

\$12.95

THE COMMODORE 64

LOGO WORKBOOK

by **M.J. Winter**

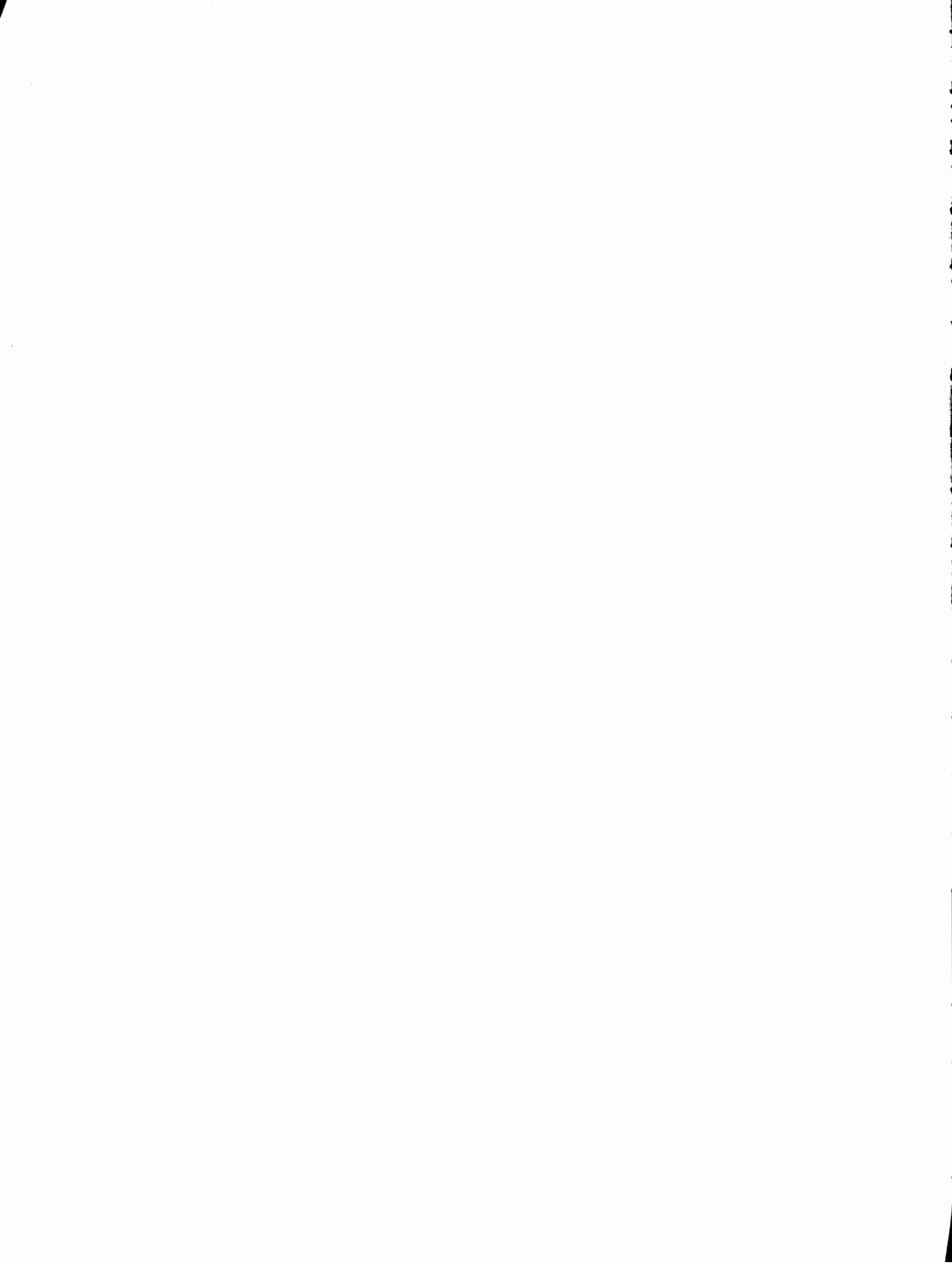


Explorations with the Turtle



The Commodore 64

LOGO WORKBOOK



The Commodore 64

LOGO WORKBOOK

by

M. J. Winter, Ph.D.

**Professor of Mathematics
Michigan State University**

**Illustrated by
Martin Cannon**



20660 Nordhoff Street, Chatsworth, CA 91311-6152
(818) 709-1202



ISBN 0-88190-364-7

Copyright © 1984 by DATAMOST, Inc.
All Rights Reserved

This Manual is published and copyrighted by DATAMOST, Inc. Copying, duplicating or otherwise distributing this product is hereby expressly forbidden except by prior written consent of DATAMOST, Inc.

The word Commodore and the Commodore logo are registered trademarks of Commodore Business Machines, Inc.

Commodore Business Machines, Inc. was not in any way involved in the writing or other preparation of this manual, nor were the facts presented here reviewed for accuracy by that company. Use of the term Commodore should not be construed to represent an endorsement, official or otherwise, by Commodore Business Machines, Inc.

Printed in U.S.A.

For Elizabeth B. MacKenzie

Acknowledgments

This is the third in a series of books written for young people (but not just for young people).

The first, *Computer Playground*, introduced BASIC through means of graphics, games, and words.

The second, *Compumath Magic*, showed those who already knew some BASIC how to use it to solve problems.

Now comes the *LOGO Workbook*, designed both to introduce the LOGO language and to teach how to use it in problem solving situations.

All three have been published by DATAMOST. It is a pleasure to acknowledge the support of Dave Gordon and the assistance of Marcia Carrozzo who has edited all the books. The artwork of Martin Cannon has been an integral part of these books.

I thank Jack Rarick, an innovative and dedicated teacher, who allowed me to haunt his classroom this year. Some of the projects in the *LOGO Workbook* were sparked by his 4th and 5th grade students. Jack Rarick supervised the classroom experiments of all the lessons, and made many valuable suggestions.

Thanks also to Linda Riest whose various computer classes, with children aged 8–14, tested these materials.

I am grateful to my colleagues in the mathematics department of Michigan State University and fellow members of the Middle Grades Mathematics Project: Bill Fitzgerald, Glenda Lappan, Elizabeth Phillips, and John Wagner, each of whom shared their

experiences with me and helped provide insight into the minds of children. Bill and Glenda especially share my interest in geometric activities as well as in using computers in teaching and learning mathematics.

For the last three summers I helped prepare and teach in “The Computer Camp” summer camp at MSU. Final thanks go to my fellow Computer Camp staff members: Ed Carlson, Mark Lardie, and John Forsyth. We’ve all learned a lot from our experiences with computers and kids.

Table of Contents

What is LOGO?	11
Why Learn It?	11
LOGO and Young Children	13
About the Workbook	15
Using the Workbook With Different Age Groups	15
List of LOGO Primitives	16
Comments and Descriptions	18
LOGO on the Commodore 64	23
Starting Up	24
Meet the Turtle	25
Lesson 1 Square Turns	29
Lesson 2 Sharp Turns	33
Lesson 3 Combines Turns	37
Lesson 4 Teaching the Turtle* Procedures and Repeat	41
Lesson 5 Using Procedures	45
Lesson 6 Super Procedures	49
Lesson 7 Erasing and Retracing	53
Lesson 8 Pen Up and Pen Down	57
Lesson 9 Inputs Telling LOGO What Size You Want	61
Lesson 10 Endless Repeats	67
Lesson 11 Donuts and Sugar Donuts	71
Lesson 12 Circles	75
Lesson 13 Thirds of Circles	79
Lesson 14 Keeping Track of the Turtle	83
Lesson 15 Clover and Petals	87
Lesson 16 Keeping on Track	91
Lesson 17 Setting Things Down	93
Lesson 18 Headings	97
Lesson 19 Christmas Tree	101
Lesson 20 Random	103

Lesson 21	Pinetrees	107
Lesson 22	Repeats With a Difference	111
Lesson 23	Solid Colors	117
Lesson 24	Using Set to Draw	119
Lesson 25	Rings	123
Lesson 26	Defining Variables	127
Lesson 27	Keep Counting	131
Lesson 28	Spirals & Angle Spirals	135
Lesson 29	Pressing a Key	141
Lesson 30	Solid Shapes	145
Lesson 31	Flags	151
Lesson 32	Fancy Forwards	157
Lesson 33	Fractals	163
Lesson 34	Snowflakes	167
Lesson 35	Outputs	171
Lesson 36	Mirror Images	175
Lesson 37	Turtle Billards	181
Lesson 38	Words	185
Lesson 39	Gluing Words	189
Lesson 40	More Words	193
Lesson 41	Lists	197
Lesson 42	Gluing Lists	201
Lesson 43	Tell LOGO Something	205
Lesson 44	Number Games	209
Lesson 45	Testing	213
Lesson 46	Search and Replace	217
Appendix 1	Comments About Editing	221
Appendix 2	Formatting a Disk — Saving and Reading Files ...	223
	Index of LOGO Primitives	225
	Index to Procedures Used in the Workbook	227

What is LOGO?

Why Learn It?

LOGO is called an educational language. It was designed for students, it is easy to begin using, and it is very powerful, i.e., it has the capability to do a lot of things. The educational value of learning LOGO, however, reaches far beyond the discipline of learning a programming language. A student learning LOGO will also learn:

- An increased understanding of geometry.
- Thinking processes and understanding of language.
- Problem solving techniques and the ability to handle multi-step problems.
- An appreciation of the power of recursion.

Two-thirds of the *LOGO Workbook* is directly concerned with geometric experiences. Students make the turtle draw pictures and designs. They investigate the effects of changes: does the box grow? will the pattern ever be complete? what happens if the angle gets smaller?, etc. The visual satisfaction of the turtle's drawings is a powerful motivation for investigation.

While using LOGO to draw boxes of different sizes, children will concretely experience the meaning of a variable. They will reinforce the concepts of points on a graph, and, of course, the measure of an angle. They will be able to investigate symmetry and the idea of a limit.

Problem solving skills are developed and applied at every level. To draw a house, one must first break down the problem into its

components: frame, roof, door, and window. When each piece or subprocedure has been solved, then one must put them together—synthesized into a greater whole.

Perceiving similarity among problems—“This is just like what we did but now the side is bigger”—leads to recursion, one of the most powerful mathematical and programming techniques available. Do it for this size, make the size bigger, do it again, keep going. . . .

Because the turtle’s graphics are so delightful, many people neglect the word handling capabilities of LOGO. Yet the ability to manipulate words, sentences, and lists is essential to an understanding of grammar. LISP and the languages used in artificial intelligence work are all what are called “list processing languages,” as is LOGO.

The last third of the *Workbook* deals with sentences and words. The applications are mainly in a “game” category. Children may be content with just doing the problems, or they may become intrigued enough to want to learn more.

Some aspects of LOGO have not been touched in the *LOGO Workbook*, which is after all, an introduction to the language. The mathematical commands—QUOTIENT, REMAINDER, ROUND, INTEGER—were not included. The purpose of the book is to teach LOGO through geometric applications and some list handling.

LOGO and Young Children

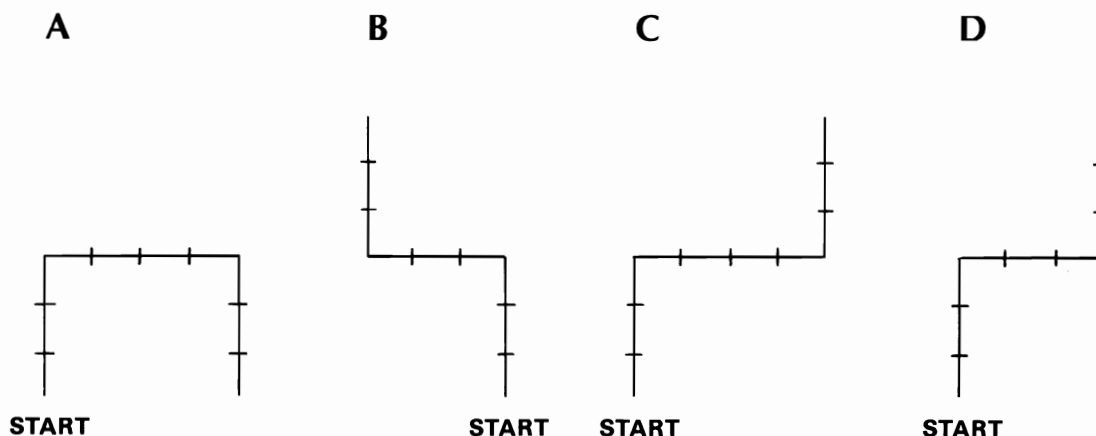
(What to do before starting the workbook)

Many children will not immediately see the connection between saying FORWARD and seeing the turtle move up the television screen. Before using LOGO, young children should first “play turtle.” Speak commands such as:

Forward 3 steps
Right turn
Forward 4 steps
Left turn
Forward 3 steps

Be sure they do not advance when told to turn.

Show them these paths. Ask them to choose which is the one they just made.



Lay out a 3-segment path with masking tape, restricting the turns to right 90 degrees or left 90 degrees. At first, mark the steps on the tape. Have the children write out a set of directions

to travel the path. Blindfold a child and have her follow the directions. Did she walk the path?

A set of directions is called a procedure. Have the children write out directions to walk around a square or rectangle. Test the procedures using a blindfold.

Make the point that “right turn” and “left turn” refer to the child’s right and left. In LOGO, they refer to the turtle’s right and left.

On the LOGO Utility Disk that comes with the Commodore LOGO package is a program called INSTANT. This program uses single letter commands: R turns the turtle right 30 degrees, so that to make a square corner, RRR must be typed. This is an excellent introduction to LOGO for younger children.

About the Workbook

Using the Workbook With Different Age Groups

LOGO can be meaningful to different people in different ways. An 8-year-old will be pleased with teaching the turtle to draw designs. An older user will enjoy the designs, but will obtain even more satisfaction by understanding how to move the turtle on its “graph paper,” thereby being able to draw solid figures and construct much more complex procedures. Not every lesson in the *LOGO Workbook* is for every user.

Suggested sequences of lessons are given below. The age groups are approximate, and offered as a guideline only.

Ages	8–9	10–12	13–??
Lessons	0–11 27 29 38–44	0–14 17–20 22–24 26–29 30–31 (optional) 35 38–45	0–3 lightly 4–46

The first group includes the 3rd and 4th grades. The second group has some acquaintance with graphing on x,y-axes. Classroom experiments have shown that 5th graders can do most of the lessons in the middle track. Beyond age 12, most children will have attained Piaget’s level of variable understanding. They will understand and learn from the advanced lessons.

It is not necessary to complete each lesson before continuing, but the first two or three problems should be attempted. The final problems in many of the lessons are open-ended. These may challenge some users, as well as encourage individual exploration.

List of LOGO Primitives

The words that LOGO recognizes without any “teaching” are officially called primitives. In the *LOGO Workbook* they are often referred to as commands. Below is a list of the primitives used in the book, and the lesson in which they first appear.

<u>Primitive</u>	<u>Long Form</u>	<u>Lesson</u>
DRAW	—	1
FD	FORWARD	1
RT	RIGHT	1
LT	LEFT	1
TO	—	4
END	—	4
REPEAT	—	4
SAVE	—	4
READ	—	5
PC	PENCOLOR	7
BK	BACK	7
BG	BACKGROUND	7
DOUBLECOLOR	—	7
SINGLECOLOR	—	7
RANDOM	—	10, 20
PR	PRINT	10
MAKE	—	10, 22
HT	HIDETURTLE	12
ST	SHOWTURTLE	12
SETH	SETHEADING	14

<u>Primitive</u>	<u>Long Form</u>	<u>Lesson</u>
HEADING	—	14
IF	—	14
STOP	—	14
SETXY	—	17
HOME	—	17
RANDOMIZE	—	20
XCOR	—	21
YCOR	—	21
SETX	—	21
SETY	—	21
MAKE	—	26
RC	READCHARACTER	29
RC?	—	29
MEMBER?	—	29
ELSE	—	29
OP	OUTPUT	35
SQRT	—	35
COUNT	—	38
FIRST	—	38
BF	BUTFIRST	38
LAST	—	38
BL	BUTLAST	38
ITEM	—	38
WORD	—	39
SE	SENTENCE	42
LIST	—	42
RQ	REQUEST	43
TEST	—	45
IFT	IFTRUE	45
IFF	IFFALSE	45
ANYOF	—	45
ALLOF	—	45
POTS	PRINTOUT TITLES	APPENDIX
CATALOG	—	APPENDIX

Comments and Descriptions

Meet the Turtle is an introduction to the turtle and its movements. The connection between “a turtle” and the LOGO turtle is made by a sequence of drawings. All LOGO commands used here are repeated in Lesson 1.

Lesson 1 is designed to reinforce the idea of right and left turns. All turns are either LT 90 or RT 90.

Lesson 2 introduces turns of 120 and 135. The greater the turn, the sharper the angle drawn by the turtle.

Lesson 3 uses supplementary (adding to 180) angles. Try making a diamond using 160 and 20.

Lesson 4 introduces procedures. If each procedure is written on an index card, a convenient file can be assembled. Later, a student will be able to pull out the procedures needed for a particular lesson.

Lesson 5 is an exercise in visual problem solving. Students must first identify the components of each drawing, then experiment with the order used.

Lesson 6 illustrates how elaborate designs can be made by repeating simple shapes.

Lesson 7 defines BK and PC. Problem 5 requires planning ahead: deciding where to place the flowers, then deciding how to get there.

Lesson 8 introduces systematic variations. Problems 2 and 3 create different pictures by varying only one thing at a time. This systematic variation is a valuable problem solving technique.

Lesson 9 introduces variables. Problem 8 has two steps: first draw a picture in the box, then write a procedure to draw it on the screen. Other pictures might be a castle or cathedral.

Lesson 10 uses *recursive* procedures. Problem 4 is an exploration that can be extended. A turning angle of 144 draws a 5-pointed star. In general, a figure with N sides will make N turns through angle A. The figure will be complete when the total of the angles turned through, NA, is a multiple of 360.

$$NA = K*360$$

For a 5-pointed star, $N = 5$, and the equation becomes

$$A = K*72.$$

If $K = 1$, then $A = 72$ and the result is a pentagon.

If $K = 2$, then $A = 144$.

What happens if $K = 3$?

What A is needed for a 10-pointed star?

Lesson 11 These drawings generate solid-looking figures. Problem 3 is a study of the effect of each variable.

Lesson 12 A regular polygon with a large number of sides looks like a circle. A circle with diameter D has circumference $\pi*D = 3.14*D$. A polygon with N sides, each side of length $3.14*D/N$ will have the same circumference (perimeter).

The following all approximate a circle of diameter :DIAM

```
REPEAT 360 [FD :DIAM*3.14/360 RT 1]
REPEAT 180 [FD :DIAM*3.14/180 RT 2]
REPEAT 120 [FD :DIAM*3.14/120 RT 3]
REPEAT 90 [FD :DIAM*3.14/90 RT 4]
REPEAT 72 [FD :DIAM*3.14/72 RT 5]
```

The product of the number of repeats times the angle is always 360.

Lesson 13 A review of arcs. The designs are pleasing.

Lesson 14 STOP ends the procedure PATTERN and goes back to PATT.

Lesson 15 PETAL repeats TINYARC RT 120 TINYARC three times. Different designs are produced from the basic pattern.

Lesson 16 This lesson is harder than it looks. The students must decide how to draw the tracks, then position and move the turtle.

Lesson 17 A review of x,y-coordinates may be necessary before starting this lesson. Many children are not familiar with the axes crossing in the middle of the graph paper.

Lesson 18 More complex pictures usually require orienting the turtle before each part is drawn.

Lesson 19 An open-ended, fun project. Some packages could be placed under the tree.

Lesson 20 To make LOGO think of a number from 1 to 6, use:

(RANDOM 6) + 1

Lesson 21 More on coordinates. Using SET is a fast way to draw lines.

Lesson 22 This lesson deals with recursive procedures that change the variable before calling themselves. :SIDE is the number that follows SHRSQ. If :SIDE was originally 100, it becomes 90, then 80, then 70 . . . decreasing by 10 each time SHRSQ is called. “Walk” students through this procedure.

Lesson 23 Fun with solid colors.

Lesson 24 More complicated drawings using SET. Problem 4 simulates animation.

Lesson 25 How to specify a circle by its center and radius. CIRCLE depended on the location and orientation of the turtle. RING does not.

Lesson 26 Practice in defining, or redefining, a variable.

Lesson 27 More recursive procedures—this time used for counting. How to count from 1 to 1000.

Lesson 28 More practice with MAKE, this time in a drawing exercise. Encourage able students to form conjectures from the results of Problem 7.

Lesson 29 This is the first interaction with a running procedure. The line:

```
IF MEMBER? :ANS "AEIOU PR. . . . .
```

means, if :ANS is one of the letters of AEIOU then print. . .

Lesson 30 Filling in a solid by shading it as it is drawn. It may help to demonstrate the back-and-forth movement on the board.

Lesson 31 A problem solving activity. Students must decide the order in which the parts of each flag should be drawn.

Lessons 32, 33, 34 The special forwards defined in Lesson 32 are used recursively to produce interesting drawings and patterns. These lessons can lead to open-ended explorations.

Lesson 35 Handling subprocedures that are functions. SGN and ABS will be useful in Lessons 36 and 37. Outputs are referred to again in Lesson 45, but can be introduced there.

Lesson 36 An excursion into symmetry. Try bouncing off the “mirrors.”

Lesson 37 Playing billiards with the turtle. This lesson does not depend on Lesson 36. The final corner depends on whether the (reduced) fraction :L/:W is odd/odd, odd/even, or even/odd. When changing to a game of pool, use ABS to test when the ball is very close to a corner.

Lesson 38 An introduction to the word functions: COUNT, ITEM, FIRST, etc.

Lesson 39 Using the word functions from Lesson 38, many complicated tasks can be performed.

Lesson 40 The student is asked to write a more complicated procedure using words.

Lessons 41, 42 An introduction to lists, and lists of lists. In Lesson 41, BABBLE could be extended to include a list of adjectives. The resulting sentence would look like adjective, noun, verb, adjective, noun.

Lessons 43, 44 Giving words to a running LOGO procedure. In Lesson 43, the procedure learns the names of animals. Lesson 44 is a “guess my number” game.

Lesson 45 uses TEST to decide whether to stop performing a recursive procedure. A test has an output—TRUE or FALSE. Use IFT and IFF to decide what to do next.

Lesson 46 The replacement procedure is a very elaborate combination of list functions, tests, and outputs. A student who doesn't understand its working (most won't) can still use it. The sequenced problems lead to a decoder to be used with an encoded message.

LOGO on the Commodore 64

The LOGO package produced by Commodore is the full MIT LOGO, with enhancements made possible by the design of the computer. The built-in enhancements include:

1. Using the function keys to go to the text screen, split screen, or full screen at any time.
Pressing <f 1> shows the text screen
 <f 2> shows the split screen
 <f 3> shows the full screen
These keys can be pressed at any time. The turtle will continue its activity without interruption.
2. From the Edit Mode, a procedure can be defined or saved by pressing the RUN/STOP key, as well as by the usual CTRL C.
3. The most recent line of commands is reprinted if the up-arrow (needs shift) is pressed. The usual CTRL P also works.

Using the LOGO Utilities disk provided in the package enables LOGO to:

4. Print copies of the screen. This takes some time as the picture must first be saved, then the printing program loaded and run.
5. Use sprites. This provides ability to have eight turtles of different shapes, sizes and colors on the screen at one time. Each "turtle" can be controlled independently. The

sprites, however, do not physically rotate when told to turn, e.g. the butterfly always faces up no matter what direction it moves.

