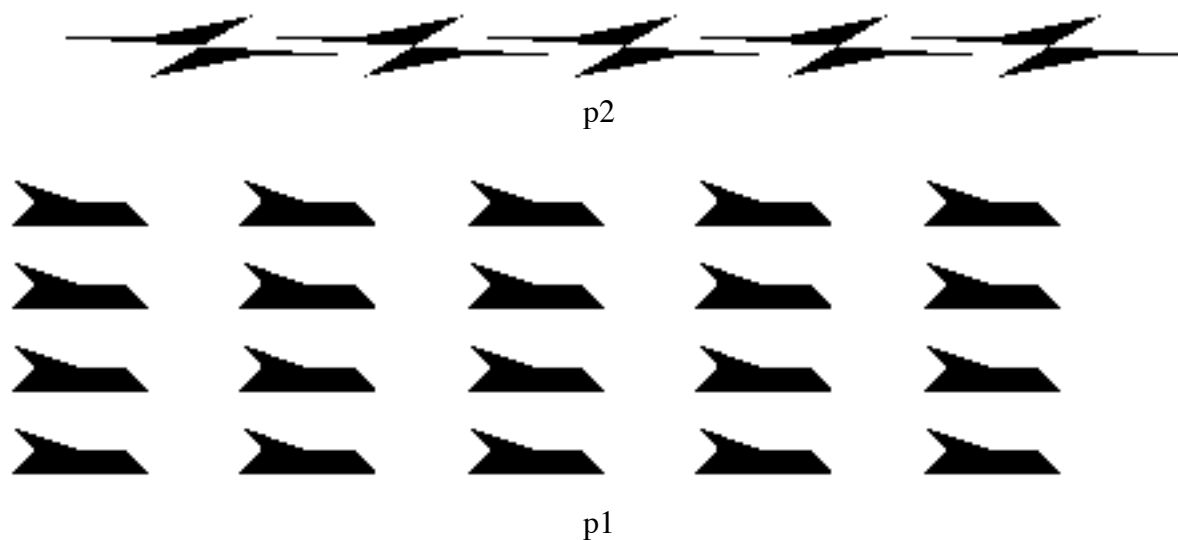
There are only five types of nets possible for the wallpaper patterns. All nets have diad (180^0) rotation symmetry about the centers of the parallelograms, the vertices and the mid-points of the sides.

Parallelogram: The only symmetry of the parallelogram cell is the diad symmetry. A wallpaper based on this net may or may not have the diad symmetry as a whole. If the motifs used have the diad symmetry singly then the pattern as a whole has the diad symmetry and is referred to as $p2$, otherwise it does not have the diad symmetry and is referred to as $p1$.

The international notation used for describing the wallpaper group is given in the form $pijk$, where the prefix p is to indicate repetition in two directions. i can take the value 1 to indicate there is no diad symmetry or 2 to indicate there is a diad symmetry. m is used for the reflection symmetries, j can take the value m to indicate there is a vertical reflection symmetry otherwise we leave j out and k can take the value m to indicate there is a horizontal reflection symmetry otherwise we leave k out.

The following figures were produced by a simple modification of the program **TranslationSever** of the section 5.1.





Rectangular: In addition to the diad symmetry the rectangular cell has both horizontal and vertical reflection symmetry about the centers of the rectangles and their vertices.

