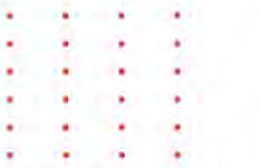
The Five Types of Plane Lattice

Parallelogram

```
TO PARALLEL
  GRID ( - 60 ) 90 80 50 205
END
```

Rhombic

```
TO RHOMB
  GRID ( - 30 ) 90 80 80 225
END
```

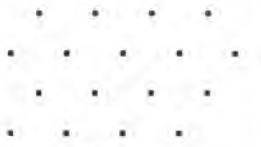


Rectangular

```
TO RECT
  GRID ( - 80 ) 90 80 50 180
END
```

Square

```
TO SQUARE
  GRID ( - 80 ) 90 80 80 180
END
```



Hexagonal

```
TO HEX
  GRID ( - 30 ) 90 80 80 210
END
```

procedure:

```
TO REFLECT
  DEFINE "UNIT REWRITE "UNIT
END
```

The rewriting now involves replacing RT with LT, and vice versa, as well as MOTIF with R.MOTIF, and vice versa. Our previous version of the rewriting procedure swapped only RT and LT (see page 915). To modify it, all we need to do is change CHANGE.WORD — which now becomes:

```
TO CHANGE.WORD :WORD
  IF ( ANYOF :WORD = "RT :WORD = "RIGHT ) THEN
  OUTPUT "LEFT
  IF ( ANYOF :WORD = "LT :WORD = "LEFT ) THEN
  OUTPUT "RIGHT
  IF :WORD = "MOTIF THEN OUTPUT "R.MOTIF
  IF :WORD = "R.MOTIF THEN OUTPUT "MOTIF
  OUTPUT :WORD
END
```

Another way to approach this problem would be to use the version of REWRITE that also changes the subprocedures of the input procedure. We give this version amongst the answers.

For most of the patterns, the movement between points is simply a translation, and there are no other transformations to be performed. TRANX and TRANY, therefore, don't do anything. PATTERN17 is an example of this type:

```
TO PATTERN17 :PROC
  DEFINE "TRANX [[] []]
  DEFINE "TRANY [[] []]
  PAT "HEX 17 :PROC
  ERASE TRANX
  ERASE TRANY
END
```

Having covered all the basic possibilities, we leave the rest of the patterns for you to define.

Logo Flavours

For all LCSI versions:

Use CS for DRAW
Use OR for ANYOF
SETPOS, followed by a list, is used for SETXY
IF has a different syntax:

```
IF :WORD = MOTIF [OUTPUT "R.MOTIF]
```

TEXT and DEFINE do not exist as primitives in Atari LOGO, although the Atari manual does give a method of defining them

Exercise Answers

1. To rotate a shape about the point (X, Y) through an angle of A degrees:

```
TO ROTATE :X :Y :A
  PU
  MAKE "H HEADING
  MAKE "XOLD XCOR
  MAKE "YOLD YCOR
  MAKE "R SQRT ( :XOLD — :X ) * ( :XOLD —
  :X ) + ( :YOLD — :Y ) * ( :YOLD — :Y )
  PU
  SETXY :X :Y
  SETH TOWARDS :XOLD :YOLD
  RT :A
  FD :R
  SETH :H + :A
  PD
END
```

2. A rewrite procedure that will rewrite subprocedures as well.

```
MAKE "PROCS.DONE []

TO REWRITE :PROC
  MAKE "PROCS.DONE FPUT :PROC
  :PROCS.DONE
  OUTPUT REWRITE.PROC TEXT :PROC
END

TO REWRITE.PROC :TEXT
  IF :TEXT = [] THEN OUTPUT []
  OUTPUT FPUT REWRITE.LINE FIRST :TEXT
  REWRITE.PROC BUTFIRST :TEXT
END

TO REWRITE.LINE :LINE
  IF :LINE = [] THEN OUTPUT []
  IF LIST? FIRST :LINE THEN OUTPUT FPUT
  REWRITE.LINE FIRST :LINE REWRITE.LINE
  BUTFIRST :LINE
  OUTPUT FPUT CHANGE.WORD FIRST :LINE
  REWRITE.LINE BUTFIRST :LINE
END

TO CHANGE.WORD :WORD
  IF ( ANYOF :WORD = "RT :WORD = "RIGHT )
  THEN OUTPUT "LEFT
  IF ( ANYOF :WORD = "LT :WORD = "LEFT )
  THEN OUTPUT "RIGHT
  IF PROCEDURE? :WORD THEN
  SUBPROCEDURE :WORD OUTPUT WORD "£ :
  WORD
  OUTPUT :WORD
END

TO PROCEDURE? :NAME
  IF NUMBER? :NAME OUTPUT "FALSE
  IF LIST? :NAME OUTPUT "FALSE
  TEST WORD ? :NAME
  IFTRUE IF WORD? TEXT :NAME OUTPUT
  "FALSE ELSE IF NOT ( TEXT :NAME = [] )
  OUTPUT "TRUE
  OUTPUT "FALSE
END

TO SUBPROCEDURE :WORD
  IF MEMBER? :WORD :PROCS.DONE THEN
  STOP
  DEFINE ( WORD "£ :WORD ) REWRITE
  :WORD
END
```