6. Play music, using the sophisticated sound available on the Commodore 64.

The *LOGO Workbook* uses only the built-in features, 1–3.

Starting Up

The Commodore 64 LOGO package includes two disks:

- A LOGO utilities disk.
- A LOGO disk.

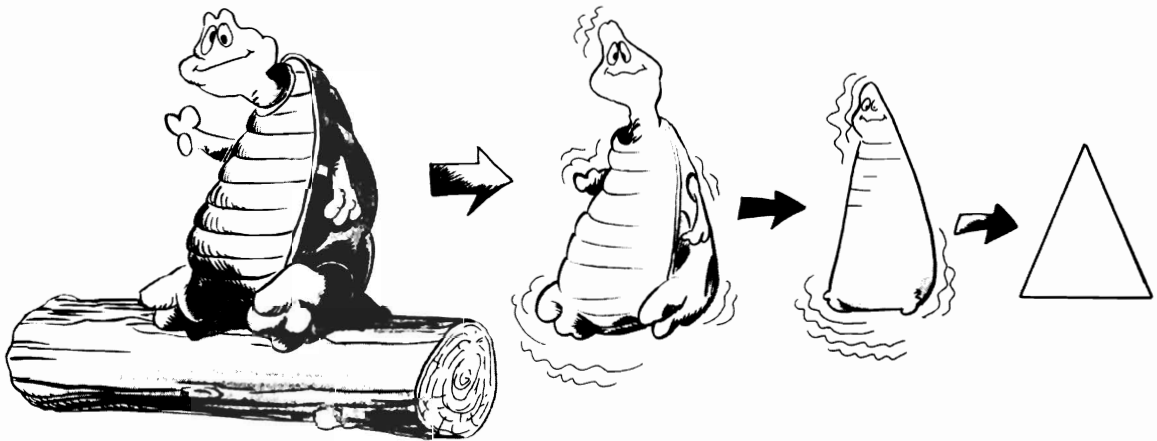
To load LOGO:

- Turn on the computer and disk drive.
- Insert the LOGO disk.
- Type `LOAD "LOGO",8` <return>.
- When `READY` appears, type `RUN` <return>.
- Wait about 3 minutes (may be less).

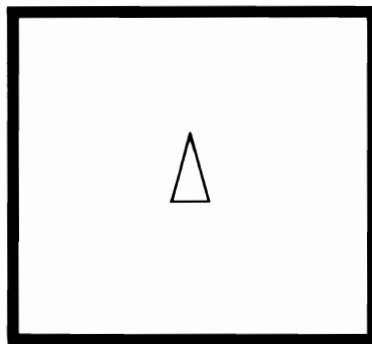
When `WELCOME TO LOGO!` appears on the screen you are ready to begin. Remove the LOGO disk and store it safely.

If after 3 minutes, LOGO hasn't begun, turn off the computer and start again.

From Lesson 4 on, it is possible to save the procedures you write. To do this you must have prepared a formatted disk before loading LOGO. See *Formatting a Disk* in Appendix 2 of this book.



Meet the Turtle



From Log to LOGO

The LOGO turtle is a thin triangle. The sharp corner is its head; the larger end is where its tail would be.

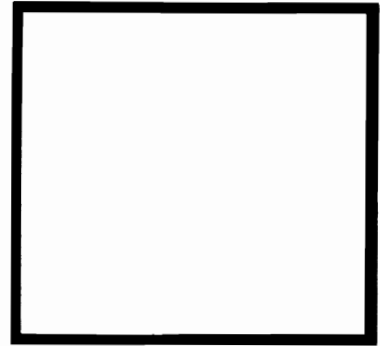
Type `DRAW` and press `<return>`.

`DRAW` clears the screen
puts the turtle in the center,
headed toward the top.

`FD 20` makes the turtle move forward 20 turtle steps.

Type FD 20 <return>
FD 60 <return>

Where is the turtle now? Make an X in the box.



Try FD 250

Type DRAW <return>. Try to find the exact number of turtle steps to reach the top of the screen.

It is about _____ steps to the top of the screen.

What happens if the turtle goes too far?

Type DRAW again. (Remember to press <return>.)
Then type:

RT 90

The turtle turned to *its* right.

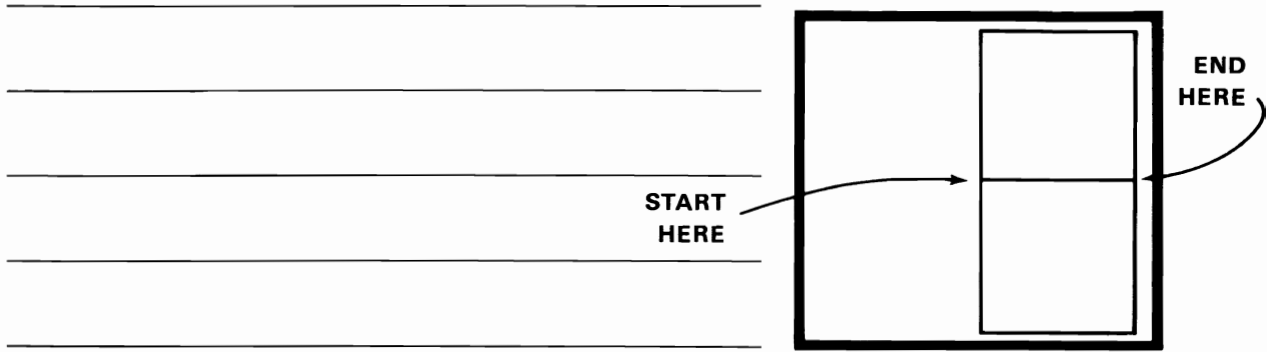
Experiment: How many turtle steps does it take to get to the right edge of the screen?

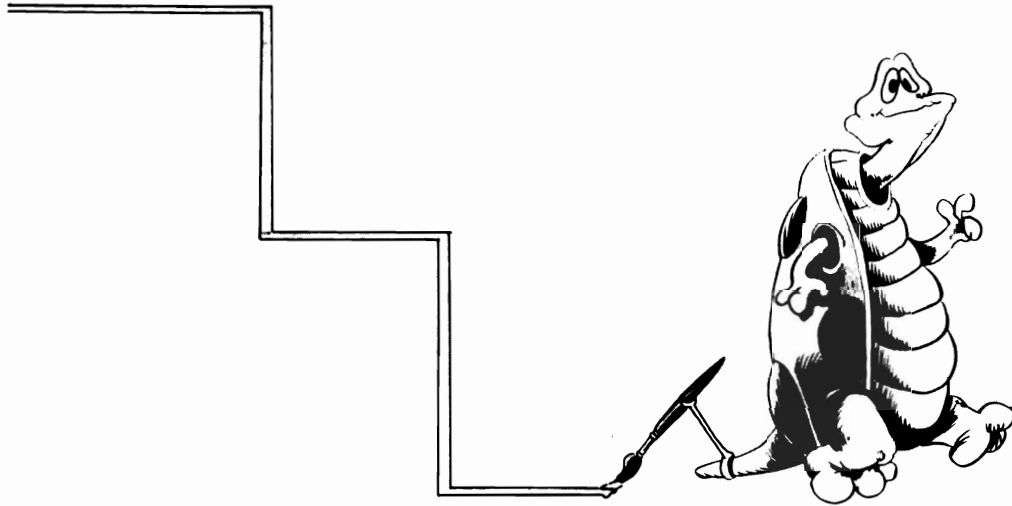
_____ steps.

Start with DRAW. Try to make the turtle draw this picture.
All the turns are RT 90s.

Write down everything you tell the turtle to do.

DRAW





Lesson 1

Square Turns

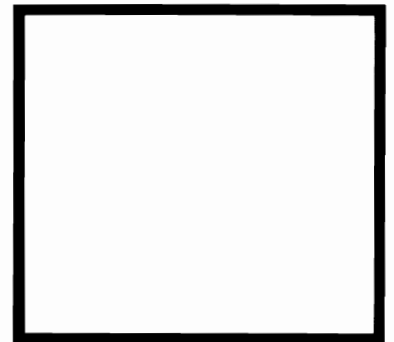
All the turns in these problems are LT 90 or RT 90.

Each problem should begin with a clear screen for the turtle to draw on. Before starting each program type:

```
ND <return>  
DRAW <return>
```

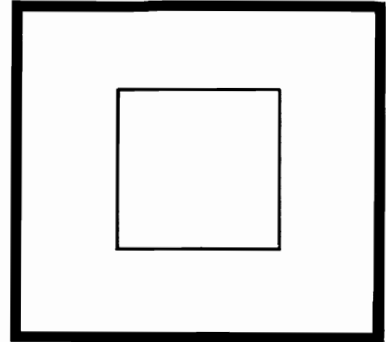
1. Type each line below and press <return>.
Watch the turtle move.
Copy the final picture in the box.

```
FD 30  
RT 90 FD 30  
RT 90 FD 30  
LT 90 FD 20
```

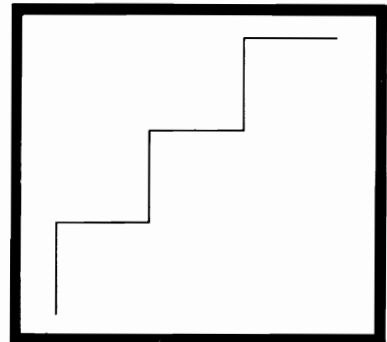


2. Complete the program so the turtle draws a square.

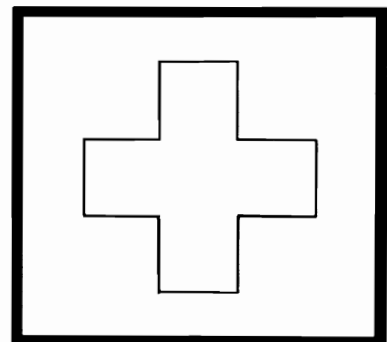
FD 30



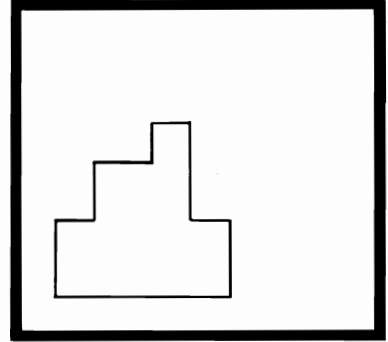
3. Make the turtle draw a flight of stairs.
Write your program here.

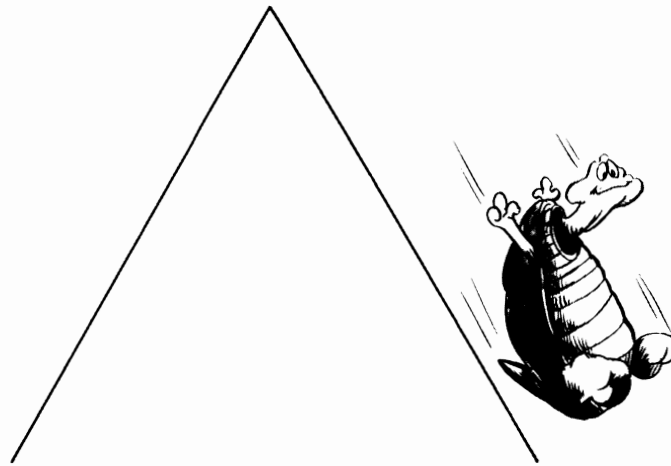


4. Make the turtle draw a plus sign.
Write your program here.



5. Make the turtle draw a skyscraper.
Write your program here.

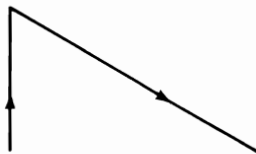




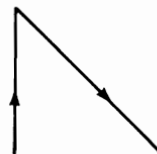
Lesson 2 Sharp Turns

These problems use turns of 120 and 135 to draw sharp corners.

RT 120

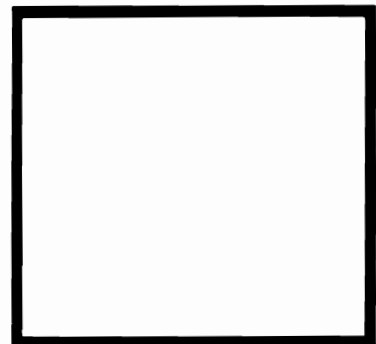


RT 135



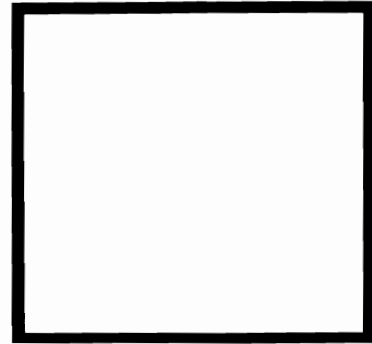
1. Press <return> after each line. Copy the final picture in the box.

```
FD 30  
RT 120 FD 30  
LT 120 FD 30  
RT 120 FD 30  
RT 120 FD 30
```



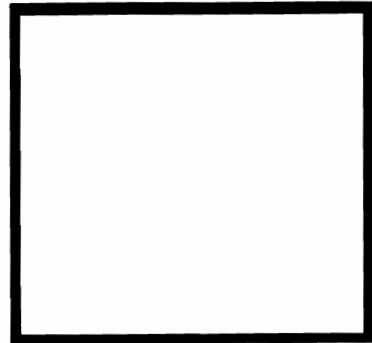
2. Using turns of 120, draw a triangle.
Write your program here.
Begin with:

FD 30



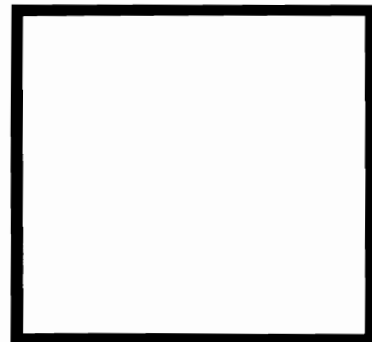
3. Copy the procedure and copy the final picture in the box.

FD 30
RT 135 FD 42
LT 135 FD 30
LT 135 FD 42

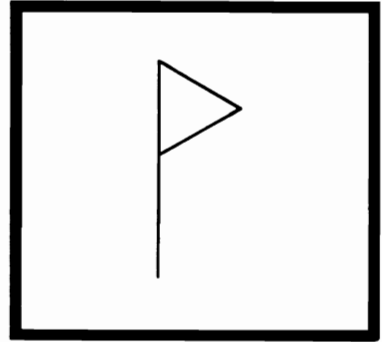


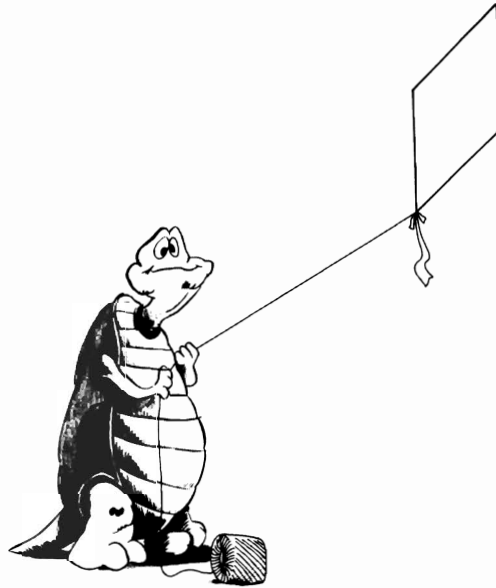
4. Using turns of 135, draw a triangle.
Write your program here.
Begin with:

FD 30



5. Draw the flag.
Write your program here.



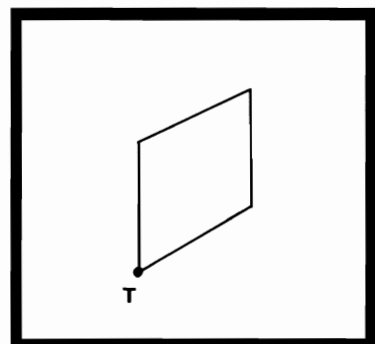


Lesson 3 Combined Turns

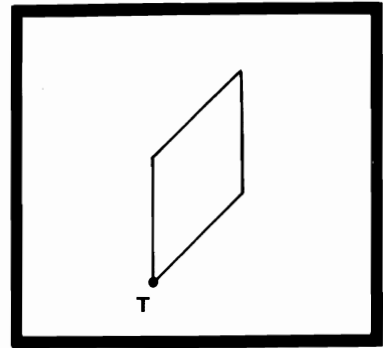
Diamonds can be made by combining turns of 120 and 60, or by combining turns of 135 and 45.

1. Using turns of 60 and 120, draw the parallelogram. The turtle begins at the point T.

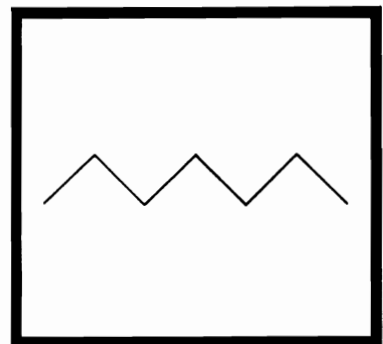
```
FD 40  
RT 60 FD 40
```



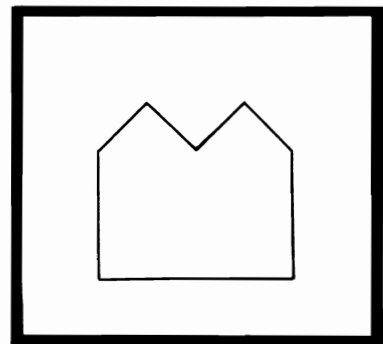
2. Make a parallelogram using turns of 45 and 135.
Write your program here.



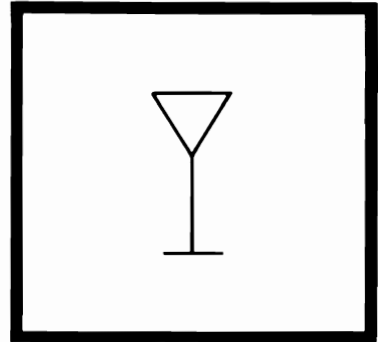
3. Make a mountain ridge. Turn the turtle before beginning to move forward. Write your program here.

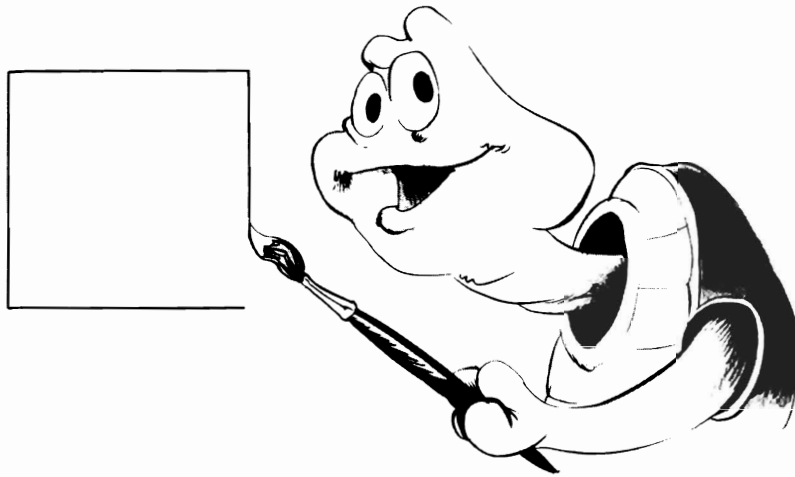


4. Draw a row of houses using turns of 45 and 90.
Write your program here.



5. Draw a glass.
Write your program here.





Lesson 4

Teaching the Turtle*

Procedures and Repeat

In Lesson 1 you drew a box (or square). To teach LOGO how to draw a box, type:

```
TO BOX <return>
```

The screen will change to black. LOGO is now in the Edit Mode. TO BOX appears at the top. Type in these lines, pressing <return> after each line.

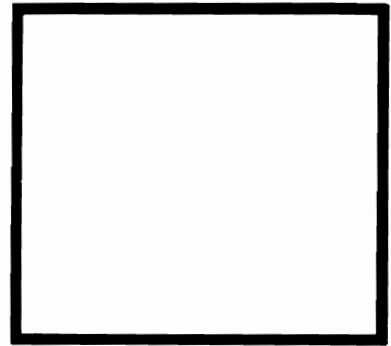
```
FD 30  
RT 90 FD 30  
RT 90 FD 30  
RT 90 FD 30  
END
```

If this is ok, press the RUN/STOP key.
If you need to correct something, read “Comments about Editing” in Appendix 1.

Now type BOX <return>.
LOGO has learned a new command, BOX.

1. Erase the screen (with DRAW).
Copy the drawing the turtle makes.
Type:

```
BOX BOX <return>
```

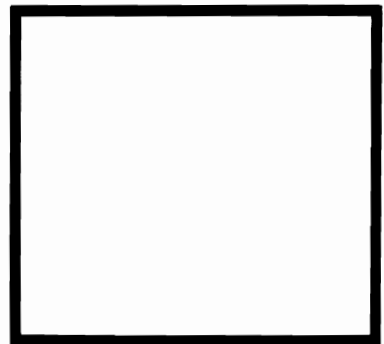


2. Erase the screen, then type:

```
REPEAT 4 [BOX]
```

Copy the drawing.

When using REPEAT be sure to use
brackets [] not parentheses .



3. Define two more procedures.

```
TO TRI1  
FD 30  
RT 120 FD 30  
RT 120 FD 30  
END
```

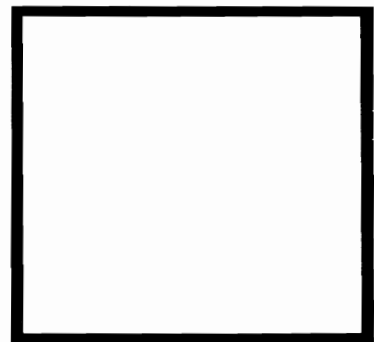
```
TO TRI2  
FD 30  
RT 135 FD 42  
RT 135 FD 30  
END
```

Press RUN/STOP to make LOGO accept the definitions. Then type either

or TRI1 TRI1
 REPEAT 2 [TRI1]

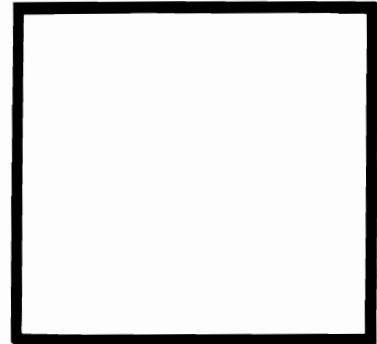
How many repeats give you a complete picture? Copy the picture in the box.

REPEAT ____ [TRI1]



4. How many repeats with TRI2 give a complete picture? Copy the picture in the box.

```
REPEAT ____ [TRI2]
```



5. Define a procedure using the new procedures.

```
TO SUPERTRI  
  TRI1  
  TRI2  
END
```

Experiment with

```
REPEAT ____ [SUPERTRI]
```

See if you can find out how many repeats of SUPERTRI make a complete design? _____ (This may be hard to tell.)

*If you have prepared a disk, you can save all these procedures for later use. Insert your personal disk and type:

```
SAVE "FIRST.SHAPES
```

See Appendix 2, to learn how to format a disk, save procedures, and read in procedures.