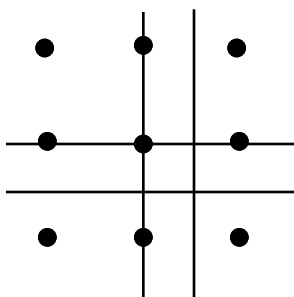
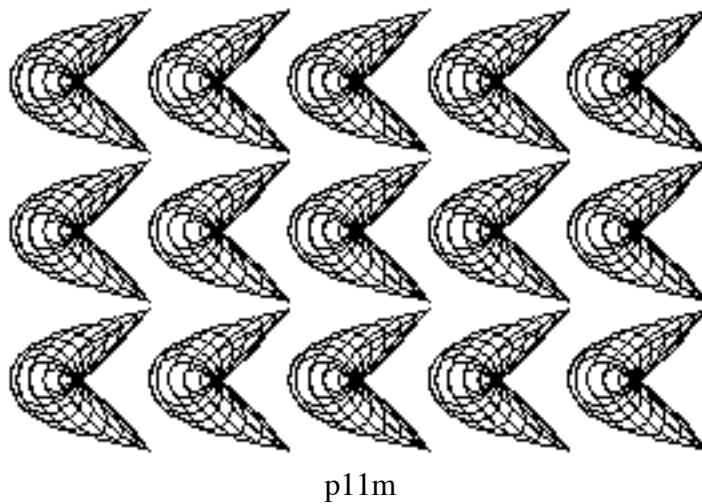
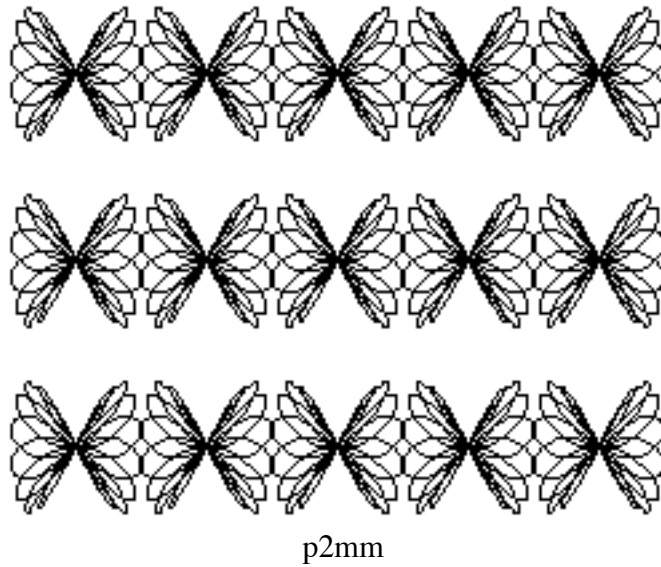


Fig. 14.

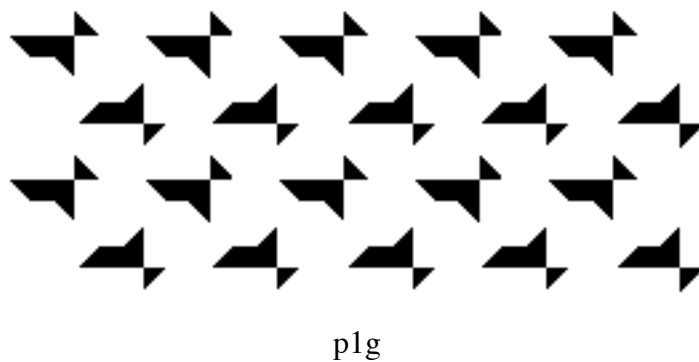
A motif applied to this net may have reflection symmetry in one or both directions. If the motifs have D_4 -

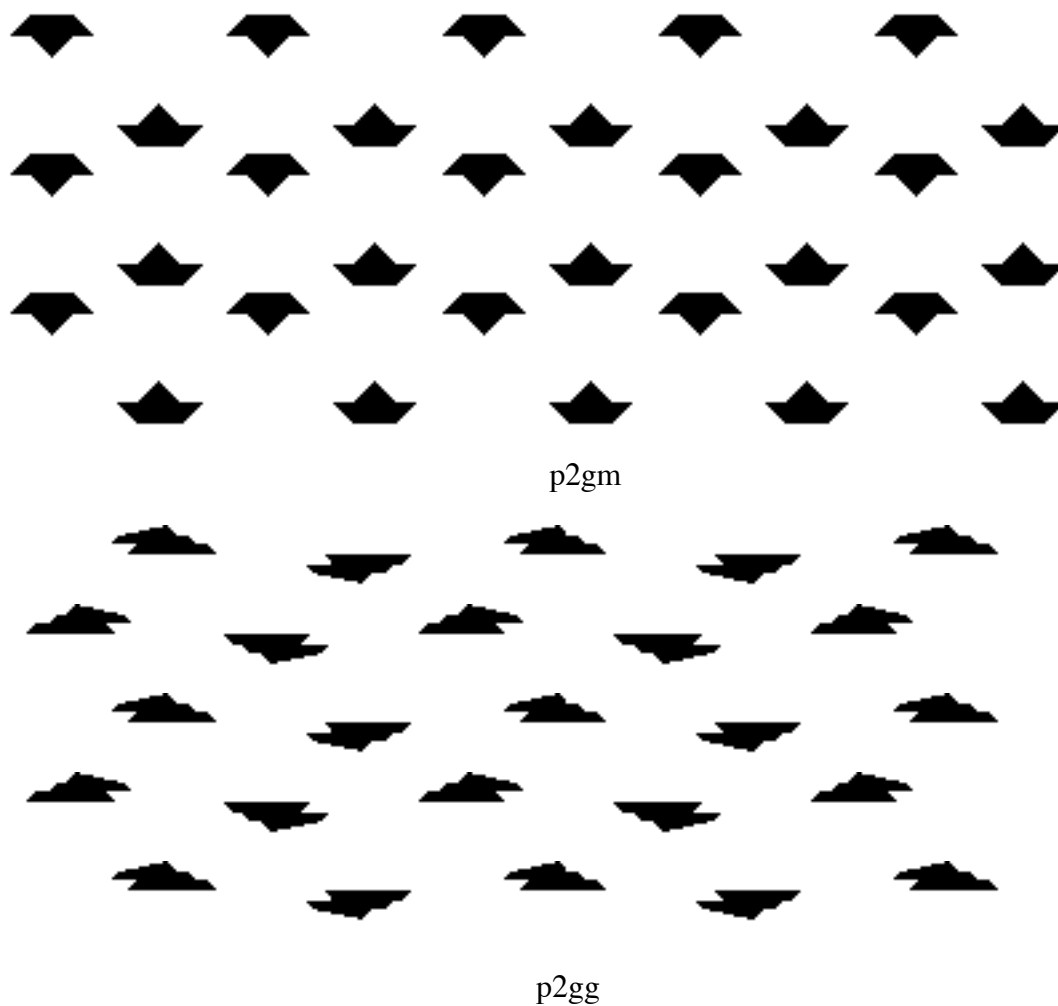
as their symmetry groups then the pattern is referred to as $p2mm$. If the motif has reflection symmetry only with respect to the x-axis, it is referred to as $p1m$.