Lesson 5 Using Procedures

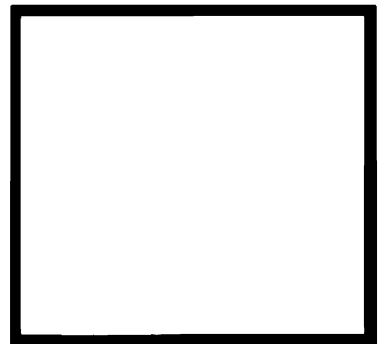
If you saved the procedures from Lesson 3, load your file by typing:

```
READ "FIRST.SHAPES
```

Otherwise, type in the procedures BOX, TRI1, and TRI2.

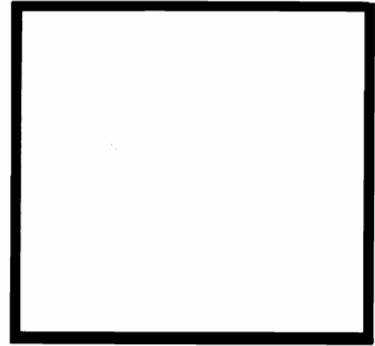
1. What does this draw?
Copy the drawing in the box.

```
BOX  
TRI2  
TRI2  
BOX
```



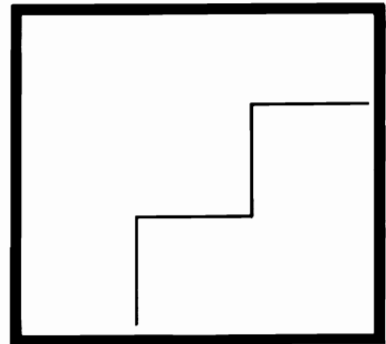
2. Put a turn before doing the steps in #1, so that the house is right-side up.

Write your program here.



3. Define a procedure STEP, which will draw this picture.
STEP should begin

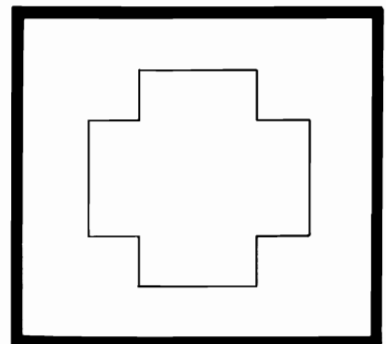
FD 30



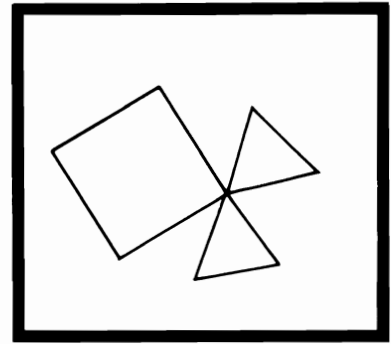
Try to determine which combination of STEP, BOX, TRI1 and TRI2 produces each of these pictures.

Write your answers here.

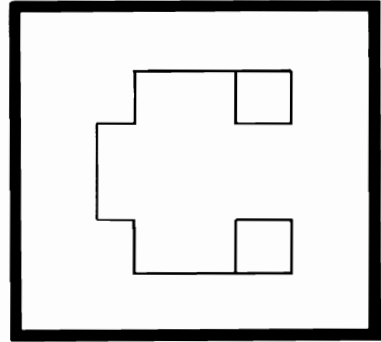
4.



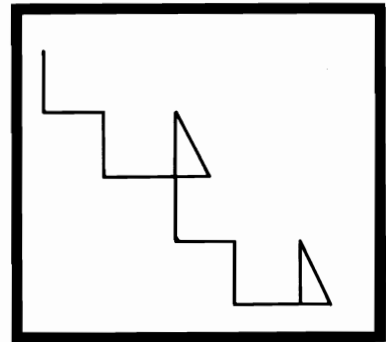
5.

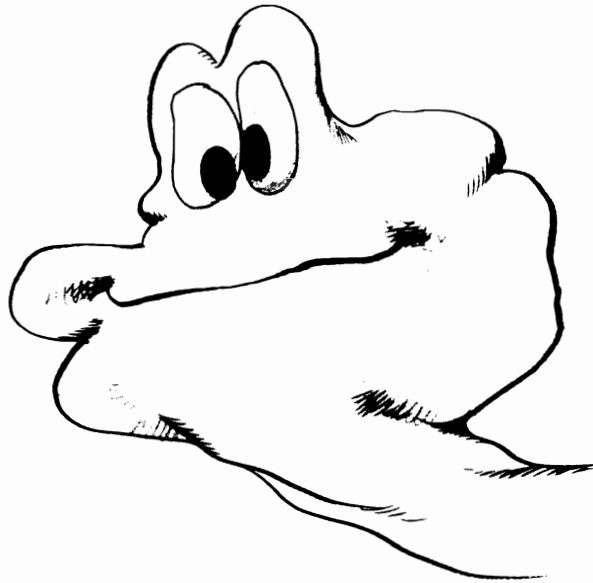


6.



7.





Lesson 6 Super Procedures

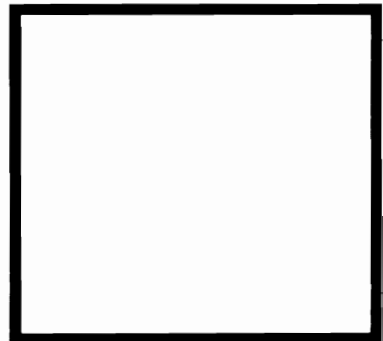
Again you need BOX, TRI1, and STEP.

Superprocedures have procedures as commands. Writing a superprocedure makes changes easy to make, without a lot of retyping.

1. Define DESIGN

```
TO DESIGN  
  REPEAT 4 [BOX RT 20]  
END
```

Copy the drawing the turtle makes.



2. Now type EDIT DESIGN <return>.

Change the 4 to 16:

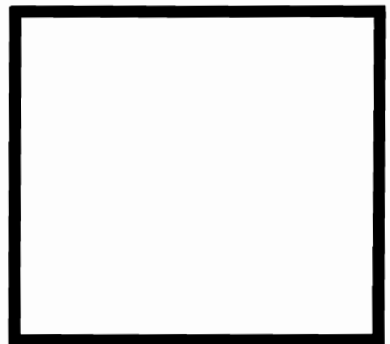
Move the cursor on top of the 4, type 16.

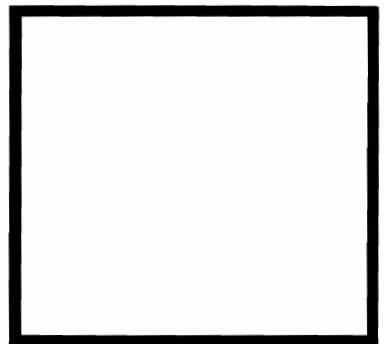
You should now have the number 164 showing.

Move the cursor to the space beyond the 4, press the delete key and press RUN/STOP to define the new DESIGN.

Consult Appendix 1, "Editing" if you have difficulty.

Try using different numbers of repeats. Record the two you like best.

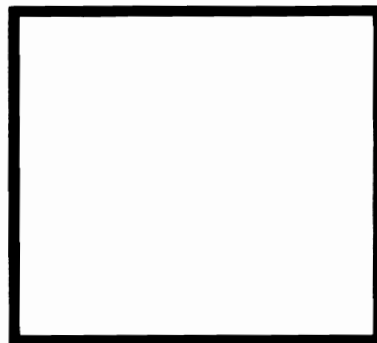




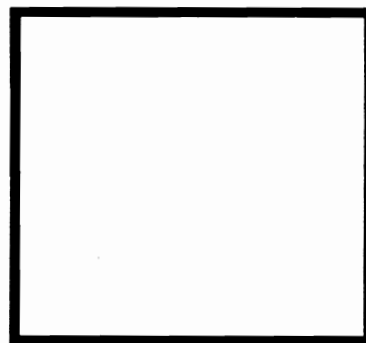
3. Define PICT

```
TO PICT  
REPEAT 3 [TRI1 FD 10 RT 60]  
END
```

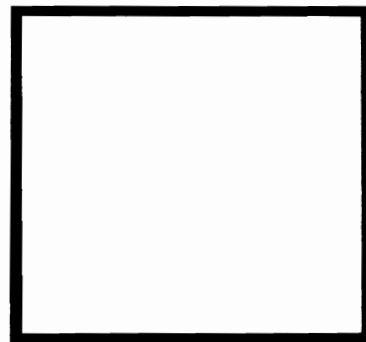
Run PICT, then change the number of repeats. How many repeats make a complete picture?



4. Use TRI2 instead of TRI1 in PICT. How many repeats will it take this time for a completed design?



5. Experiment using some or all of your procedures in a superprocedure. Record at least one that makes an interesting design.





Lesson 7

Erasing and Retracing

These problems use BK, backward, and PC, pen color.
BK is the opposite of FD.

PC is the color ink. If you type:

PC 0 the ink will be black.

PC 1 the ink will be white.

PC 2 the ink will be red.

etc.

If you type:

PC - 1 the pen is filled with ink eradicator.

Drawing over a line using PC - 1 will erase the line.

1. Try this. Press <return> after each line.

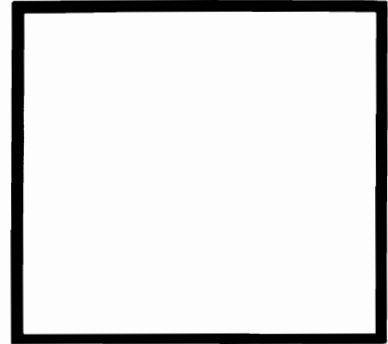
FD 50

PC - 1

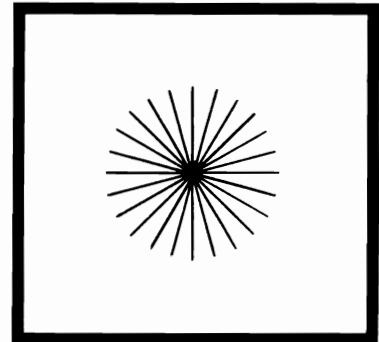
BK 40

2. BK can be used to draw several lines radiating from one point.
What does this draw?

```
TO MYSTERY  
REPEAT 9 [FD 50 BK 50 RT 15]  
END
```



3. Use the same idea to define FLOWER.
Make the “petals” 20 units long.



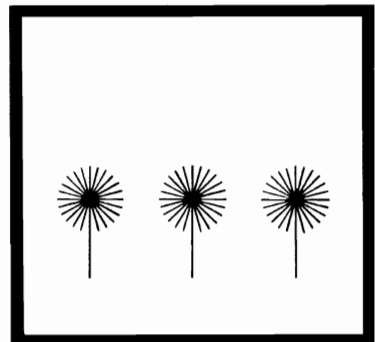
4. Define STEM which draws a green stem under your flower.
Then try FLOWER STEM.

```
TO STEM
```

Press <f 5> to show the picture on the full screen.
Press <f 3> to return to the split screen.

5. Write GARDEN which draws three flowers with stems, one red, one yellow and one blue.
Put in a green line for the ground.
Use PC - 1 to move invisibly.
The first command in GARDEN will make the background white.

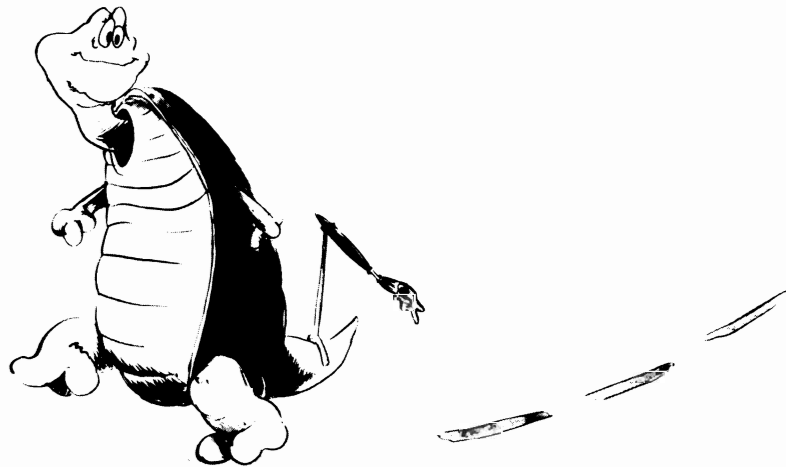
TO GARDEN
BG 1



6. Type DOUBLECOLOR
GARDEN

The lines are now twice as thick.
Type SINGLECOLOR to get back to normal.

10



Lesson 8

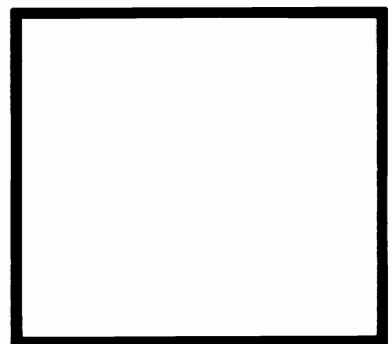
Pen Up and Pen Down

With PU (pen up) and PD (pen down) the turtle can move around without leaving a trail.

1. Try this.

```
TO KALEID  
REPEAT 6 [TRI1 PU FD 30 RT 60 PD]  
END
```

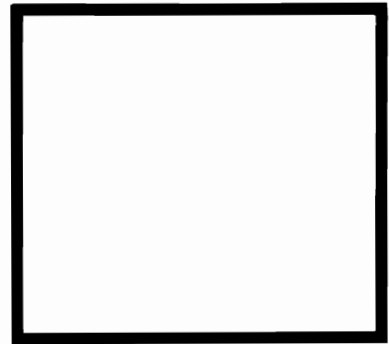
Copy the drawing the turtle makes in the box.



2. Change the repeat to:

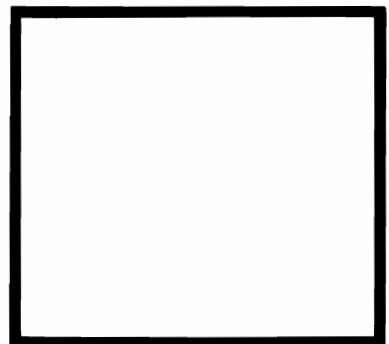
```
REPEAT 6 [BOX TRI1 TRI1 PU FD 80 RT 60 PD]
```

Copy the drawing the turtle makes in the box.

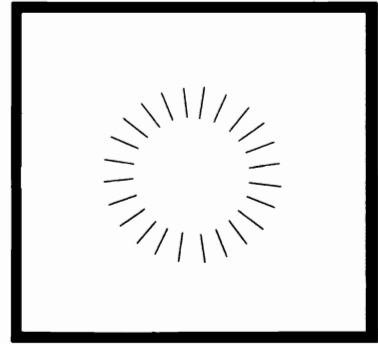


3. Experiment! Change the repeat to each of these. Copy the drawing you like best in the box.

```
REPEAT 6 [BOX TRI1 TRI2 PU FD 80 RT 60 PD]  
REPEAT 6 [BOX TRI2 TRI2 PU FD 80 RT 60 PD]  
REPEAT 6 [BOX TRI2 TRI1 PU FD 80 RT 60 PD]
```



4. A halo is a ^{PETALS} FLOWER with the center half of the petals not drawn. Define HALO, making both the gap and the ray 20 units long.

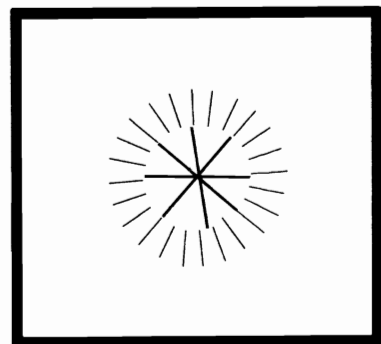


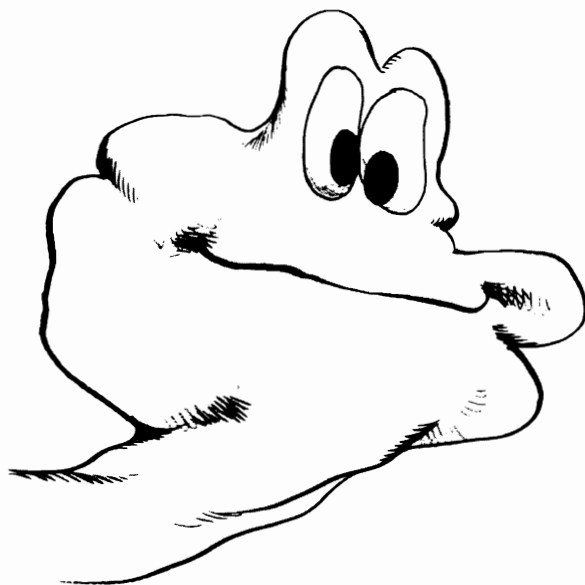
5. You can make another kind of blossom if you put a halo of one color around a flower of another color. (You might decide to give the outside HALO more rays, and a smaller angle.)

Your procedure should:

- turn the background white,
- select color,
- draw flower,
- change color,
- draw halo,
- make sure halo starts at edge of flower.

TO SUPERFLOWER





Lesson 9

Inputs Telling LOGO What Size You Want

A LOGO procedure can be told a number to work with. The procedure `SQ` will draw a square of any given size. `SQ 10` will be a square with side 10; `SQ 80` will be a square with side 80.

To write `SQ`, type

```
TO SQ :SIDE (be sure to put a space after SQ)
```

`:SIDE` is a variable. When a variable is referred to, its name always begins with a colon.

```
TO SQ :SIDE  
  REPEAT 4 [FD :SIDE RT 90]  
END
```

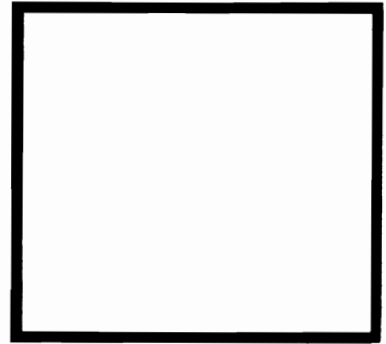
Run SQ 30. It looks the same as BOX, but something is different. If you don't see what it is, run:

```
SQ 30  
SQ 30
```

and compare the picture with the result of BOX BOX in Lesson 4.

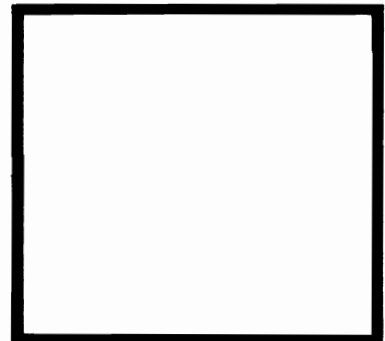
1. What does this draw?

```
SQ 10  
SQ 20  
SQ 30
```

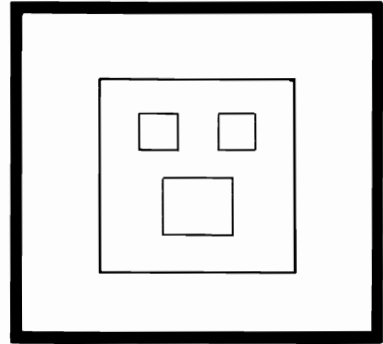


2. What does this draw?

```
SQ 80  
PU  
FD 30 RT 90  
FD 35  
PD  
SQ 10
```



3. Write a procedure that will make a face out of different squares.



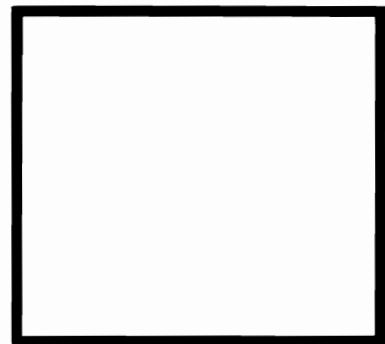
4. Here is a procedure that makes different triangles shaped like those made by TRI1.

```
TO TR :EDGE  
  REPEAT 3 [FD :EDGE RT 120]  
  END
```

The picture drawn by TR 30 looks the same as TRI1, but the turtle turns a final corner.

Try:

```
TR 10  
TR 20  
TR 30  
TR 40
```

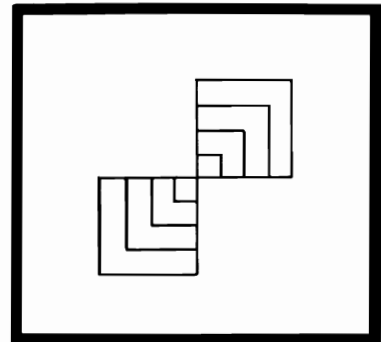


Copy the picture. Then, without erasing, type:

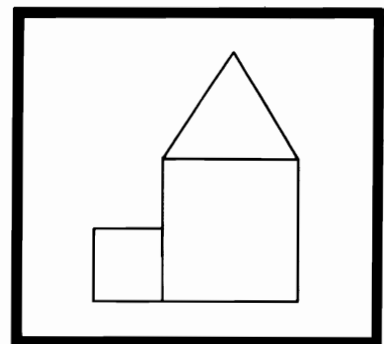
```
RT 180
```

and repeat TR 10 TR 20 TR 30 TR 40

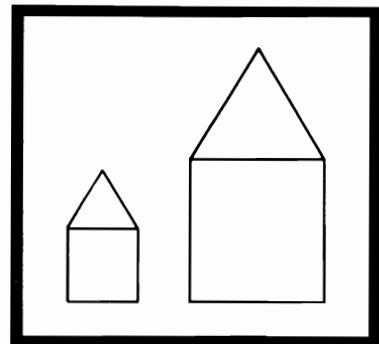
5. Write a procedure to draw this:



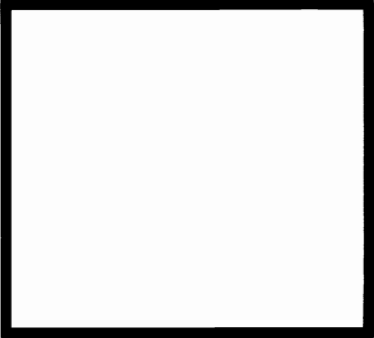
6. Use TR, SQ (combined with invisible movement around the screen) to draw a house and garage.

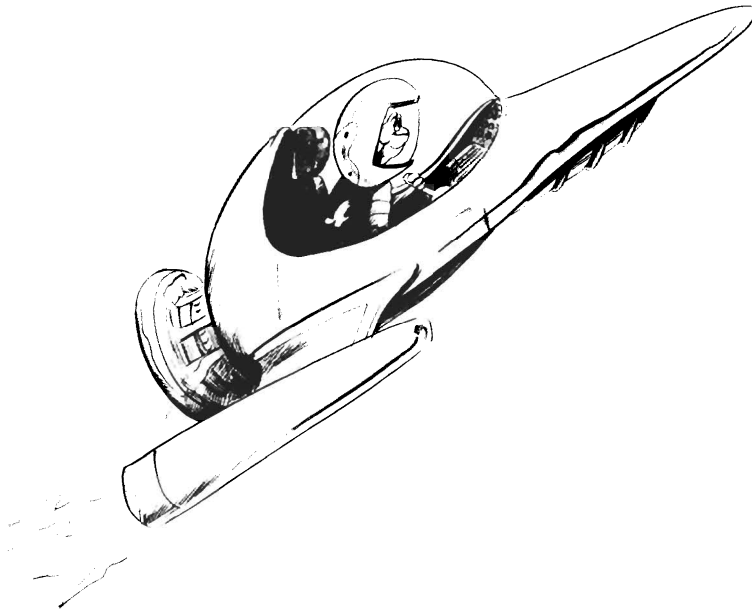


7. Write a procedure to draw the picture in the box. Be as efficient as you can!



8. Draw a picture of a telescope or a rocket in the box. Write a procedure to make LOGO draw your picture.





Lesson 10

Endless Repeats

If a procedure calls itself, it will repeat until you stop it by pressing CTRL G. A procedure which calls itself is a *recursive* procedure.

1. Try this.

```
TO ENDLESS  
  PRINT [FOREVER]  
  PRINT [AND EVER]  
  PRINT [ ]  
ENDLESS  
END
```

The third PRINT prints a blank line.

Try to write procedures (you'll need more than one) that print.

```
CLANCY LOWERED THE
```

```
BOOM  
BOOM  
BOOM
```

```
.  
.  
.
```

2. Try this.

```
TO FIGURE  
RT 115  
FD 100  
FIGURE  
END
```

Press <f 5> to see the entire screen.

3. A colored version of FIGURE uses a different color ink for each line.

```
TO FIGURE  
RT 115  
FD 100  
MAKE "C RANDOM 9  
PC :C  
FIGURE  
END
```

After a while the lines are so close that the computer does strange things with the colors.

You'll learn more about MAKE " in Lessons 22 and 26.
You'll learn more about RANDOM in Lesson 20.

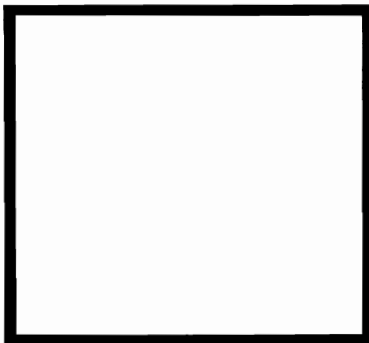
4. FIGURE drew a shape with side length 100 and turning angle 115.

PATTERN is like FIGURE except that both the side and angle are inputs.

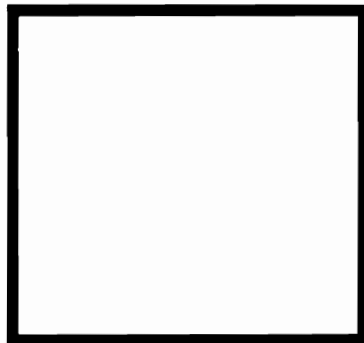
```
TO PATTERN :SIDE :ANGLE
FD :SIDE
RT :ANGLE
PATTERN :SIDE :ANGLE
END
```

(Try changing the pen color each time.)

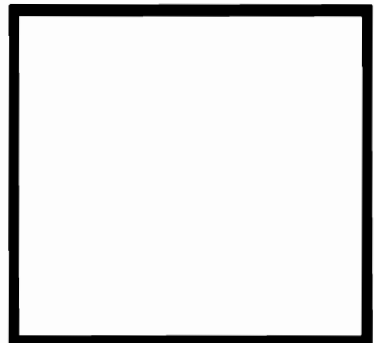
What happens for each of these?



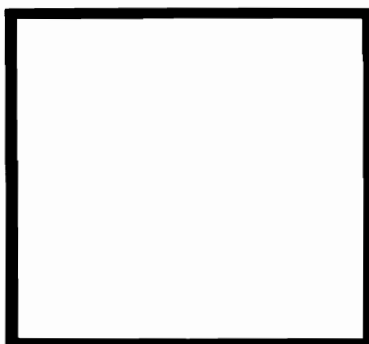
PATTERN 30 60



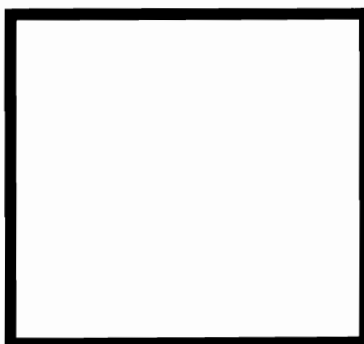
PATTERN 30 45



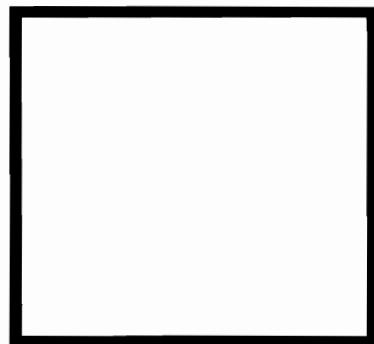
PATTERN 30 90



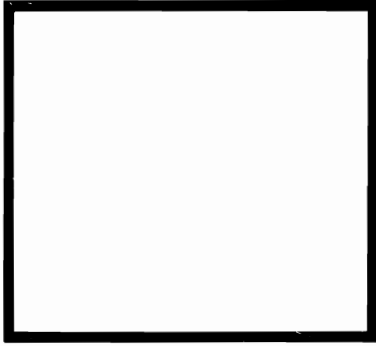
PATTERN 5 5



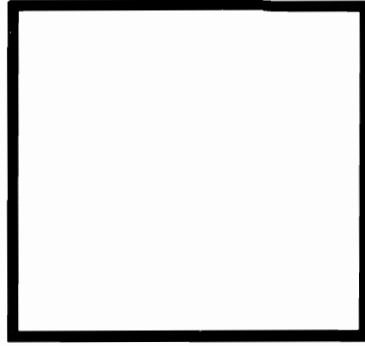
PATTERN 1 1



PATTERN 40 120



PATTERN 40 135



PATTERN 40 145

5. Try to find the angle that makes PATTERN draw a 5-pointed star.



6. Experiment with PATTERN. Try PATTERN 1000 300.
Try PATTERN 5000 300
Try other large numbers for the sides.
Try different angles.



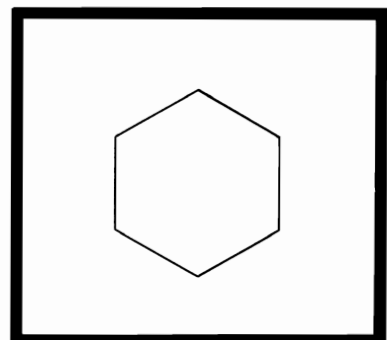
Lesson 11 Donuts and Sugar Donuts

(More Inputs and Recursions)

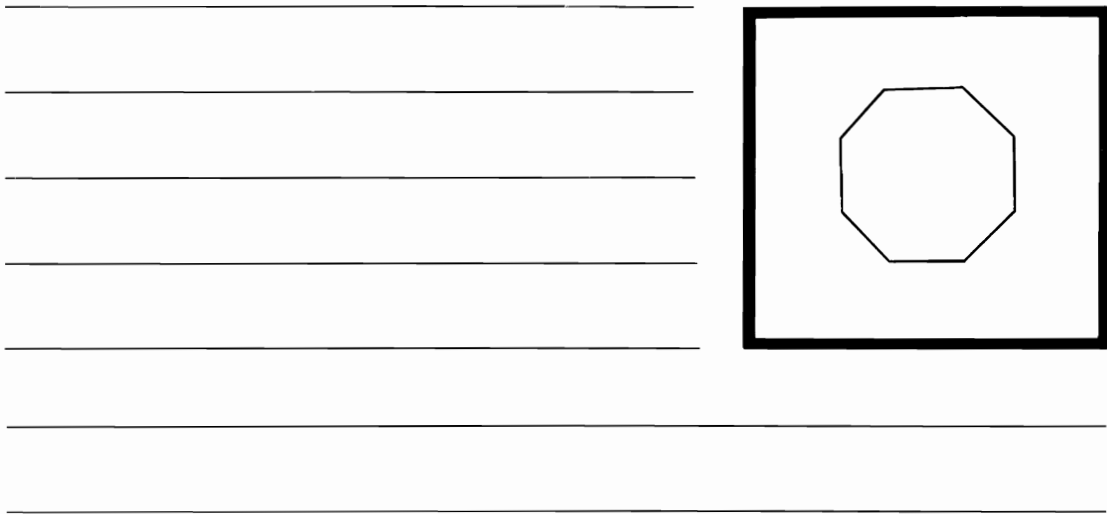
This project's goal is to draw a solid looking donut, sometimes with sugar sprinkles on it.

1. Write procedures to draw a hexagon and an octagon of variable sizes.

TO HEX :SIZE



```
TO OCT :SIZE
```



2. DONUT will draw the hexagon, move a little, and draw it again.

```
TO DONUT :SIZE  
  HEX :SIZE  
  FD 12  
  RT 10  
  DONUT :SIZE  
END
```

Try DONUT 3 and DONUT 10.

Change the move command in DONUT to FD 5 and try DONUT 10 again.

Change DONUT to use OCT instead of HEX.

Try DONUT 3 and DONUT 10 again.

3. It will be easier to experiment if you don't have to EDIT every time you want to change the distance moved and/or the angle. Rewrite the procedure DONUT so it takes three inputs: size, distance, and angle.

TO DONUT :SIZE :D :A

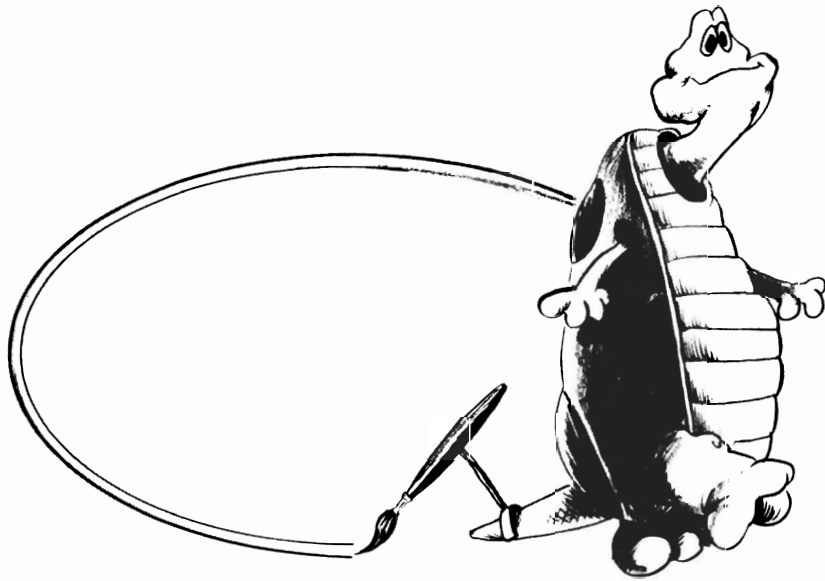
Experiment until you can draw a large, solid-looking donut. Write down each try. Circle the ones that gave good results.

What effect does each of these changes have on DONUT?

Changing the angle only? _____

Changing the distance only? _____

Changing the size only? _____



Lesson 12

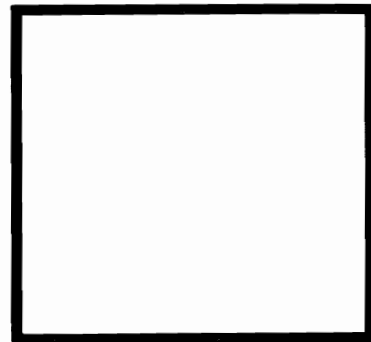
Circles

You'll need procedures SQ and TR.

An octagon (eight sides) is rounder than a hexagon (six sides). To make a circle, we draw figures with even more sides – sometimes 180 sides, sometimes even 360 sides.

1. The procedure EYEBALL will draw 3 circles.

```
TO EYEBALL  
REPEAT 360 [FD 1 RT 1]  
PC 7  
REPEAT 180 [FD 1 RT 2]  
PC 4  
REPEAT 360 [FD 1.3 RT 1]  
END
```



Copy the picture in the box. Draw a line from each REPEAT to the circle it generated.

2. To draw a circle of a given diameter (width) use:

```
TO CIRCLE :DIAM
HT
REPEAT 360 [FD :DIAM*3.14/360 RT 1]
ST
END
```

HT hides the turtle; ST shows the turtle. The turtle draws faster if it's hidden.

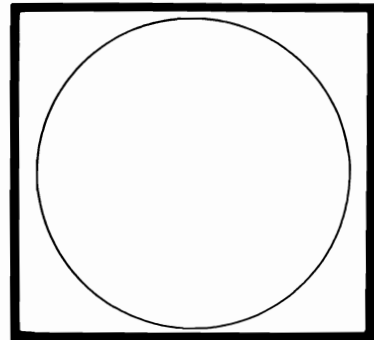
Try:

```
CIRCLE 40
CIRCLE 80
```

What is the largest the diameter can be before the circle goes off the screen at the top?

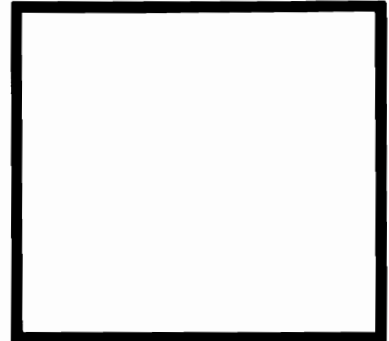
3. Write a procedure BIGCIRCLE to draw the largest possible circle on the screen. (The center will be at the middle of the screen.) BIGCIRCLE should move the turtle, then call CIRCLE.

```
TO BIGCIRCLE
```



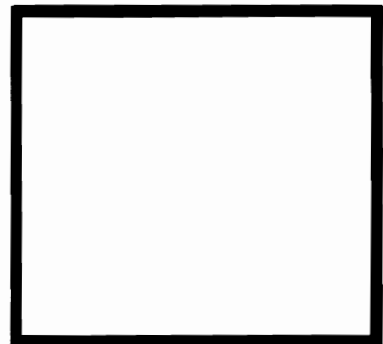
4. Try this program and copy the drawing in the box.

```
SQ 40  
CIRCLE 40
```



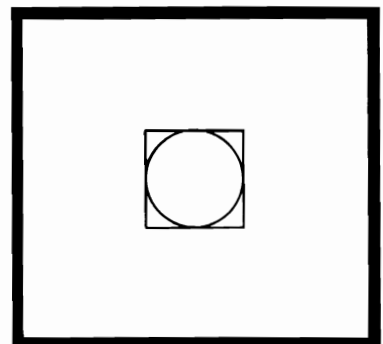
5. Try:

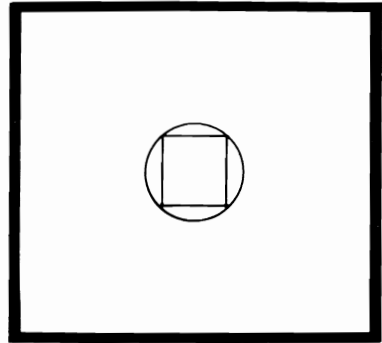
```
CIRCLE 40  
SQ 40
```



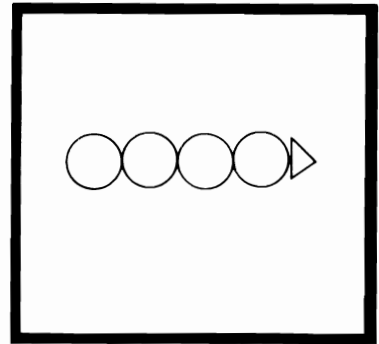
The procedures CIRCLE, SQ, and TR all return the turtle to the starting point, headed in the starting direction.

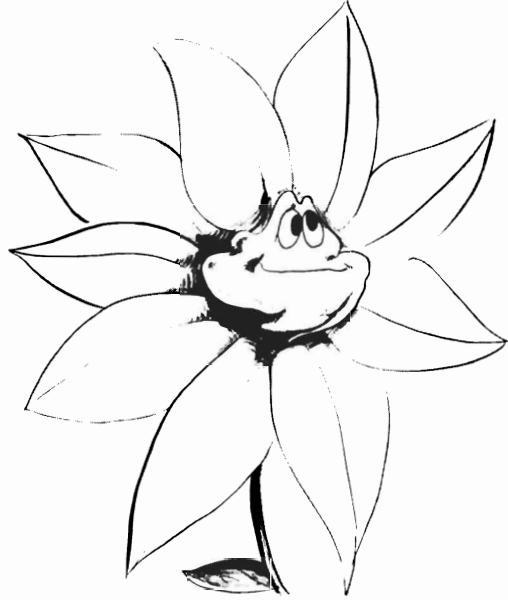
6. Write a procedure to draw each of these:





7. Write a procedure to draw a caterpillar.





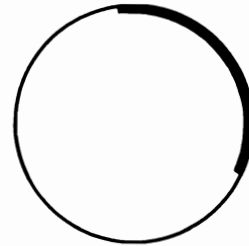
Lesson 13

Thirds of Circles

CIRCLE required 360 steps to complete. If only 120 steps are done, the result will be an arc which is one third of a circle.

1. Define the procedure ARC.

```
TO ARC :DIAM  
HT  
REPEAT 120 [FD :DIAM*3.14/360 RT 1]  
ST  
END
```



Try:

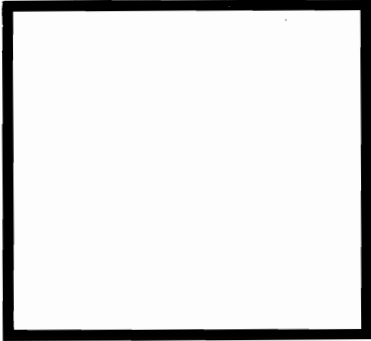
```
ARC 50 <return>  
ARC 50 <return>  
ARC 50 <return>
```

2. ARC can be used to make floral designs.

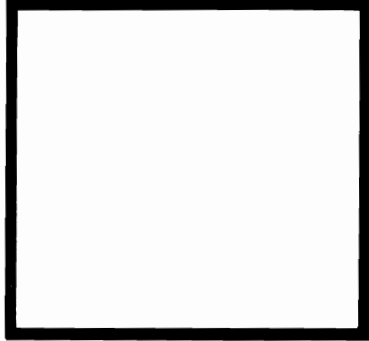
```
TO FLORAL :S :A  
REPEAT 100 [ARC :S RT :A]  
END
```

Use CTRL G to stop the procedure when the picture is complete.

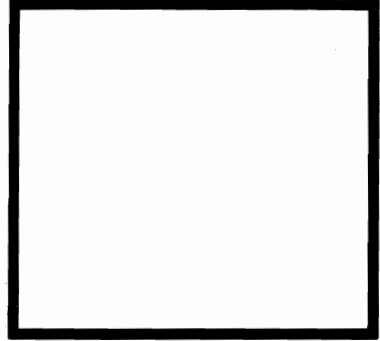
Try to copy the results of each of these. If it's too hard to copy, count the petals.



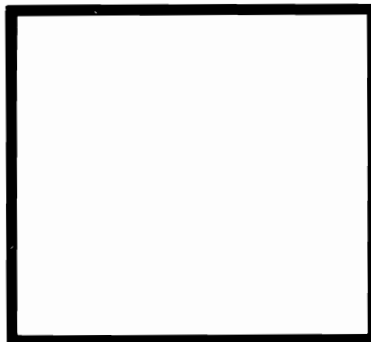
FLORAL 50 180



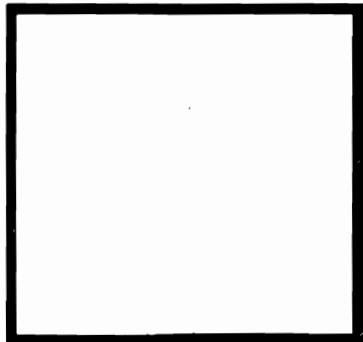
FLORAL 100 120



FLORAL 100 135



FLORAL 100 60

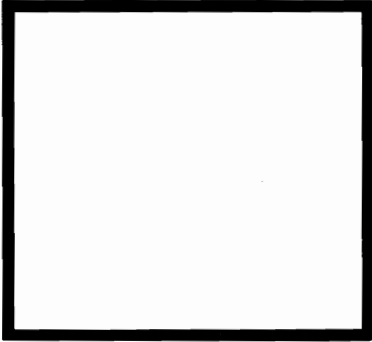


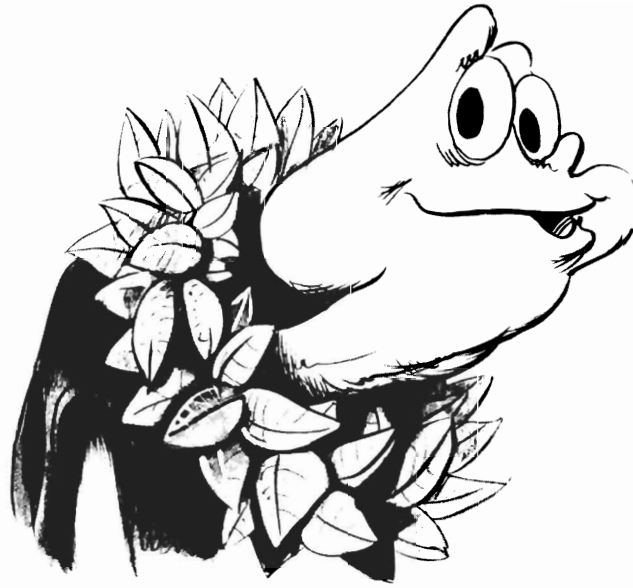
FLORAL 100 90

3. FLORAL repeats unnecessarily; for specific examples, you can lower the number of repeats.

Make a flower arrangement with green leaves and at least one colored flower. Try to draw a vase or pot, too.

List your ARRANGEMENT procedure here.
Make a copy of your drawing.

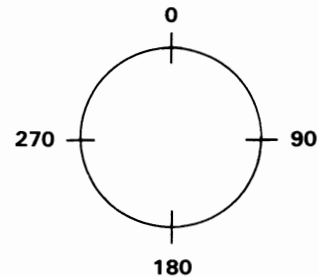




Lesson 14

Keeping Track of the Turtle

The turtle carries a compass and map with it. LOGO lets you use the compass in two ways.



HEADING tells the direction the turtle is heading. 0 is north (top of screen). Directions are in degrees clockwise from north.

SETH 0 aims the turtle in the direction of 0 (north).
SETH stands for SET Heading.

SETH 90 aims the turtle toward 3 o'clock.

In both PATTERN and FLORAL the design is complete when the turtle is headed in the starting direction.

If you're going to use the compass, the starting heading has to be specified before PATTERN is called. Here's how:

```
TO PATT :S :A
  SETH 0
  PATTERN :S :A
  PRINT [DONE]
  END
```

```
TO PATTERN :SIDE :ANGLE
  FD :SIDE
  RT :ANGLE
  IF HEADING = 0 STOP
  PATTERN :SIDE :ANGLE
  END
```

Try PATT 60 75 and PATT 60 45

Write a starting procedure for FLORAL. Call it FLOW. Change FLORAL so it stops when done.

1. Make a circle of stars. Your procedure will call PATT which will call PATTERN.

2. Make a wreath of green leaves.

3. Try to put a wreath of green leaves both inside and outside a wreath of flowers.

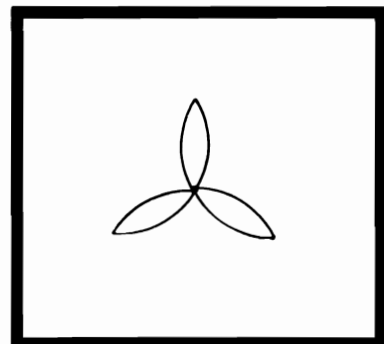


Lesson 15

Clover and Petals

FLORAL 100 120, in Lesson 13, produced a clover-like design. Define a procedure CLOVER to draw the design. CLOVER should not need any inputs.