Note that since we can interchange the x-axis and the y-axis in a wallpaper group, patterns of the form $p11m$ are considered congruent to the patterns of the form $p1m$. The following figures were created using a modified **Version 1.0** program of section 5.4.



As with frieze patterns, there are glide reflections associated with the rectangular cell s. We can derive three new wallpaper patterns by using glide reflections in one or both directions. The following **FriezeGlide** of section 5.4.





Rhombus : The Rhombus cell which is also referred to as centered rectangular has both horizontal and vertical reflection symmetry about the vertices and half way between the vertices and the center of the rhombus.

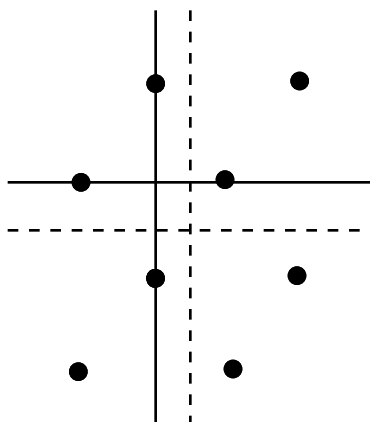
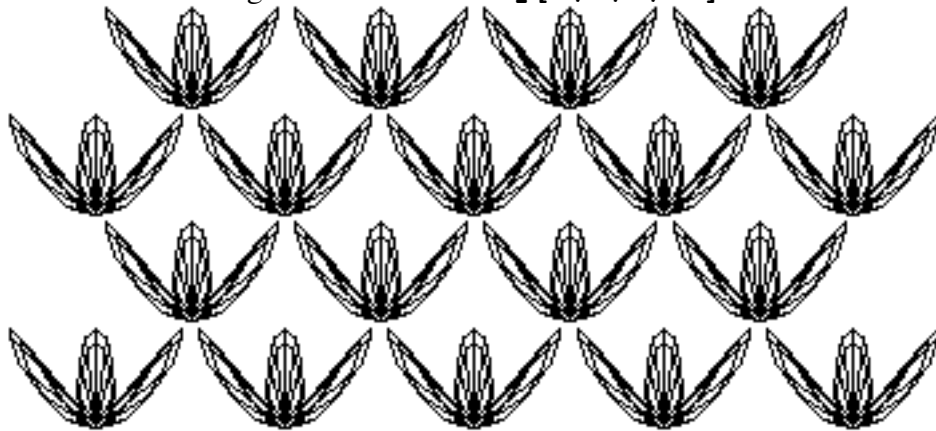