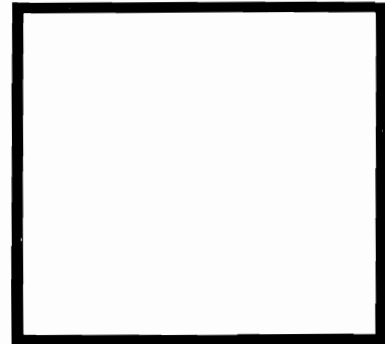
```
TO CLOVER  
REPEAT _____  
END
```



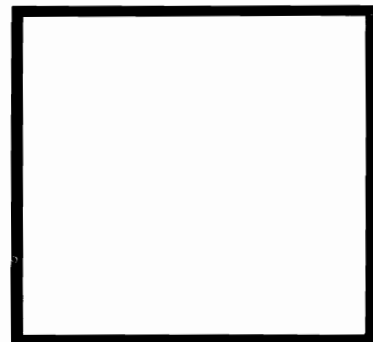
Mark on the picture where the turtle is when CLOVER begins and ends.

1. Experiment with CLOVER. First try these, then find another pleasing design. Draw your results in the box.

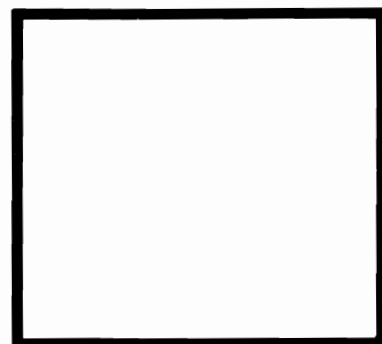
REPEAT 5 [CLOVER RT 72]



REPEAT 8 [CLOVER RT 45]



REPEAT [CLOVER RT 60]

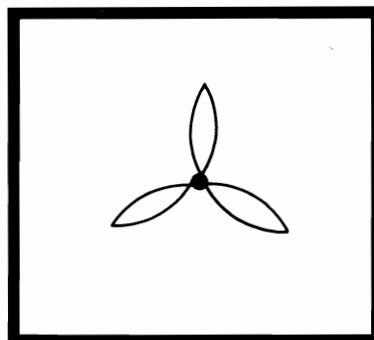


4. PETAL draws the same design as CLOVER except the starting point is at the middle of the flower.

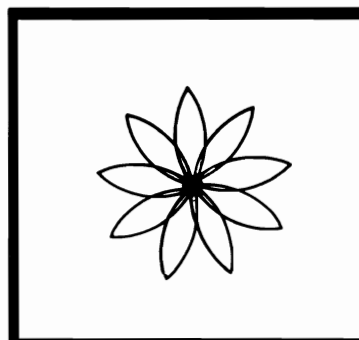
To write PETAL, you'll need TINYARC, which draws half of ARC 100.

```
TO TINYARC  
REPEAT 60 [FD 314/360 RT 1]  
END
```

Write PETAL. (What turns do you make?)



5. Write a procedure BLOSSOM to make a flower by spinning PETAL.



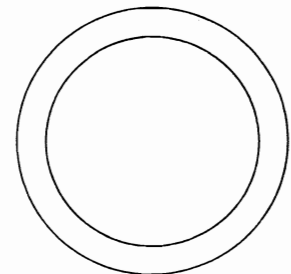


Lesson 16

Keeping on Track

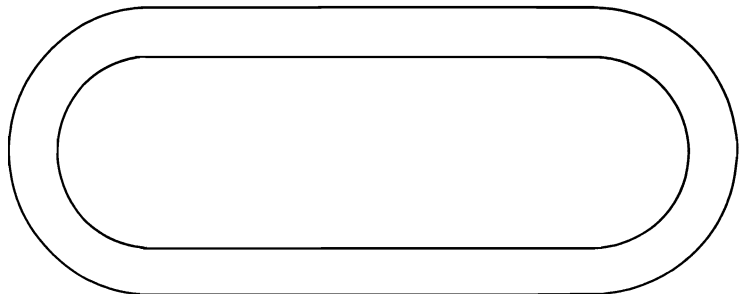
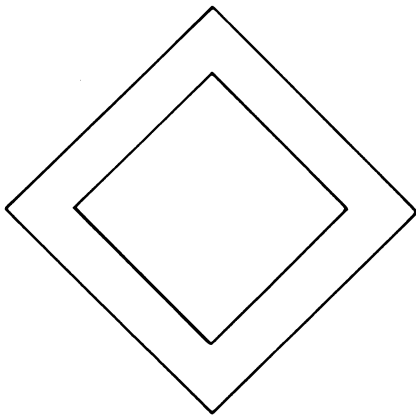
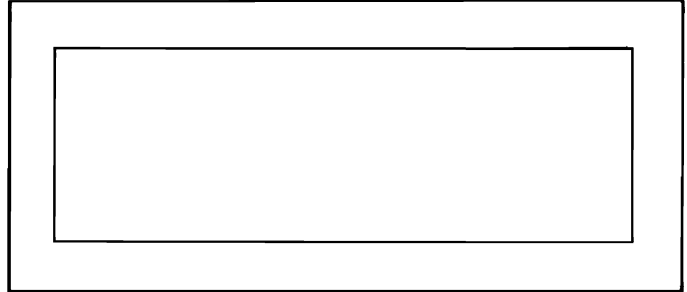
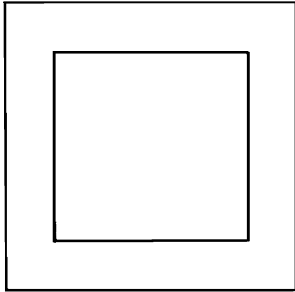
1. Write a procedure to draw the circular track and move the turtle around the track. The turtle should not leave a trail; it should keep going until CTRL G is pressed.

The outer circle has diameter 150.
The inner circle has diameter 120.



There are several “tracks” shown on this page. Choose a track; then write a procedure to draw the track and then move the turtle around it.

Draw an arrow to the track you are using.
Write your procedure here.

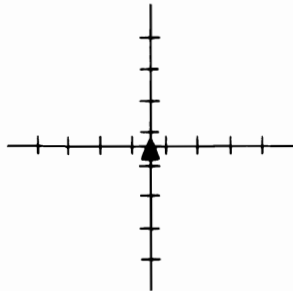




Lesson 17

Setting Things Down

The turtle is actually crawling over an invisible sheet of graph paper. When you type `DRAW` or `HOME` the turtle moves to the point in the center where the axes meet (the origin).

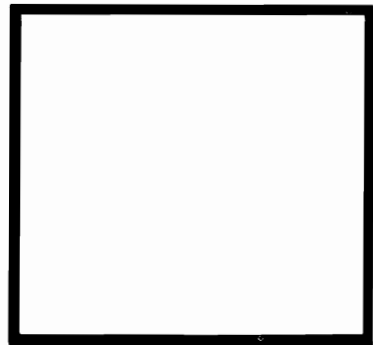


You can send the turtle to any point on the graph paper. The turtle will leave a trail.

Type DRAW. Then type each of these and watch the turtle.

```
SETXY 50 90  
SETXY 100 0  
SETXY 50 30  
SETXY 0 0
```

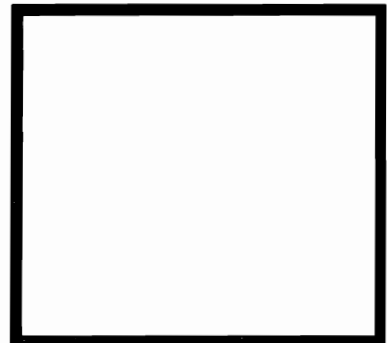
Copy what was drawn.



SETXY A B moves the turtle to the point (A,B). Unless the pen is up, a line is drawn.

1. Try this.

```
DRAW  
PU  
SETXY -150 0  
PD  
SETXY 0 (-125)  
SETXY 150 0  
SETXY 0 125  
SETXY -150 0
```

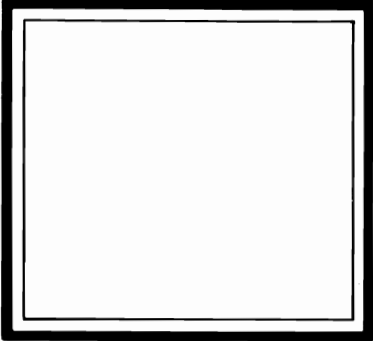


The diamond touches the (invisible) edges of the LOGO screen.

Warning: If the second coordinate of the point is negative it must be enclosed in parentheses.

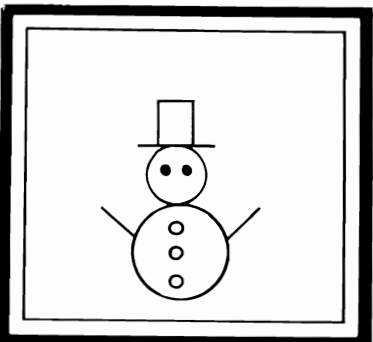
2. Using SETXY, write a procedure FRAME that frames the LOGO screen.

TO FRAME



3. Use FRAME, SETXY, SQ, CIRCLE, PU and PD to draw the snowman. Add arms when you've got the rest of him.

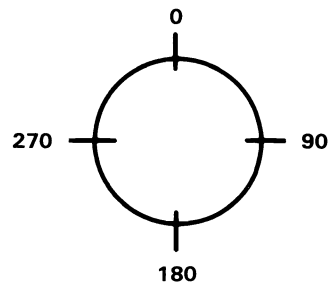
TO SNOWMAN





Lesson 18 Headings

You'll need SQ and TR.

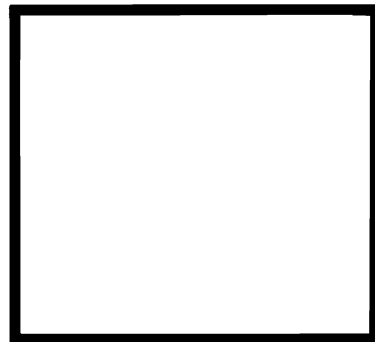


SETH is useful in turning the turtle before drawing a picture.

1. In HUT, SETH turns the turtle before drawing the roof.

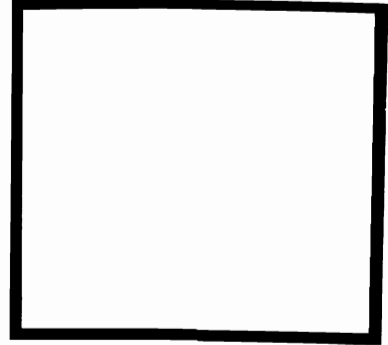
```
TO HUT :SIZE  
SQ :SIZE  
FD :SIZE  
SETH 30  
TR :SIZE  
END
```

Try HUT 60.



2. Before adding a door, you have to move the turtle. Add these lines to HUT.

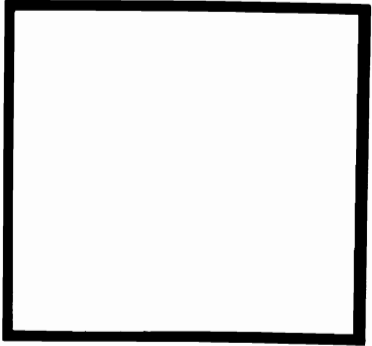
```
SETH 180  
FD :SIZE  
LT 90  
FD :SIZE/4  
LT 90
```



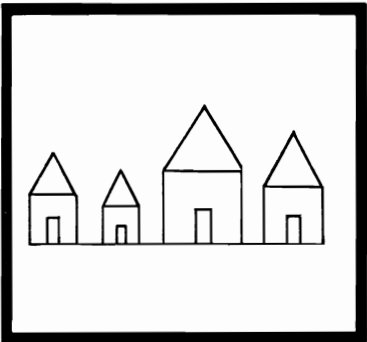
Try HUT 60 again. Notice the direction of the turtle.

Add lines to draw a door. The height of the door should be :SIZE/2. The width should be :SIZE/4.

Add a doorknob, chimney, or window if you like. Remember to state all distances in terms of :SIZE.

A simple black outline of a square, representing a door. It is positioned to the right of the first five lines of the writing area.

3. Use HUT to draw a city street. Write your procedure here.





Lesson 19 Christmas Tree

You'll need the new version of PATTERN (Lesson 14) and TR.

Try this.

```
TO XMAS  
PU  
SETXY -120 (-120)  
PD  
SETH 30  
TR 240  
PU  
SETXY 0 0  
PD  
SETH 0  
PATTERN 30 115  
END
```





Lesson 20

Random

You'll need PATTERN and CIRCLE.

1. The LOGO command (it's called a primitive) RANDOM makes LOGO think of a whole number.

Type RANDOM 6 <return>.

To repeat this several times, press SHIFT and , then return. Repeat at least 10 times. What were the largest and smallest numbers to appear?

_____ largest.

_____ smallest.

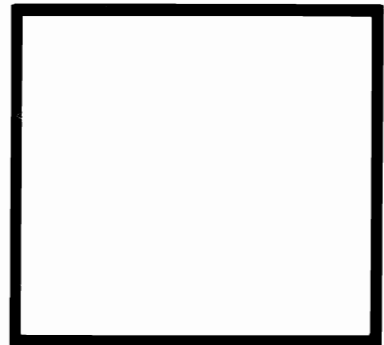
RANDOM N gives a whole number
between 0 and N-1 .

```
RANDOM 10 gives 0, 1, 2, . . ., or 9
RANDOM 4 gives 0, 1, 2, or 3.
(RANDOM 4) - 2 gives -2, -1, 0, or 1
```

2. Try to draw a box using random sides.

```
TO RBOX
REPEAT 4 [FD RANDOM 40 RT 90]
END
```

Run RBOX a few times. Copy the craziest boxes.



3. Try a random pattern. Keep the angle at 60, but allow the side to range from 0 to 50.

```
PATTERN RANDOM 50 160
```

To move the pattern to a random place on the screen, we need a random x-coordinate between -150 and 150, and a random y-coordinate between -125 and 125.

Since the highest values may cause wrapping, we'll keep the x-coordinate between -140 and 140, and keep the y-coordinate between -100 and 100.

```
TO SNOW
PU
SETXY (RANDOM 280)-140 (RANDOM 200)-100
PD
PATTERN RANDOM 50 160
END
```

To make a blizzard, turn SNOW into a recursive procedure.

TO SNOW

4. Write a recursive procedure BUBBLES. It should:
- Select a random spot on the screen.
 - Select a random color ink (use PC RANDOM 9).
 - Draw a circle of random diameter (up to 50).
 - Call itself.

Turn the background white before you start.

Interesting things happen to the screen if this is left running for 5-10 minutes. Copy your procedure here.

After this lesson, if you plan to use
RANDOM, type:

RANDOMIZE <return>

before you begin to work. Otherwise,
you'll always get the same sequence of
random numbers.



Lesson 21

Pinetrees

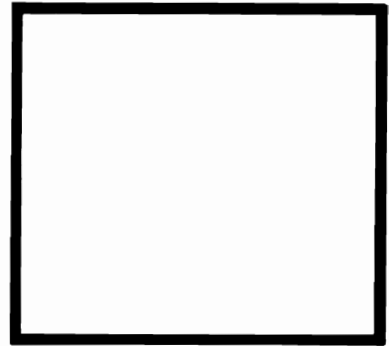
Besides the compass, which keeps track both of the turtle's heading and true north, the turtle carries a map. You can use the map both to find out where on the invisible graph paper the turtle is sitting, and to move it.

- | | |
|------------------------|---|
| <code>XCOR</code> | is the x-coordinate of the turtle's position. |
| <code>YCOR</code> | is the y-coordinate of the turtle's position. |
| <code>SETXY A B</code> | moves the turtle to (A,B). |
| <code>SETX A</code> | moves the turtle to (A,YCOR); only the x-coordinate is changed. This is useful when you want the turtle to move over. |
| <code>SETY B</code> | moves the turtle to (XCOR,B). Used to move the turtle up or down. |

1. PINETREE will make a tree with green branches and a brown trunk.

An outline of the procedure is:

```
TO PINETREE
start at top
SETH 220
BRANCHES
go to top
SETH 140
BRANCHES
TRUNK
END
```



BRANCHES draws a set of eight parallel lines:

```
TO BRANCHES
turn ink green
REPEAT 8 [FD 30 BK 30 SETY YCOR-8]
END
```

Write out the full procedure PINETREE.
Also write the procedure for TRUNK.

TO PINETREE

TO TRUNK

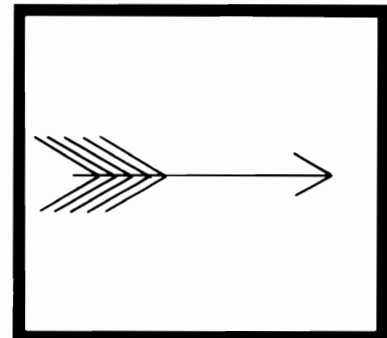
2. Make a forest of 20 trees. Select the tops at random, but be sure they are tall enough on the screen, but not high enough so the trees wrap around vertically.

Add a HUT for a country scene.

TO FOREST

3. Write a procedure to draw an arrow with red top feathers and blue bottom feathers.

TO ARROW





Lesson 22

Repeats With a Difference

Not only can LOGO procedures call themselves, but they also can *change* the inputs.

1. Define the procedure SHRSQ. SHRSQ is like SQ except for the next to last line, which shrinks the side by 10 before calling itself.

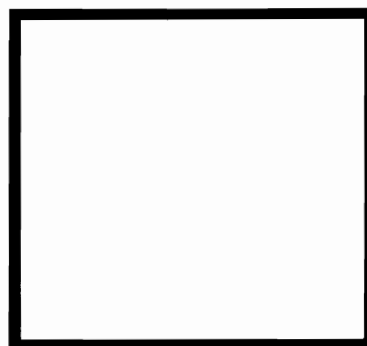
```
TO SHRSQ :SIDE  
  REPEAT 4 [FD :SIDE RT 90]  
  MAKE "SIDE :SIDE - 10  
  SHRSQ :SIDE - 10  
END
```

Run SHRSQ 60; stop with CTRL G when it begins to wrap around.

Look at what happened. First it drew squares of side 60, 50, 40, . . . , 10, 0, -10, -20, . . .

FD - 10 is the same as BK 10 so the squares are drawn in the opposite direction.

Run SHRSQ 60 again. Let it go until there's no change. Copy the picture in the box.

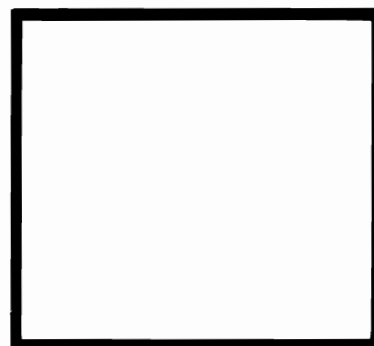


To stop SHRSQ from calling itself forever, it should test the value of :SIDE.

```
TO SHRSQ :SIDE  
  REPEAT 4 [FD :SIDE RT 90]  
  IF :SIDE < 0 STOP  
  SHRSQ :SIDE - 10  
END
```

Try SHRSQ 60 again. Then move the IF line to a better place.

Copy the final version of SHRSQ here.



2. Write GRSQ, a procedure to enlarge a square up to side 100.

TO GRSQ :S

3. Change :SIDE -10 to :SIDE -1

Try SHRSQ again.

Try different colors.

4. Try to shrink and enlarge the triangles. Use SHTR and GRTR as names.

TO SHTR :S

TO GRTR :S

5. If SHTR and GRTR have no stopping test, do they eventually look the same?



Lesson 23

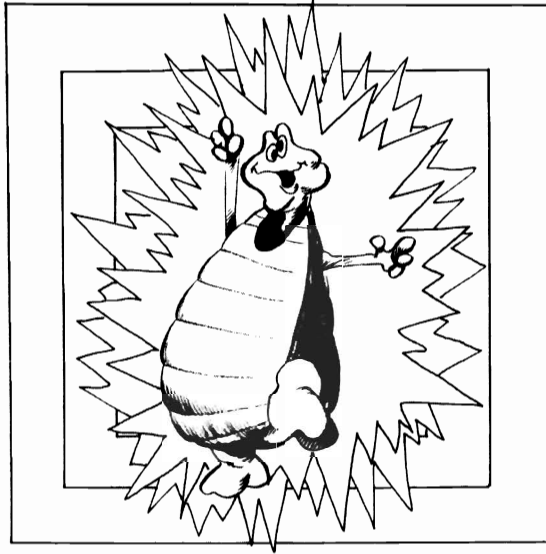
Solid Colors

1. If you were to build the hut in Lesson 18, using SHRSQ and SHTR from Lesson 22, you would have “painted” the hut.

Try to draw a hut with a solid red roof and blue walls.

List all procedures (including SHTR and SHRSQ) here.

If possible, draw a street of different colored, different sized huts.



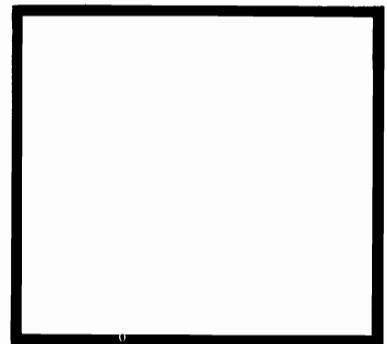
Lesson 24

Using Set to Draw

LOGO draws faster if the turtle is hidden. It draws much faster if straight lines are drawn using SETXY rather than FD.

1. SETSQ draws a square with side twice the input. The center of the square is the HOME position. Try to figure out how the square will be drawn, before trying the program.

```
TO SETSQ :S
  PU HT
  SETXY :S :S
  PD
  SETY - :S
  SETX - :S
  SETY :S
  SETX :S
END
```



Copy the square. Number the sides in the order they are drawn.

2. Because SET is faster than FD, and because it doesn't retrace lines, SETSQ will make a solid-color square much faster than SHRSQ. Write a procedure to make a solid square using SETSQ. Try SOLSQ 30.

```
TO SOLSQ :S
```

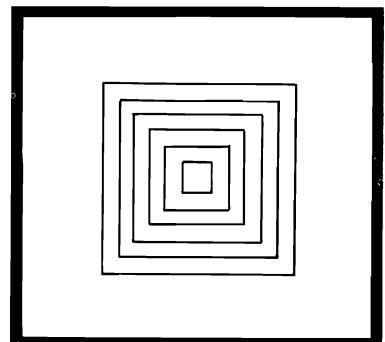
3. SETSQ is limited to a square centered at the origin. Make a succession of solid squares, largest one first, in random colors to get a quilt pattern. Decrease :T by 10 each time MASTER is called:

```
TO MASTER :T
```

```
IF _____
```

```
_____  
SETSQ :T
```

```
_____  
END
```



4. Without the recursion, SETSQ draws one square at a time. Delete the recursion line from SETSQ. A procedure that erases a square before drawing the next smaller one will make it look as if the square is shrinking.

```
TO SHRINK :N
IF :N < 1 STOP
PC -1
SETSQ :N+1
PC 7
SETSQ :N
MAKE "N :N-1
SHRINK :N
END
```

5. Write a procedure BLAST, that is the opposite of SHRINK. BLAST starts with a small square, which appears to become larger and larger.

```
TO BLAST
```



Lesson 25

Rings

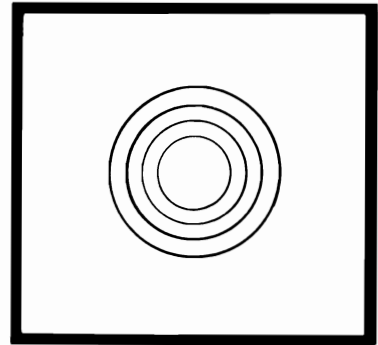
You'll need CIRCLE from Lesson 12.

Sometimes it is more convenient to specify a circle by its center and radius. That way, LOGO will always draw the same circle, no matter which direction the turtle is heading.

RING takes three inputs, the coordinates of the center and the radius.

```
TO RING :XCEN :YCEN :RAD
  PU
  SETXY :XCEN :YCEN
  FD :RAD
  RT 90
  PD
  CIRCLE 2*:RAD
END
```

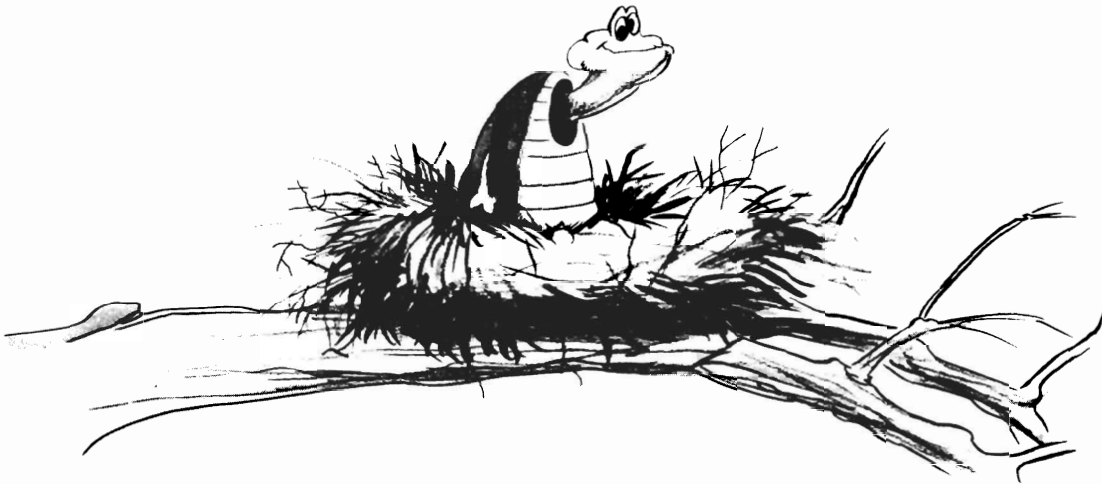
1. Use RING to make a nest of four circles. Each should have a radius 10 more than the last. Draw the smallest circle first. Write your procedure here.



2. Go the other way: draw nested circles, the largest first. Reduce the radius by 10 each time. Keep going until the radius is < 1 . Write your procedure here.

3. Write a procedure to shrink and blast circles.

4. Try to draw a solid circle by using the procedure from problem 2 with each radius 1 less than the previous one.



Lesson 26

Defining Variables

You'll need SQ from Lesson 9.

When a variable is *used* in a LOGO procedure its name is preceded by a colon.

Examples: :SIZE :EDGE :ANGLE

When a procedure *defines* a variable, MAKE " is used.

Examples: MAKE "X 6
MAKE "N 2*:X

1. Try this one.

```
MAKE "DOG 50
SQ :DOG
MAKE "HELLO [HI THERE]
PR :HELLO
PR [HELLO]
```

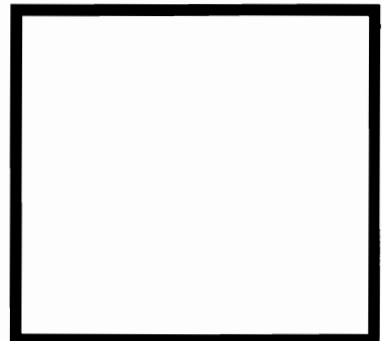
2. Write a procedure SUM which has two inputs, :X and :Y. SUM defines a new variable :S to be the sum of :X and :Y. After defining :S, SUM should print it.

```
TO SUM :X :Y
```

3. MAKE " is used when you want to change the value of a variable during a procedure. DEMO doubles :N.

```
TO DEMO  
MAKE "N 40  
PR :N  
MAKE "N 2*:N  
PR :N  
END
```

4. Change DEMO so that instead of printing :N each time, it draws SQ :N.



5. NESTS is a procedure which calls itself, but changes :N every time.

```
TO NESTS :N
  IF :N = 0 STOP
  SQ :N
  MAKE "N :N-10
  NESTS :N
END
```

See if you can predict what NESTS 80 will do.

6. Before NESTS calls itself again, move the turtle. Insert these lines before NESTS :N.

```
FD 10
RT 90
FD 10
```




Lesson 27

Keep Counting

The procedure COUNTER will keep going until CTRL G is pressed.

```
TO COUNTER :N  
PR :N  
MAKE "N :N+1  
COUNTER :N  
END
```

Try COUNTER 1.
Change COUNTER so it counts by fives:

2. To make COUNTER stop, put in a test:

```
IF :N > 100 STOP
```

Rewrite COUNTER to count to 1000 by 3s.

3. Define BLASTOFF. It should count from 10 to 0.
Pause between each number (use REPEAT 500 [] to make it do nothing 500 times).

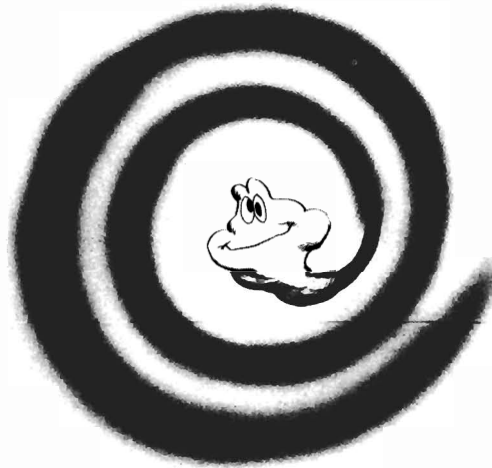
When the count reaches 0, something fancy should happen.
Change the screen color several times.

Record your final BLASTOFF procedure—and any procedures it uses.

4. Write a reaction tester. Tell the user to be ready to press CTRL G when the numbers appear.

Use a pause of random length before printing the numbers.

Write your procedure here.



Lesson 28

Spirals & Angle Spirals

SPIRAL is like PATTERN (Lesson 10), except that each time the turtle turns, the side gets bigger.

The inputs are :A the turning angle and :R the number of repeats.

The length of the side, :S, is a variable. When :S is given a value or changed, MAKE "S is used.

The procedure SPIRAL is:

```
TO SPIRAL :A :R
  MAKE "S 1
  REPEAT :R [FD :S RT :A MAKE "S :S + 2]
  END
```

(Use CTRL G to stop SPIRAL.)

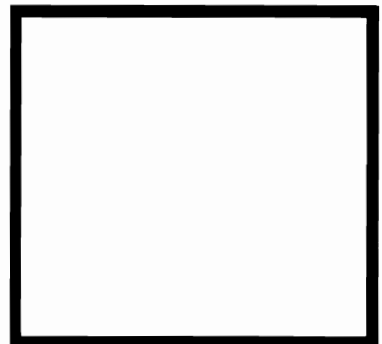
1. Some spirals look, at first glance, like polygons. Try these inputs for SPIRAL:

A	R	Looks like a
120	50	_____
144	50	_____
90	50	_____
160	50	_____

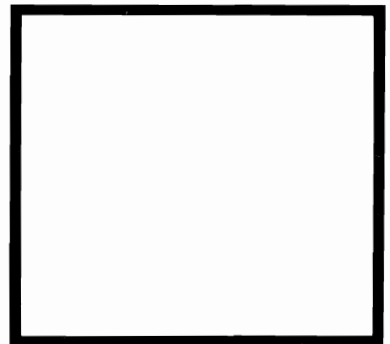
2. If the angle is changed slightly from a “regular polygon” spiral, the result looks like a spiral galaxy.

Try these. Copy the curved lines that seem to appear.

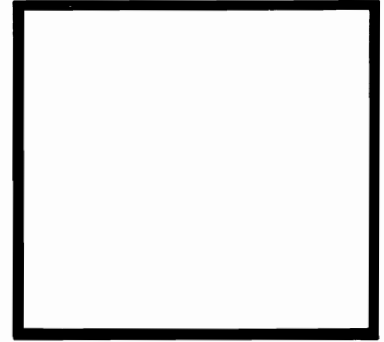
SPIRAL 145 68



SPIRAL 91 60



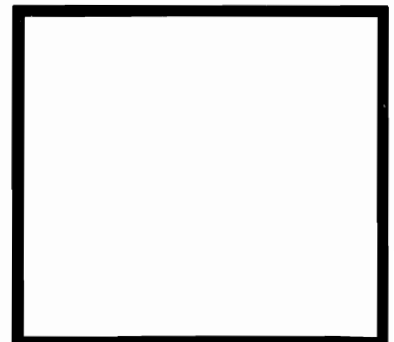
SPIRAL 121 60



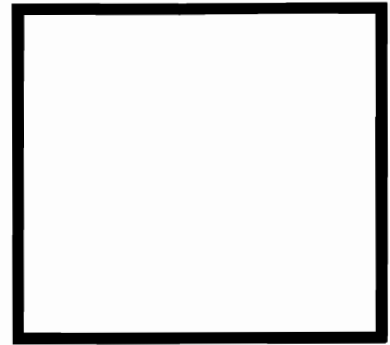
3. Write a procedure UNSPIRAL which is like PATTERN, except that the angle increases by 5 each time the turtle turns.

TO UNSPIRAL :S :R

4. Try UNSPIRAL 10 300.
Copy the picture in the box.



5. Change the amount the angle increases to 6. Try UNSPIRAL 10 300 again.
Copy the picture in the box.

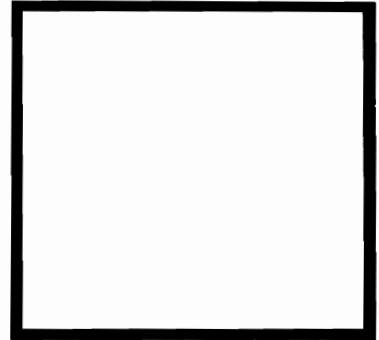
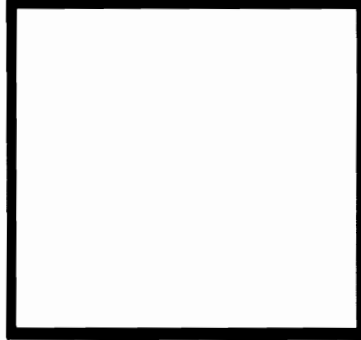
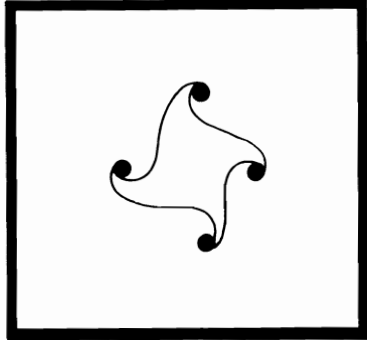


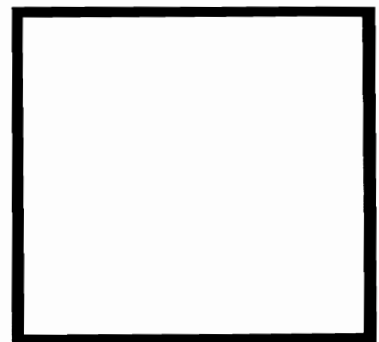
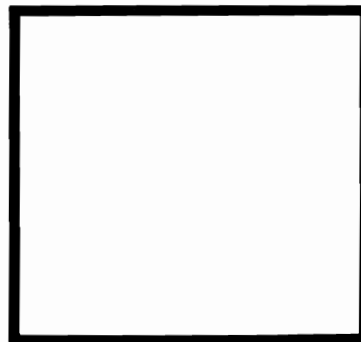
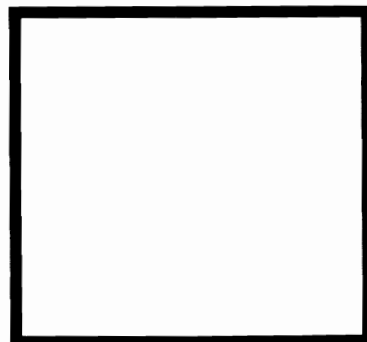
6. Since changing from 5 to 6 made such a difference, rewrite UNSPIRAL so the angle increase :A| is an input.

TO UNSPIRAL :S :R :A|

7. Try angle increases of 2, 4, 8, 9, 10, at least. Start with side 10 and 1000 repeats. You may have to use a smaller side and/or use SETXY at the beginning to fit the unspiral on the screen.

Copy the pictures here. Record the side length, angle increase, and any other changes you made under each picture.







Lesson 29

Pressing a Key

You'll need MASTER and SETSQ.

You can ask LOGO whether a key is being pressed, and you can tell LOGO to wait until a key is pressed.

RC or READCHARACTER means wait until a key is pressed. Then RC will have the value of the key which was pressed.

Here are two examples of using RC:

1. Define FLASHY and run it.

```
TO FLASHY  
  BG RC  
  FLASHY  
END
```

Press `<f 5>` for a full screen. Press the number keys. Press non-number keys.

Change FLASHY so that pressing 1 turns the screen black, 2 turns it white, 3 turns it red, etc.

TO FLASHY

2. Define VOWEL and run it.

```
TO VOWEL
PR [PRESS A VOWEL KEY]
MAKE "ANS RC
IF MEMBER? :ANS "AEIOU PR [RIGHT] ELSE PR [WRONG]
PR [ ]
END
```

Instead of a list of vowels, type in a secret word. Let a friend guess your word by trying different letters.

Using RC stops the action until a key is pressed. RC? checks whether a key is being pressed. If no key is being pressed, the procedure continues.

3. Edit MASTER. Insert a new second line.

```
TO MASTER :T  
IF RC? BG RC  
IF :T < 10 STOP  
etc.
```

Run MASTER 100; press number keys while it's running.

4. The test

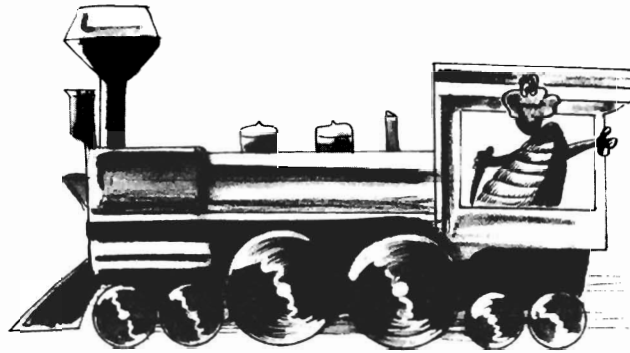
```
IF RC? BG RC
```

can be put inside a REPEAT.

Try this.

```
REPEAT 100 [FD 40 RT 117 IF RC? BG RC]
```

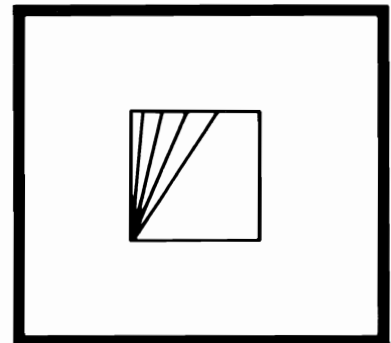
Then change another one of your procedures to allow you control over the background color while it's running.



Lesson 30

Solid Shapes

Imagine the turtle drawing a box or something, but everytime it goes forward a step, it then hops to the lower left-hand corner and back, leaving a trail.



The result will be a solid shape.

The procedure SFD (for solid forward) will make the turtle move forward a step—hop to a fixed point and then hop back. When we draw the box, we'll use SFD :N instead of FD :N.

Before drawing the box, the coordinates of the fixed point :XF and :YF, must be defined. SFD will define :XT and :YT, the current point on the line being drawn. The turtle will move FD a step, hop to (:XF,:YF), hop back to (:XT,:YT).

1. Here's the procedure for SFD:

```
TO SFD :N
IF :N = 0 STOP
FD 1
MAKE "XT XCOR
MAKE "YT YCOR
SETXY :XF :YF
SETXY :XT :YT
MAKE "N :N-1
SFD :N
END
```

A procedure to draw a solid green box of side :S is:

```
TO SBOX :S
MAKE "XF XCOR
MAKE "YF YCOR
PC 5
REPEAT 4 [SFD :S RT 90]
END
```

Try SBOX 40. Change SBOX to hide the turtle while coloring in the box.

Define a new set of procedures to draw solid shapes.

```
STR
SCIR
SPATTERN
SREC—to draw a solid rectangle of given height and
width.
```

Make both the size and the pen color an input in your procedures.

TO STR :SIDE :C

TO SPATTERN :S :A :C

TO SCIR :DIAM :C

TO SREC :H :W :C



Lesson 31

Flags

Define a procedure to draw a solid-color rectangle. Inputs should be the width, height, and color of the rectangle.

```
TO SREC :H :W :C
```

Use SREC and your other solid procedures to draw a flag from each group.

1. Group 1

ITALY

GREEN		
	WHITE	
		RED

FRANCE

BLUE		
	WHITE	
		RED

CHAD

BLUE		
	YELLOW	
		RED

2. Group 2

MADAGASCAR

WHITE	RED
	GREEN

COLOMBIA

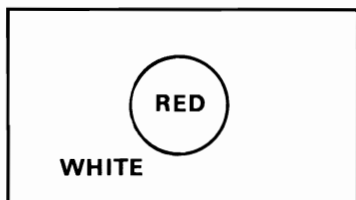
YELLOW
BLUE
RED

UNITED ARAB
EMIRATES

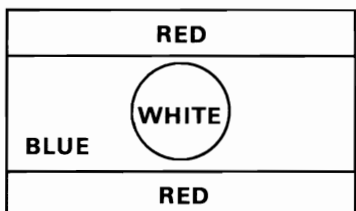
RED	GREEN
	WHITE
	BLUE

3. Group 3

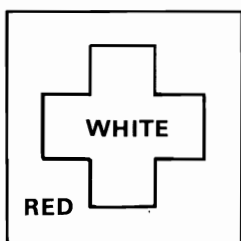
JAPAN



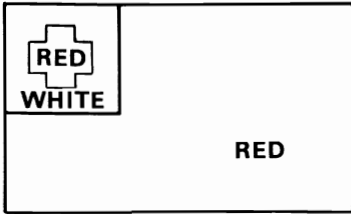
LAOS



SWITZERLAND

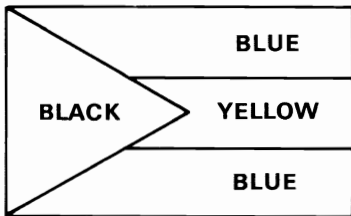


TONGA

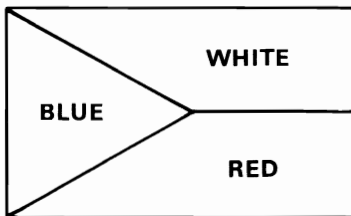


4. Group 4

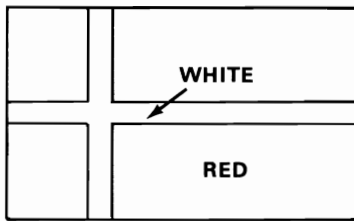
BAHAMAS



CZECHOSLOVAKIA



DENMARK

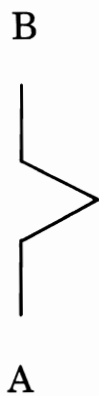




Lesson 32 Fancy Forwards

SFD from Lesson 29 was a replacement for FD. You can define other types of FORWARDSs, other ways to move from A to B.

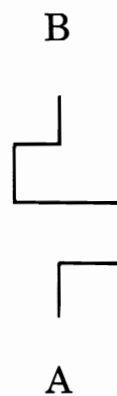
Here are some examples:



PFD



BFD



JFD

Call them:

Where,

PFD stands for pointed forward.

BFD stands for box forward.

JFD stands for jog forward.

If the distance from A to B is :N, the length of each segment of PFD and BFD is :N/3. JFD is more complicated.

Write a procedure for each of the new forward paths.

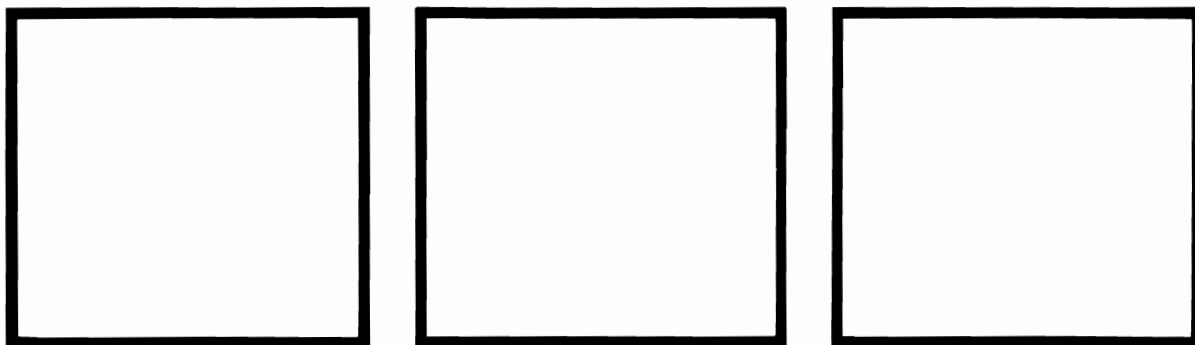
```
TO PFD :N
FD :N/3
RT 60
FD :N/3
LT 120
FD :N/3
RT 60
FD :N/3
END
```

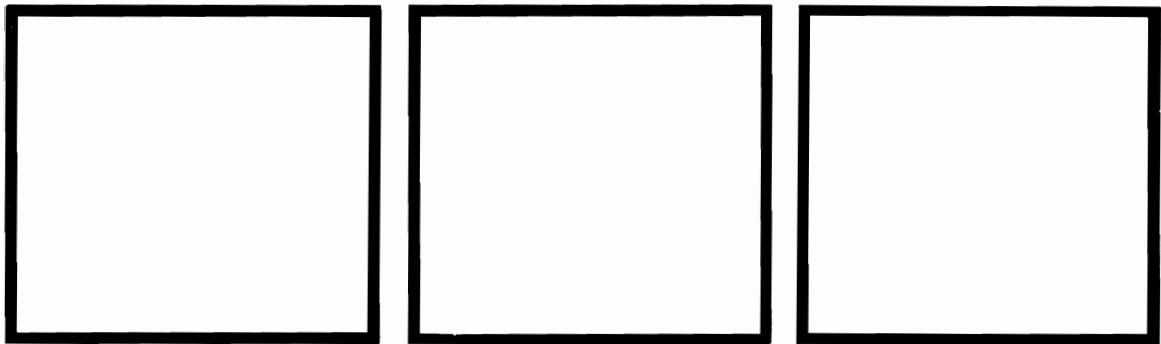
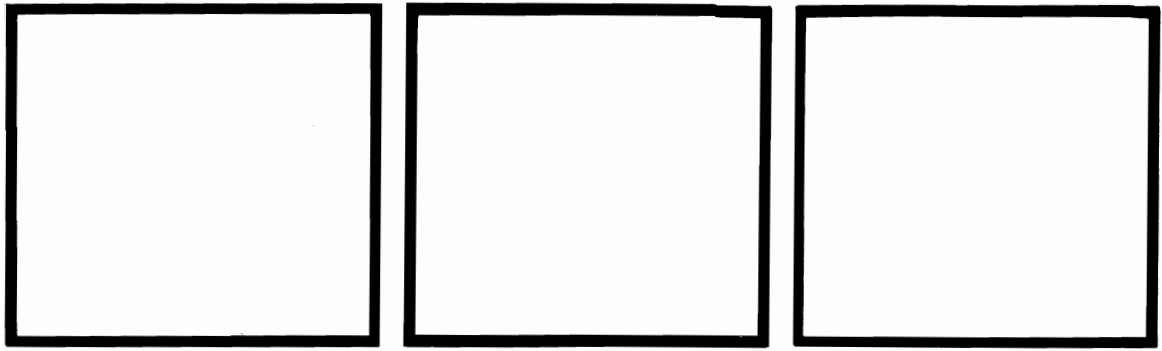
```
TO BFD :N
```

TO JFD :N

Test each procedure: use PFD 50, etc.