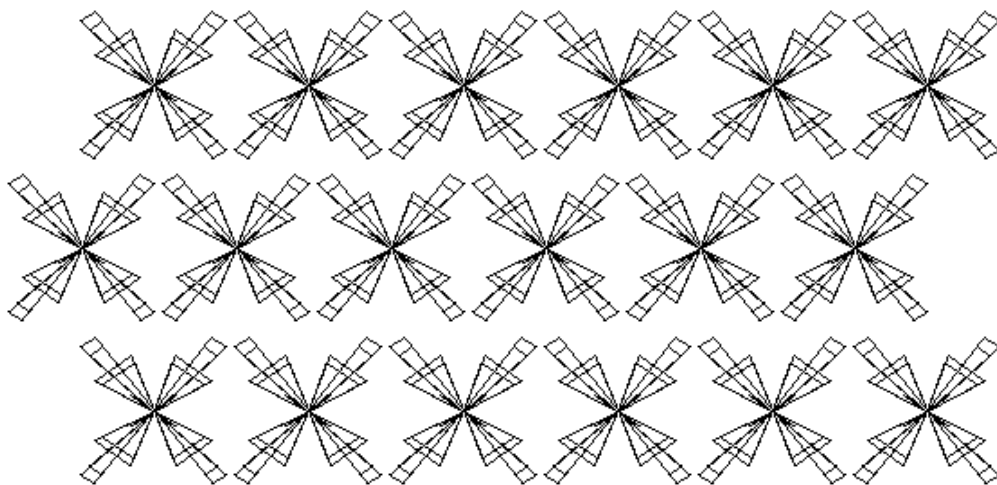
Fig. 15.

To distinguish the rhombus nets with the rectangular ones we use the letters c instead of p as a prefix to indicate repetition with a centered rectangular net. A motif applied to this net may have reflection symmetry in one or both directions. If the motifs have D_4 as their symmetry groups then the pattern is referred to as $c2mm$. If the motif has reflection symmetry only with respect to the y -axis, it is referred to as $c1m$. Note that if the motif has reflection symmetry only with respect to the x -axis, it is referred to as $c11m$ but this pattern is congruent to $c1m$ and does not count as a separate pattern. Since the glide reflection occurs naturally in these two types no other group is obtained by considering glide reflection.

The following figures **MoireButterfly** created using `cMoireButterfly[9,3,9,180]` and the second using `cMoireButterfly[5,3,2,30]`.



$c1m$



$c2mm$

Square: With a square net there is a wider range of possible symmetries. In addition to the diad symmetry about the mid-points of the sides, the net also has a tetrad symmetry (45^0 rotation) about the vertices and centers of the squares. There are reflection lines not only along the squares, with intermediate lines through the centers, but also along the diagonals, with intermediate glide lines.

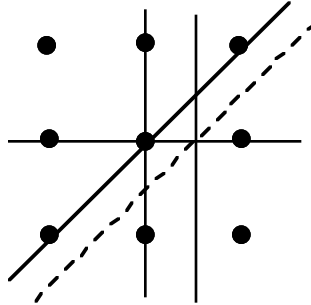


Fig. 16.

There are three patterns based on a square net. The motifs may have a tetrad rotation ($p4$), a tetrad rotation with reflections in both sides of the square, ($p4mm$) or a tetrad rotation with a glide in the center of the square, ($p4gm$). Patterns of this type are shown below.

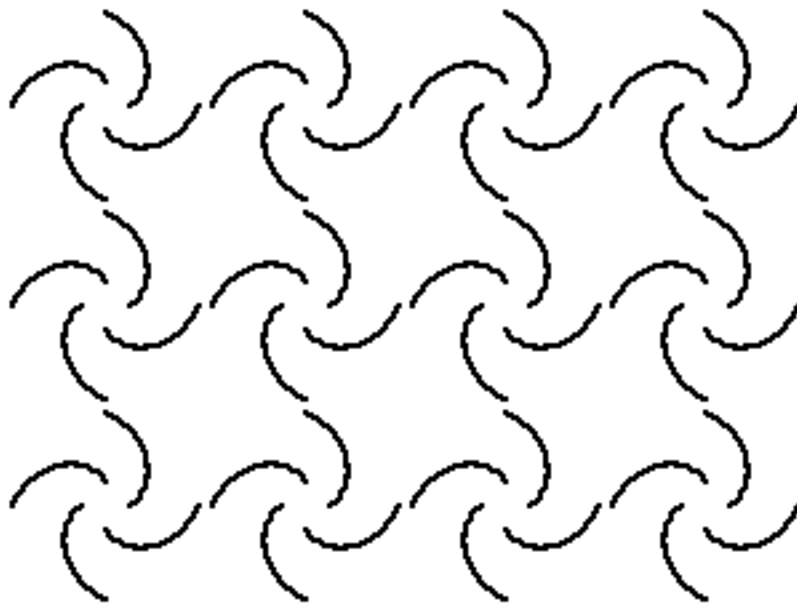
The pattern $p4$ below was created by a repetition of **Spiral[4]** given by the following program:

Program 5.15: Spiral

```
*****
Spiral[n_] := (
xp[t_, s_] := (t+.5)*( Cos[s] Cos[t]+Sin[s] Sin[t]);
yp[t_, s_] := (t+.5)*(-Cos[t] Sin[s]+Cos[s] Sin[t]);

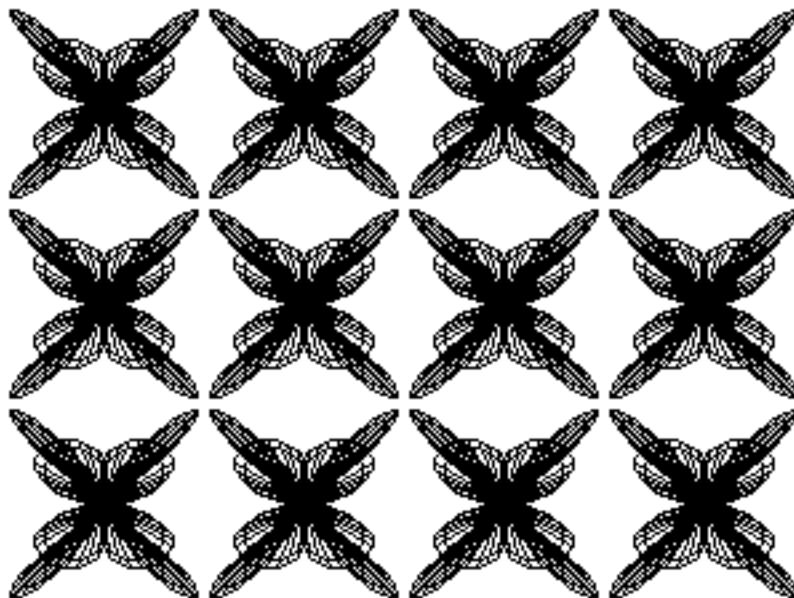
sp[s_] := ParametricPlot[
  {xp[t,s],yp[t,s]},{t, 0, Pi/2},
    AspectRatio->Automatic,
    Axes->None,DisplayFunction->Identity,
    PlotStyle->Thickness[.01]];

Show[{sp /@ Range[0, 2 Pi, 2 Pi/n]},
  DisplayFunction->$DisplayFunction]
*****
```



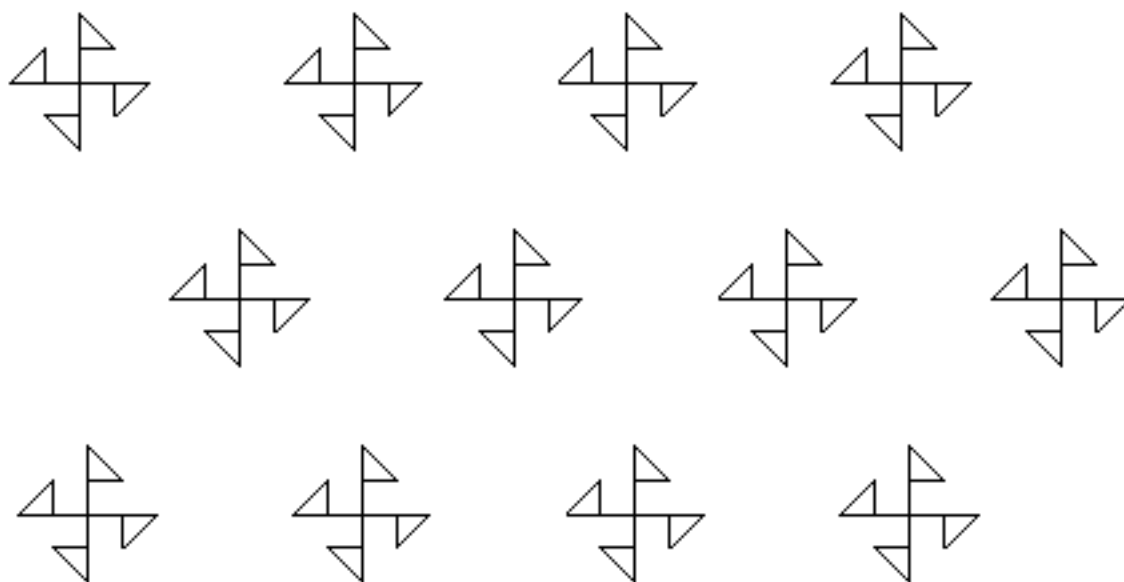
p4

The figure p4mm below was produced by **MoireButterfly[5,3,2,180]**.



p4mm

The figure p4gm below was created by using the **translationSeveral** of the section 5.1, with a modified function **flag**.



p4gm

Hexagonal: The 60° rhombus, which divides into two equilateral triangles had hexad rotation (60° rotation) about the vertices of the triangles, triad rotation (120° rotation) about their centers, and diad about the mid-points of the sides, as well as reflection in the sides and altitudes of the triangles. We call this net *hexagonal*.

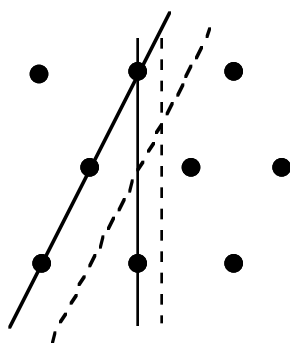


Fig. 17.

The motifs superimposed on a hexagonal net may have a triad rotation (p3), a triad rotation with reflections in the altitudes of the triangles (p3m1) or reflections in the sides (p31m). They also may have a hexad rotation (p6), or a hexad rotation with reflections in both the altitudes of the triangles and the sides (p6mm). Patterns of this type are shown below.

The following pattern p3 was created using a series of images created by the program **MoiréSpiral** given below with $n=3$, $m=40$.

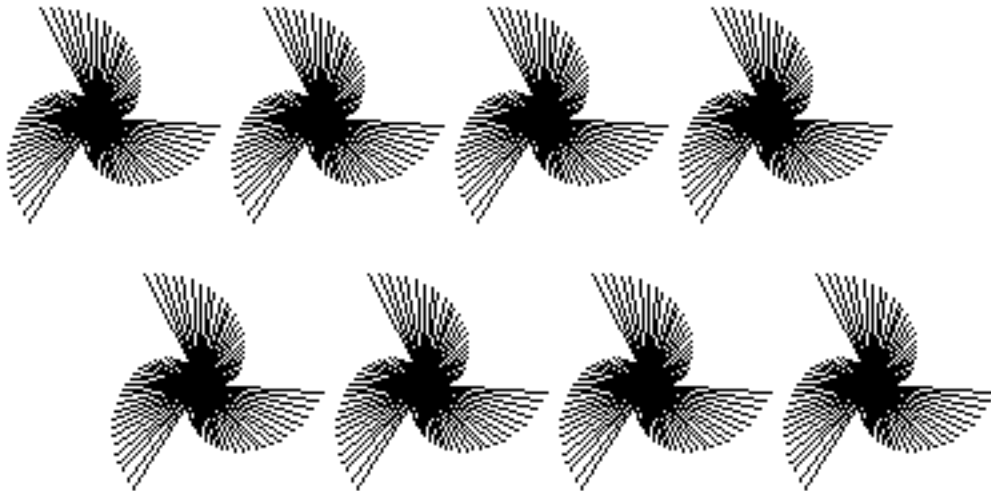
Program 5.15 MoiréSpiral

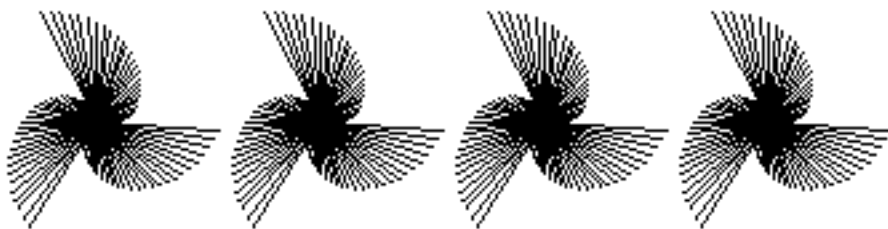
```
*****
MoiréSpiral[n_, m_] := (
xp[t_, s_] := (t+.5)*( Cos[s] Cos[t]+Sin[s] Sin[t]);
yp[t_, s_] := (t+.5)*(-Cos[t] Sin[s]+Cos[s] Sin[t]);

r[t_, x_] := {xp[t,x], yp[t,x]};

v[s_] := (values= r[#,s]& /@ Range[0, 2 Pi/n, Pi/m/N ];
Graphics[{Thickness[.0005],
Map[Line[{{0,0}, #}&, values] ]});

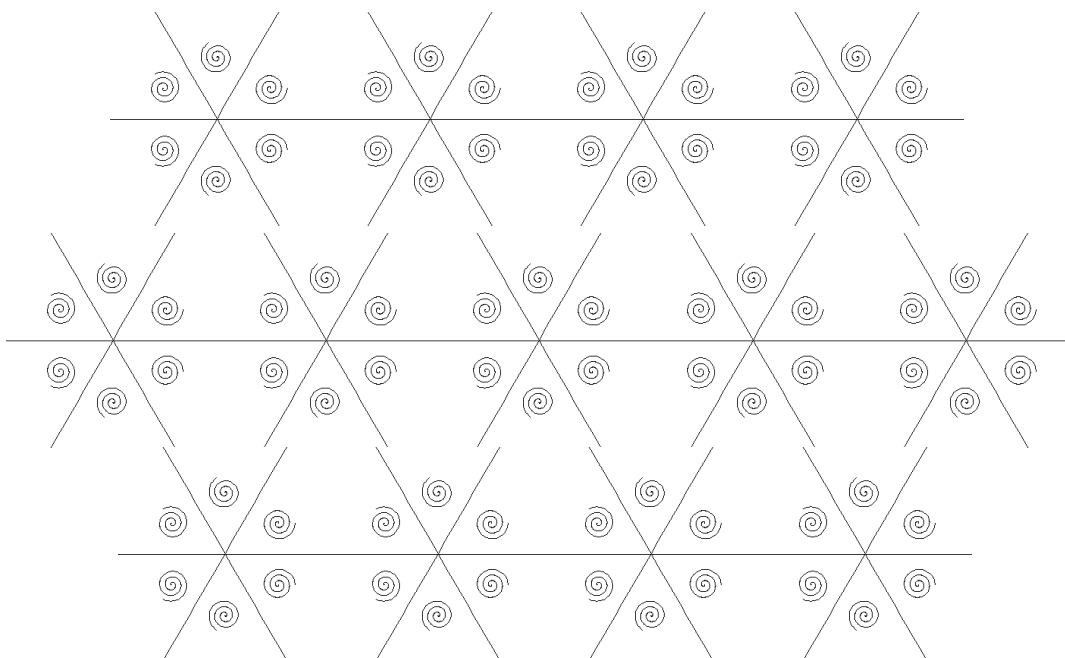
Show[{v /@ Range[0, 2 Pi, 2 Pi/n]},
PlotRange->All, AspectRatio->Automatic,
DisplayFunction->$DisplayFunction]
*****
```