Use each type of forward in drawing a box, a triangle with right turns, a triangle with left turns. Copy the results here.

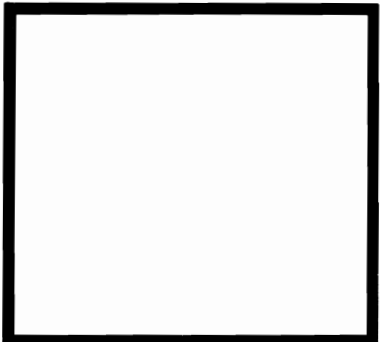


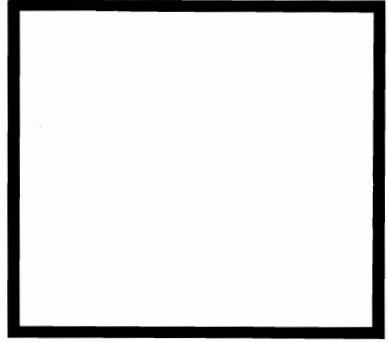


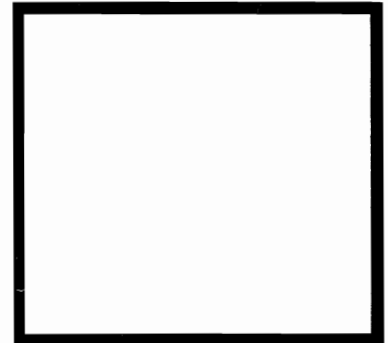
Experiment:

Use the new forwards to draw other shapes—octagons, hexagons, etc.

Record the procedures and results.









Lesson 33

Fractals

To create a fractal, start with a fancy forward, such as BFD.

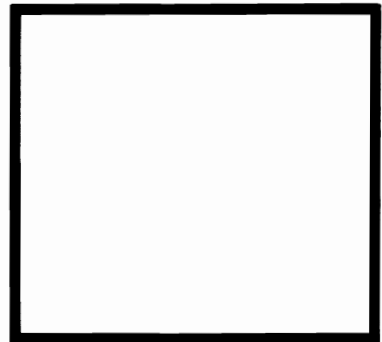
```
TO BFD :N  
FD :N/3  
RT 90  
FD :N/3  
LT 90  
FD :N/3  
LT 90  
FD :N/3  
RT 90  
FD :N/3  
END
```

Change all the FD :N/3 commands to BFD :N/3. Try BFD 80
What happens?

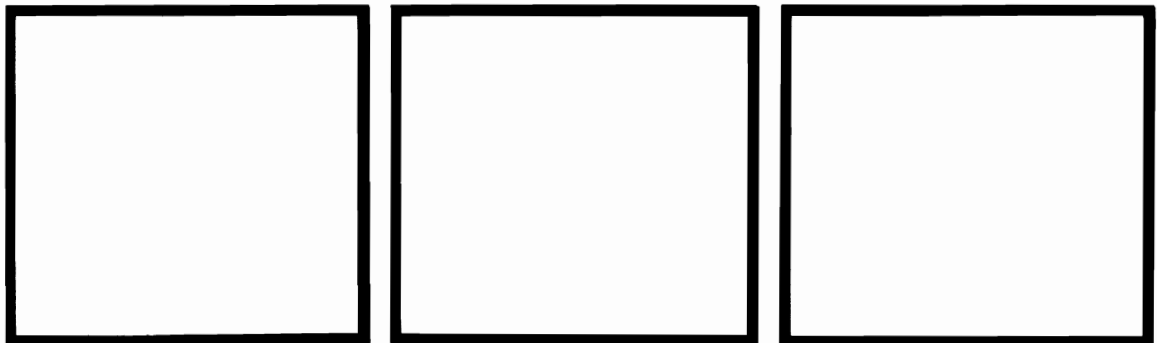
At some point we must stop being fancy and just draw a line.
Insert, at the beginning, the test:

```
IF :N < 30 FD :N STOP
```

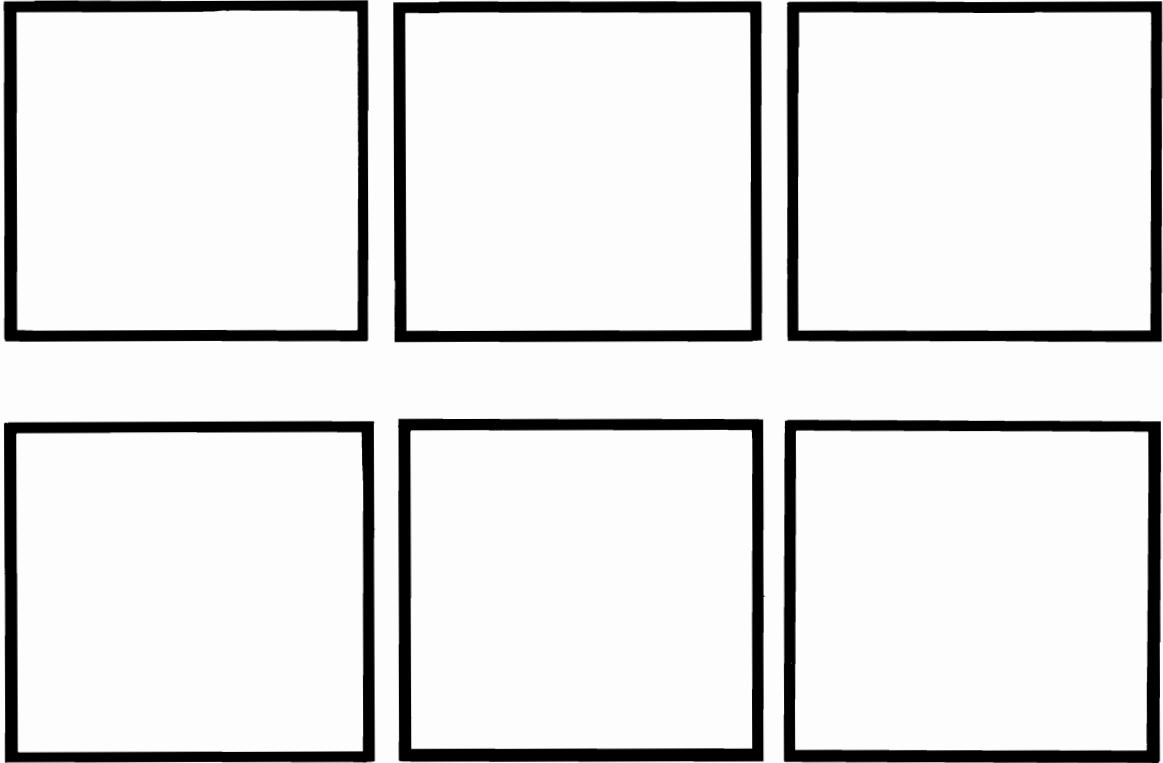
Try BFD 80. Copy the picture.



Change the 30 to 20, then to 10. Each time run BFD 80 and copy the picture.



Make fractals with PFD and JFD. Copy the results for :N < 30,
:N < 20, :N < 10.

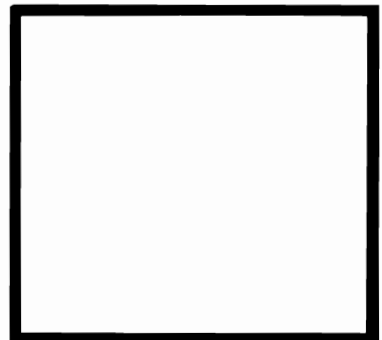


Design another fancy forward and turn it into a fractal.

B



A





Lesson 34 Snowflakes

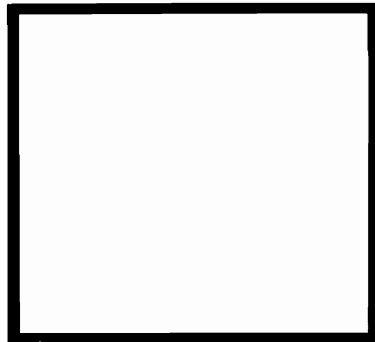
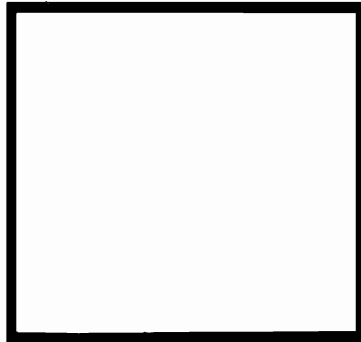
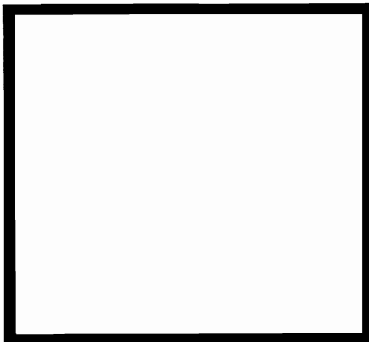
This time, we'll use the fractal PFD in TR :SIDE. You'll have to redefine TR; call it SNOWFL.

```
TO SNOWFL :SIDE  
  REPEAT 3 [RT 120 PFD :SIDE]  
END
```

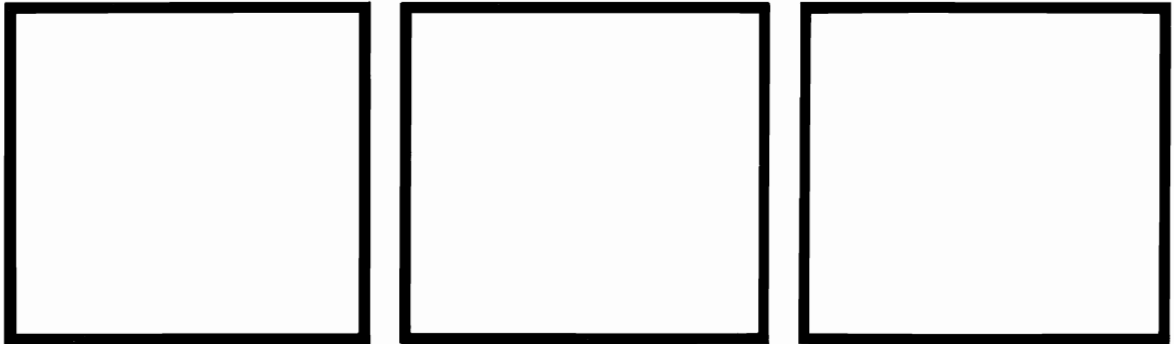
Try SNOWFL 80 with different stopping sizes in PFD.

Use :N < 30, :N < 10, :N < 5.

Copy the pictures here. (At least the first two.)

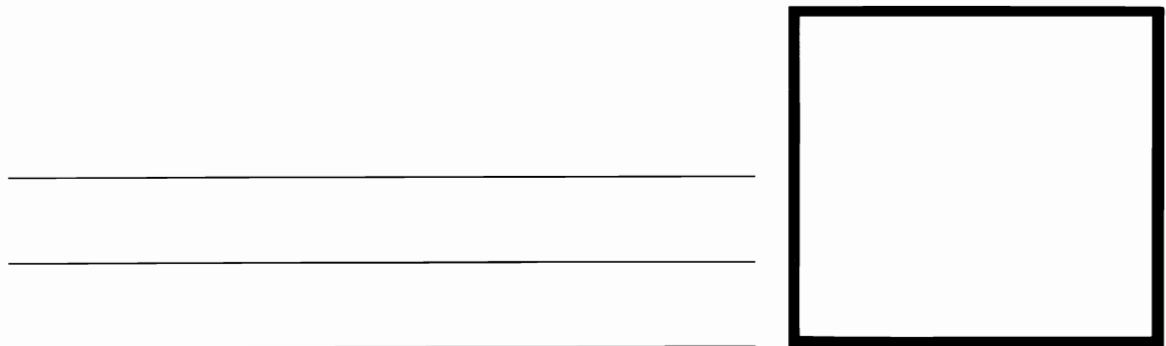
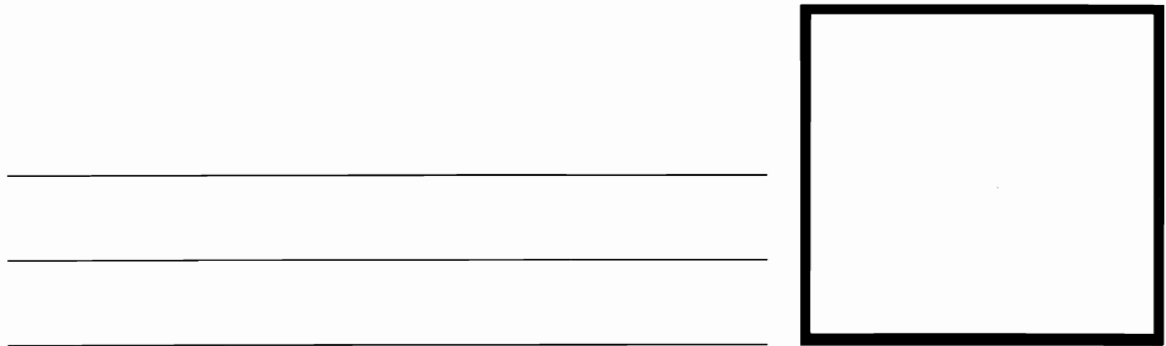


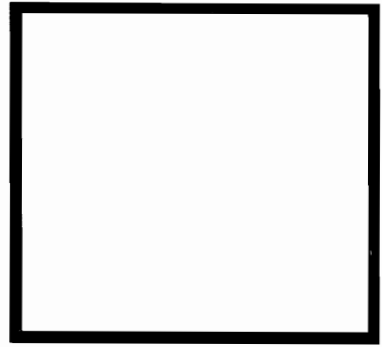
Change the turn in SNOWFL to LT 120. Do another sequence of snowflakes. Copy them here.

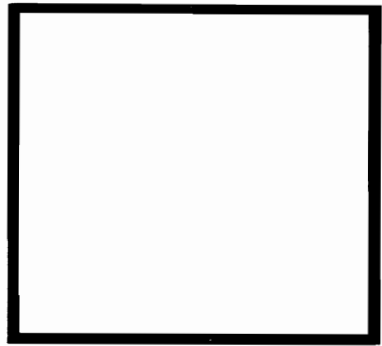


Try using HEX instead of TR to generate snowflakes. Use both LT and RT.

Try the other fractals with TR and HEX. Copy at least three different snowflakes—state which fractal and which shape produced them.









Lesson 35

Outputs

In advanced LOGO programs, you sometimes want to calculate something, but not print the calculated value.

A good example of this is the *absolute value* of a number. LOGO doesn't have an ABS function; however it's easy to write one. We want:

$$\text{ABS :X to be } \begin{cases} \text{:X if :X > 0} \\ \text{-:X otherwise} \end{cases}$$

To have ABS :X computed, but not necessarily printed, use OUTPUT.

```
TO ABS :X
  IF :X > 0 OUTPUT :X ELSE OUTPUT -:X
END
```

OUTPUT can be written OP

```
TO ABS :X
IF :X > 0 OP :X ELSE OP -:X
END
```

Define ABS, then try ABS -6.

Copy what LOGO printed: _____

Now try IF ABS -6 < 10 PR [HAPPY]

What happened this time? _____

Write each of the following LOGO procedures:

1. SGN :X outputs +1 if :X > 0
0 if :X = 0
-1 if :X < 0

SGN is known as the sign function.

```
TO SGN :X
```

2. AV :X :Y outputs the average of :X and :Y.

```
TO AV :X :Y
```

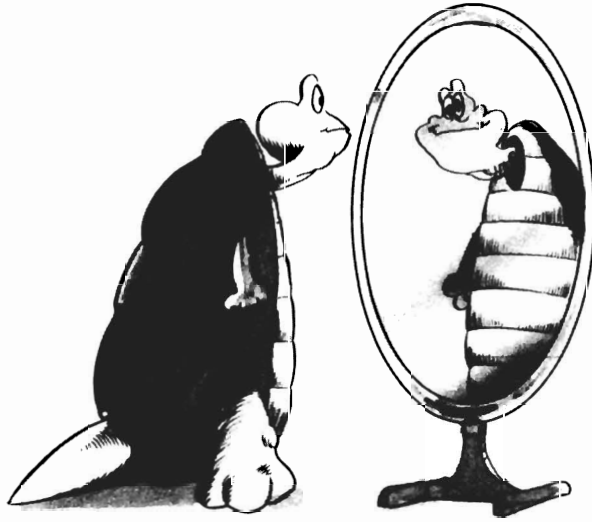
3. MAX :X :Y outputs the larger of :X and :Y.

4. MIN :X :Y outputs the smaller of :X and :Y.

5. DIST :X1 :Y1 :X2 :Y2 outputs the *square root* of $(X1 - X2)^2 + (Y1 - Y2)^2$.

LOGO has a command SQRT; SQRT :X outputs $\sqrt{:X}$

TO DIST :X1 :Y1 :X2 :Y2

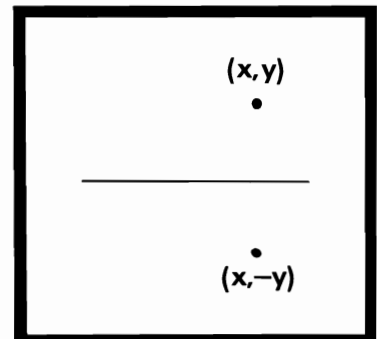


Lesson 36

Mirror Images

In this lesson, you'll make the turtle move in front of a mirror. To do this, you'll need a new procedure MFD (mirror forward).

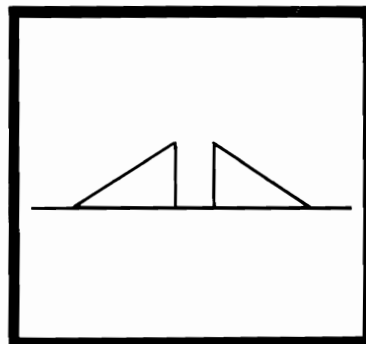
If the mirror goes across the middle of the screen, the image of (X,Y) is $(X,-Y)$.

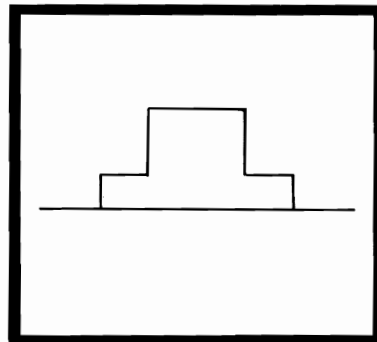


First draw a line across the middle of the screen. Return the turtle to the center with HOME.

In these pictures, only the turtle's path is shown.

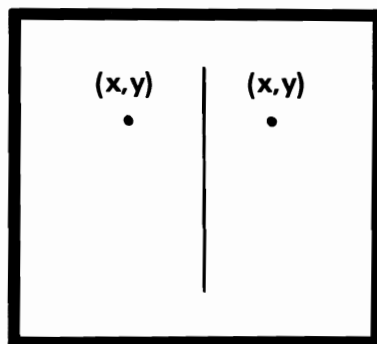
1. Use MFD, RT, LT, and FD (to lift the pen), to move the turtle along the path, drawing the reflections as well.



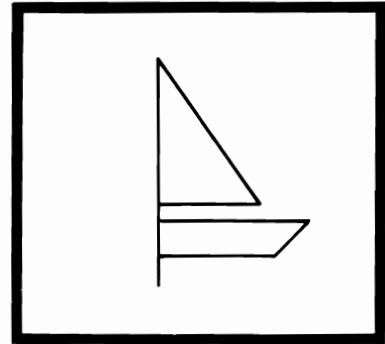


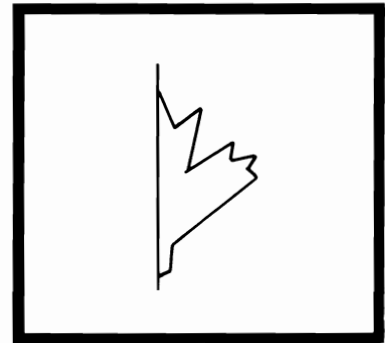
2. Change MFD to VMF (vertical mirror forward). This mirror goes up and down through the middle of the screen.

TO VMF :N



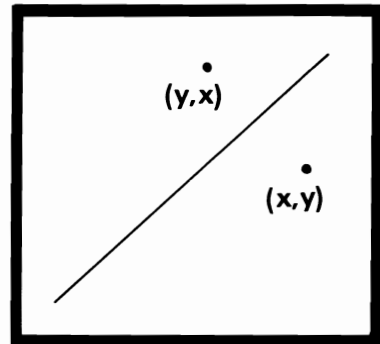
Move the turtle along these paths.
 Your procedure should draw the reflections, too.



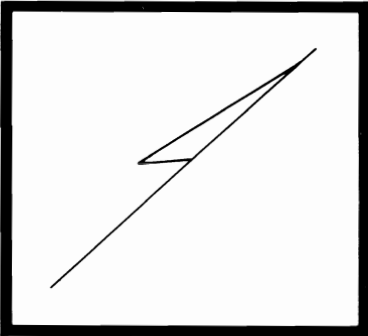


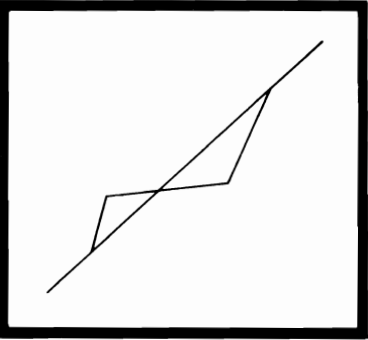
3. If the mirror goes from lower left to upper right, the image of (X,Y) is (Y,X) . Let DMF (diagonal mirror forward) be the procedure which reflects movement in this mirror.

TO DMF :N



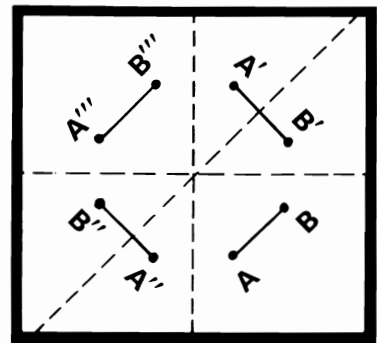
Use DMF to complete these pictures.

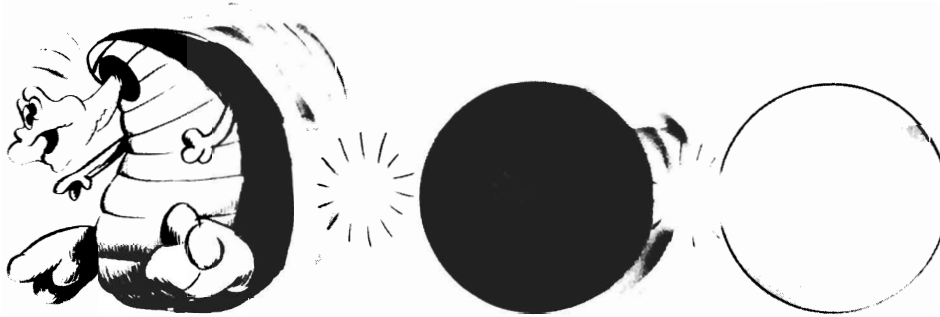




4. What happens when CMF, (combined mirror forward), is used?
CMF :N should do all three reflections:

TO CMF :N
move FD :N from A to B
draw A' B'
draw A'' B''
draw A''' B'''
hop back to B





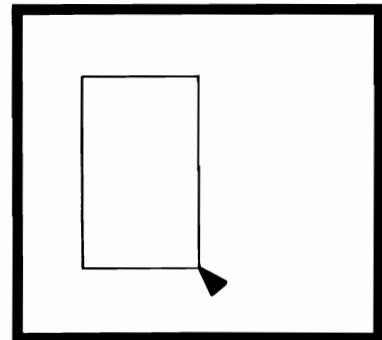
Lesson 37

Turtle Billiards

You'll need ABS.