p3

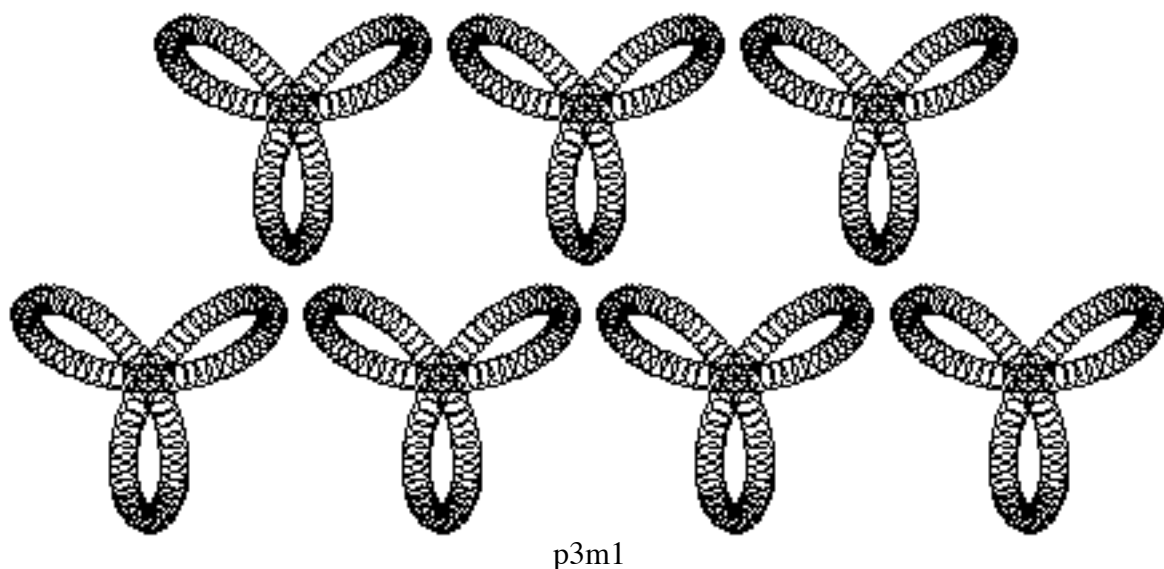
The pattern p31m below was created using the program **ReflectSeveral** of the section 5.2. (See problem 7, exercise set 5.2)