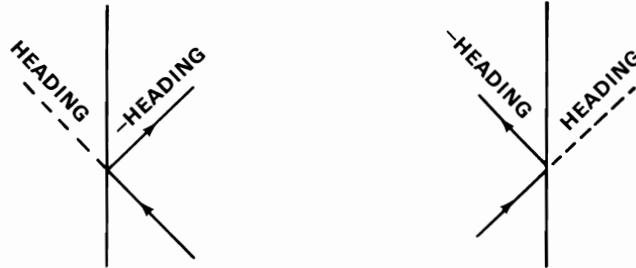
When this procedure is completed, you'll be able to:

- Tell LOGO the size of a rectangle (width :W and length :L).
- Have the rectangle appear on the screen.
- Have the turtle start at the lower right corner of the table with a heading of 315 (same as -45).
- Have the turtle proceed in a straight line; when it hits the edge of the table, it will bounce off the edge of the table like a billiard ball.

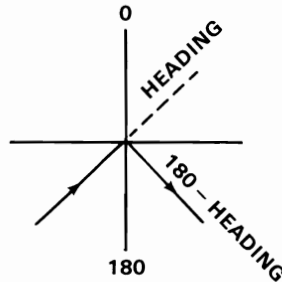


Break this project up into parts:

1. Draw the table, centered on the screen. Use SETXY, SETX, SETY. The lower right-hand corner is at $:W/2 - :L/2$
2. Place, aim, and start the turtle moving. Move 5 units at a time.
3. Check if so, $ABS :XCOR > :W/2$. If so, change heading to $-HEADING$.



4. Check if $ABS :YCOR > :L/2$. If so, change heading to $180 - HEADING$.



5. Move the turtle again.

Write the procedure BILLIARDS.

TO BILLIARDS :L :W

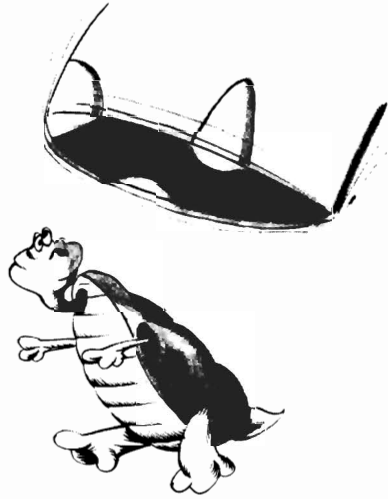
Optional Exercises

When BILLIARDS is working, change it to POOL. The turtle will stop when it reaches a corner. (“Reaches” means gets very close.)

Investigate which corner the turtle reaches on the different tables below. Use UR for upper right, LL for lower left, etc.

TO POOL :L :W

Table Length	80	120	80	100	110
Table Width	40	40	60	40	50
Final Corner					



Lesson 38

Words

When LOGO sees BOX, HOUSE, etc., it assumes these are the names of procedures. To tell LOGO that a *word* is intended, you use "BOX "HOUSE.

LOGO can do a lot of things with words. To see some of them, enter and run DEMO.

1. Write down next to each PR what it produces.

```
TO DEMO
MAKE "W "ELEPHANTS
PR :W
PR COUNT :W
PR FIRST :W
PR BF :W
PR LAST :W
PR BL :W
```

```
PR [ ]
PR ITEM 4 :W
PR MEMBER? "H :W
PR MEMBER? "Z :W
END
```

If "ELEPHANTS were changed to "ZEBRAS, what would DEMO do? Write your guess here. Then change DEMO and run it.

2. To indicate a word with no letters at all, use ". A word with no letters is called a *null* word. It's used mainly in tests.

The procedure DROP chops off one letter at a time.

```
TO DROP :W
IF :W = " STOP
PR :W
MAKE "W BF :W
DROP :W
END
```

To use DROP, type DROP "THERE

Change DROP to drop the last letter each time. Write your procedure here.

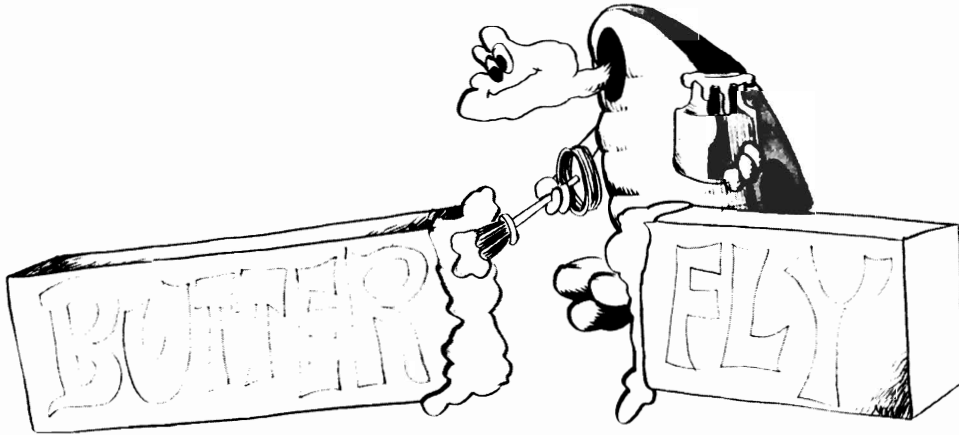
3. Write a procedure CHOP that drops the first *and* last letter each time. Try CHOP "TIGERS then CHOP "ELEPHANTS.

TO CHOP :W

Put in another test so that CHOP works on words with any number of letters. (Using COUNT is one possibility.) Record your final CHOP here.

4. Write a procedure to print the letters of a word in a column.

TO COL :W



Lesson 39 Gluing Words

LOGO uses WORD to glue together two words to make a new one. To glue more than two words, use parentheses.

1. Run GLUE to see how it works. Write down what is printed.

```
TO GLUE
MAKE "W "BUTTER
MAKE "T "FLY
MAKE "P WORD :W :T
PR :P
MAKE "Q WORD :T "AWAY
PR :Q
MAKE "Z (WORD :W :T :Q)
PR :Z
END
```

2. Write ECHO, which doubles the word and prints the new word.

```
TO ECHO :W  
MAKE "W _____
```

Try ECHO "THUMP

Make ECHO call itself; add ECHO :W as a last line. Then try ECHO "A.

3. Change ECHO so that, instead of printing :W, it prints the number of characters in :W. How long can :W get before LOGO can't hold any more?

4. Write a procedure, CYCLE, to remove the first letter from a word and glue it onto the end. If you run CYCLE "SHARE, the result should be HARES.

TO CYCLE :W

5. Finally, write a pig latin translator. You enter a word:

If the word begins with a vowel, add WAY to the end.

If the word doesn't begin with a vowel, transfer the first letter to the end of the word. Keep this up until you get a word starting with a vowel. Then, add AY to the end.

You can assume there are, at most, four consonants at the beginning of any word.

TO PIG :W



Lesson 40

More Words

These problems are more complicated. They involve all the word functions as well as REPEAT and, perhaps, recursion. There are several ways to do each one.

1. Write a procedure REV which takes a word :W and reverses the letters to form a new word. Call the new word :RW.

Hint: :RW begins with LAST :W
Its next letter is LAST of something else.
Keep building up :RW. (How many letters will it have?)

TO REV :W

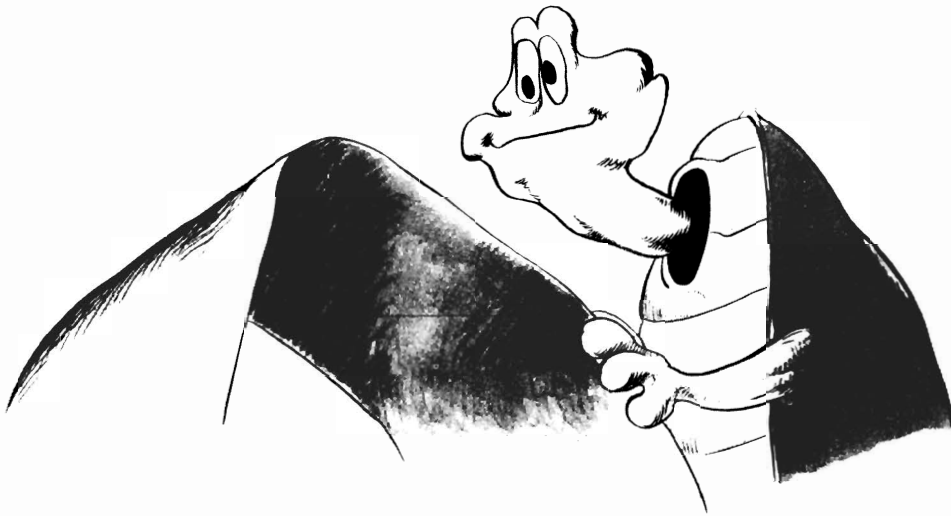
Try REV "123456789. Does it work for numbers?

A word which is the same read forward or backward is called a palindrome. Change REV to print PALINDROME, if the word was one.

2. Write a procedure DOUBLE to check whether a word has a double letter. DOUBLE should print TRUE if there is a double letter, FALSE otherwise.

Hint: Look at FIRST :W and FIRST BF :W
Are they the same?
Replace :W with BF :W and look again.

TO DOUBLE :W



Lesson 41

Lists

LOGO can handle *lists* as well as separate words. A word begins with " and ends with a space. A list begins with [and ends with].

COUNT, ITEM, FIRST, LAST BF, BL all work with lists.

1. Run TRYLIST.

```
TO TRYLIST
MAKE "L [A BLUE FLY WAS BITING AT THE DOG ONCE]
PR :L
PR COUNT :L
PR ITEM 2 :L
END
```

Change TRYLIST so that every other word in the list, starting with word 2, is printed. (Use REPEAT.)

Change TRYLIST so that every third word, starting with word 3, is printed.

2. Write BABBLE, a procedure in which:

:N is a list of nouns, for example, [DOGS CATS MICE
ELEPHANTS]

:V is a list of verbs, for example, [LIKE FEAR CHASE]

:BABBLE prints a random item from :N, followed by a
random item from :V, followed by another random item
from :N.

3. The elements of a list can also be lists.

Change BABBLE so that :V looks something like:

MAKE "V [[ARE AFRAID OF] CHASE [CAN'T STAND]]

Make it more interesting. Change :N, too.

4. Write MENU which contains:
- A list of appetizers. :A
 - A list of main dishes. :MD
 - A list of vegetables. :V
 - A list of desserts. :D

MENU should select a random item from each list.

If you want the chance of skipping vegetables, let one of the items in :V be [], the empty list.



Lesson 42 Gluing Lists

Two lists can be glued together in different ways.
SE (SENTENCE) makes one big list of the words in both lists.
LIST makes a list whose items are the original lists.

1. Run DEMOLIST.

```
TO DEMOLIST
MAKE "A [DOGS CATS MICE]
MAKE "B [HOUSES TREES BIRDS]
MAKE "C LIST :A :B
PR :C
PR ITEM 2 :C
MAKE "D SE :A :B
PR :D
PR ITEM 2 :D
END
```

For getting at individual words, use SE.

4. Combine TALKBACK with REV, from Lesson 40, which reverses a word, to define a procedure which prints a list completely backward.

Try your procedure on: ABLE WAS I ERE I SAW ELBA and MADAM I'M ADAM.



Lesson 43

Tell LOGO Something

You can tell LOGO to wait until something is typed and <return> is pressed. RQ (REQUEST) does this.

RQ is the list that was entered.

1. Try LOGIC.

```
TO LOGIC
PR [WHAT'S YOUR FULL NAME?]
MAKE "NAME RQ
PR [WHAT'S YOUR FAVORITE COLOR?]
MAKE "COL RQ
PR (SE :NAME [LIKES] :COL)
END
```




Lesson 44

Number Games

Numbers also have to be changed from lists to words.

1. Write a simple addition quiz.

Follow this outline:

- Let LOGO pick two numbers at random, say :A and :B.
- Print $3 + 4 = ?$ (use the values of :A and :B)
- Change the answer to a word. Check if it's correct.
- Print RIGHT or WRONG. If wrong, tell the answer.



Lesson 45

Testing

There are times when IF ELSE is hard to use, especially if the actions are complicated or lengthy.

Another way is to use the commands: TEST
IFT (IFTRUE)
IFF (IFFALSE)

1. Try this.

```
TO SIZE :N  
TEST :N > 1000  
IFT PR [A LARGE NUMBER]  
IFF PR [NOT SO LARGE]  
END
```

Try SIZE 100 and SIZE 9999.

4. If RULE is to check that :W has *both* an E and an R, use ALLOF.

```
IF ALLOF MEMBER? "E :W MEMBER? "R :W OP "TRUE ELSE . . .
```

To check for *either* an E or R, use ANYOF.

```
IF ANYOF _____
```

ALLOF means *both* of two possibilities.

ANYOF means *either* of two possibilities.

Write a secret rule game:

- LOGO asks for a word.
- Player enters a word.
- LOGO responds with YES or NO.
- LOGO asks for another word.
- Player keeps trying words until sure of the rule.

Make up your own rule for YES and NO.

TO GAME



Lesson 46

Search and Replace

Suppose you wrote a story about a dog, and then decided that you wanted to change DOG to CAT everywhere in the story.

LOGO can look at the words of the list and make a replacement. It looks at the first word, decides whether or not to change it, then looks at the first word of BUTFIRST of the list.

Here's how to do it:

```
TO REPLACE :OLD :NEW :L
  IF :L = [] OP []
  TEST :OLD = FIRST :L
  IFT OP SE :NEW REPLACE :OLD :NEW BF :L
  IFF OP SE FIRST :L REPLACE :OLD :NEW BF :L
END
```

Try REPLACE "DOG "CAT [THE DOG LOOKED LIKE MY DOG]

Run TELLME. When TYPE A SENTENCE appears, type:

'THE QUICK BROWN FOX'

Write LISPER which changes every S to TH. Try LISPER with:

SISTER SUSIE SITTING ON A SOFA.

3. Use WORDPLACE in a decoding program. After the user types in the coded message, the procedure should print the message.

(Use apostrophes to enter the message as a word.)

Then LOGO should print CHANGE? (user types a letter)
TO (user types a new letter)

The message should be changed and printed out.

LOGO should ask for the next change.

Try your decoder on this message:

ERY GZQT VABACGTF LGT OZCL XIRSOTH AB LGT ORNR
JRIWSRRW.

APPENDIX 1

Comments About Editing Procedures

Making changes in a procedure, even correcting typographical errors, can be difficult at first. These observations may help.

1. Typing characters *inserts* them where the cursor is.
2. Pressing the *delete* key erases the character to the *left* of the cursor.

If your line reads
FD 250 RT 90
and you want it to read
FD 25 RT 90

Move the cursor to the space after 250 and press .

3. CTRL K erases everything to the right of the cursor.
4. <Return> moves everything to the right of the cursor to the next line.
5. Press RUN/STOP or CTRL C to save a procedure.



APPENDIX 2

Formatting a Disk Saving and Reading Files

If you wish to save your LOGO procedures, you'll need a formatted disk.

Before loading LOGO, insert a blank disk in the drive and type:

```
OPEN 15, 8, 15 <return>  
PRINT#15, "NØ:LOGOFILES,MW" <return>
```

(use your initials instead of MW)

The disk drive will make noises for about two minutes as the disk is formatted.

When the red light goes off, remove the disk, and label it.

Now load LOGO and proceed with your work.

At the end of your session, to see the names of all the procedures you have defined, type:

```
POTS <return>  
(POTS stands for print out-titles)
```

To save all your procedures, pick a file name that means something to you, say, MWSHAPES.

Type:

```
SAVE "MWSHAPES <return>
```

The drive will make noises as the file is saved.

To read these procedures in the next time you use LOGO, first start up the LOGO program. When it is running, type:

```
READ "MWSHAPES <return>
```

When the file has been read, type:

```
POTS
```

to see what procedures you have.

Each time you write a file, use a different file name. If you've forgotten what you called a file, type:

```
CATALOG <return>
```

to see the names of everything on the disk.

Index of LOGO Primitives

<u>Primitive</u>	<u>Long Form</u>	<u>Lesson</u>
ALL OF	—	45
ANYOF	—	45
BF	BUTFIRST	38
BG	BACKGROUND	7
BK	BACK	7
BL	BUTLAST	38
CATALOG	—	APPENDIX 2
COUNT	—	38
DOUBLECOLOR	—	7
DRAW	—	1
ELSE	—	29
END	—	4
FD	FORWARD	1
FIRST	—	38
HEADING	—	14
HOME	—	17
HT	HIDETURTLE	12
IFF	IFFALSE	45
IFT	IFTRUE	45
IF	—	14
ITEM	—	38
LAST	—	38
LIST	—	42
LT	LEFT	1
MAKE	—	10, 22
MEMBER?	—	29
OP	OUTPUT	35
POTS	PRINTOUT TITLES	APPENDIX 2
PC	PENCOLOR	7
PR	PRINT	10
RANDOMIZE	—	20
RANDOM	—	10, 20

RC?	—	29
RC	READCHARACTER	29
READ	—	5
REPEAT	—	4
RQ	REQUEST	43
RT	RIGHT	1
SAVE	—	4
SETH	SETHEADING	14
SETXY	—	17
SETX	—	21
SETY	—	21
SE	SENTENCE	42
SINGLECOLOR	—	7
SQRT	—	35
STOP	—	14
ST	SHOWTURTLE	12
TEST	—	45
TO	—	4
WORD	—	39
XCOR	—	21
YCOR	—	21

Index to Procedures Used In the Workbook

<u>Procedure</u>	<u>Lesson</u>
ABS :X.....	35
ANIMAL.....	43
ARC :DIAM.....	13
ARRANGEMENT.....	13
ARROW.....	21
AV :X :Y.....	35
BABBLE.....	41
BFD :N.....	32
BIGCIRCLE.....	12
BILLIARDS :L :W.....	37
BLASTOFF.....	27
BLAST.....	24
BLOSSOM.....	15
BOX.....	4
BRANCHES.....	21
BUBBLES.....	20
CHOP :W.....	38
CIRCLE :DIAM.....	12
CLOVER.....	15
CMF :N.....	36
COL :W.....	38
COUNTER :N.....	27
CYCLE :W.....	39
DEMOLIST.....	42
DEMO.....	26
DEMO.....	38
DESIGN.....	6
DEVOWEL.....	42
DIST :X1 :Y1 :X2 :Y2.....	35
DMF :N.....	36

DONUT	11
DOUBLE :W	40
DROP :W.....	38
ECHO :W.....	39
ENDLESS	10
EYEBALL.....	12
FIGURE.....	10
FLASHY.....	29
FLORAL :S :A	13
FLOWER.....	7
FOREST.....	21
FRAME	17
GAME.....	44
GARDEN.....	7
GLUE.....	39
GRSQ :S	22
GRTR :S.....	22
HALO	8
HEX :SIZE	11
HUT :SIZE	18
INFO :W	40
JFD :N	32
KALEID.....	8
KFD :N.....	36
LOGIC.....	43
MASTER :T	24
MASTER :T	29
MAX :X :Y	35
MENU.....	41
MFD :N	36
MIN :X :Y	35
MYSTERY	7
NESTS :N.....	26
OCT :SIZE	11
PATTERN :SIDE :ANGLE	10
PATT :S :A	14
PETAL.....	15
PFD :N.....	32
PICT	6

PIG :W	39
PINETREE	21
POOL :L :W	37
QUIZ :ANIM.....	43
REPLACE :OLD :NEW :L	46
REV :W	40
RING :XCEN :YCEN :RAD	25
RULE :W	45
SBOX :S	30
SCIR :DIAM :C	30
SECRET :W	45
SETSQ :S.....	24
SFD :N.....	30
SGN :X	35
SHRINK :N	24
SHRSQ :SIDE	22
SHTR :S.....	22
SIZE :N	45
SNOWFL :SIDE.....	34
SNOWMAN	17
SNOW	20
SOLSQ :S	24
SPATTERN :S :A :C.....	30
SPIRAL :A :R	28
SQ :SIDE.....	9
SREC :H :W :C	30
STEM	7
STEP	5
STR :SIDE :C	30
SUM :X :Y	26
SUPEFLOWER.....	8
SUPERTRI.....	4
TALKBACK.....	42
TELLME.....	46
TINYARC.....	15
TRI1	4
TRI2	4
TRUNK	21
TRYLIST	41

TR :EDGE.....	9
UNSPIRAL :S :R.....	28
VMF :N	36
VOWEL.....	29
WORDDEMO.....	46
WORDPLACE :OLD :NEW :W.....	46
XMAS.....	19

Background (BG) and Pen Colors (PG)

BLACK	0
WHITE	1
RED	2
CYAN	3
PURPLE	4
GREEN	5
BLUE	6
YELLOW	7
ORANGE	8
(RUST)	9
LT RED	10
DK GRAY	11
MED GRAY	12
LT GREEN	13
LT BLUE	14
LT GRAY	15

Mary Jean Winter is a professor of Mathematics at Michigan State University. She has been actively involved in mathematics education (K-12) for the last 8 years. She was formerly a numerical analyst in industry and academia. She is the author of The Computer Playground series and Compumath Magic published by DATAMOST, Inc. She has also written Chivalry, Great Adventure, Wordspot and Witchnumber, published by Comm*Data; numerous articles on using calculators and computers in the classroom, as well as calculator activities for Scott Foresman's elementary series math textbook. The author also teaches computing at MSU Summer Computer Camps for children ages 10-16. She has a Ph.D. in mathematics from Carnegie Mellon University and an A.B. in mathematics and history from Vassar College.



THE COMMODORE 64

LOGO WORKBOOK

DATAMOST proudly introduces THE COMMODORE 64 LOGO WORKBOOK, the third in a series of educational computer books written especially for young people in grades 2 through 6. The first series, COMPUTER PLAYGROUND, introduced BASIC programming by means of graphics, games and words. The second, COMPUMATH MAGIC, showed those who already knew some BASIC how to use it to solve problems. Now comes THE COMMODORE 64 LOGO WORKBOOK, which introduces the LOGO language and teaches how LOGO can be used in problem-solving.

Written for children, this wonderful book shows them what a "turtle" is, what procedures are, how to use visual problem-solving, variables, geometry, and even recursion. Combined with the Commodore 64 LOGO package, this book becomes a powerful learning experience with the educational language.

Structured in lesson form, two-thirds of THE COMMODORE 64 LOGO WORKBOOK is directly concerned with graphic geometric lessons, providing immediate visual satisfaction and teaching about symmetry and the idea of a limit at the same time. Problem-solving skills are developed and applied at every level. The last third of THE WORKBOOK deals with sentences and words, with applications in the game category.

ISBN 0-88190-364-7



 **DATAMOST**[™]
INC

20660 Nordhoff Street, Chatsworth, CA 91311-6152
(818) 709-1202