p31m

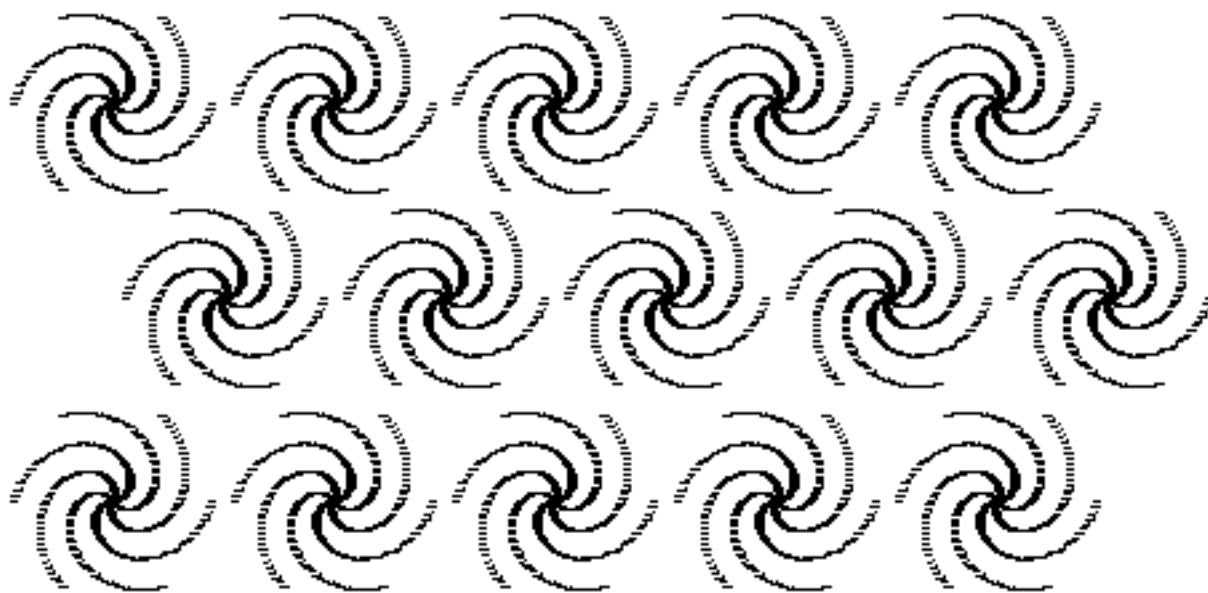
5.4. The pattern p3m1 below was created using the program **Symmetric** of the section





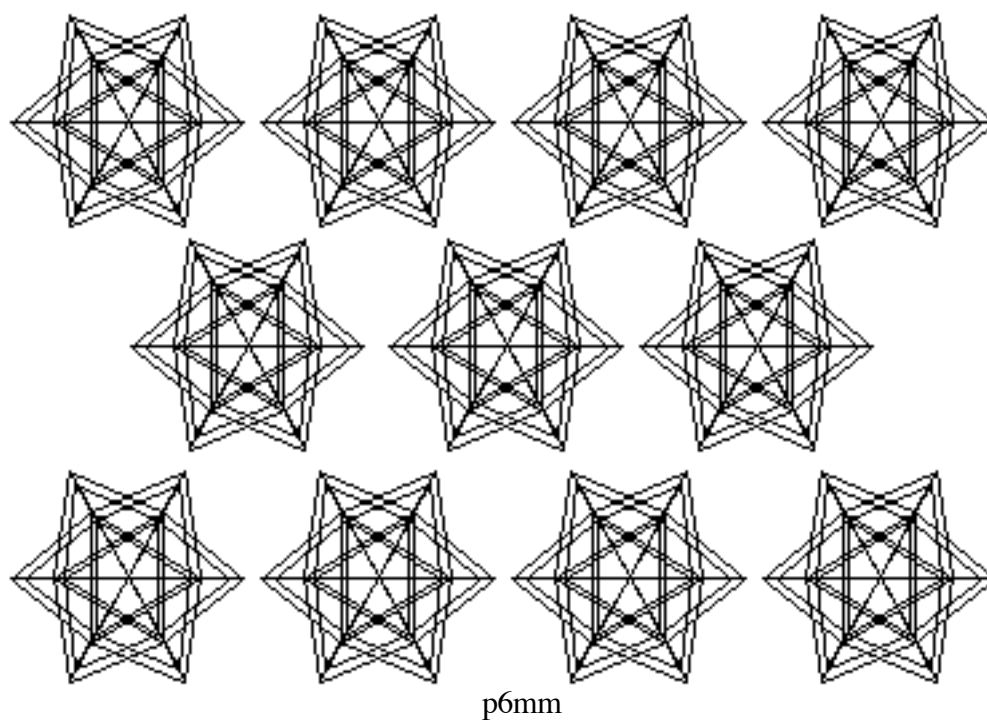
p3m1

The following pattern p6 was also created using the program **Symmetric**.



p6

The last image in the hexagonal net is p6mm created by **MoireButterfly**[6,7,7,30].



Exercise Set 5.5

Generate several examples of the wallpaper group by creating different images using the programs discussed in this chapter.