ARITHMETIC TEACHER, vol. 32(2), Outubro 1984

Training Elementary School Teachers to Use Computers—with Emphasis on Logo

By Edward J. Davis, Jim Helms, F. A. Norman, *and* Joao Ponte

For the past two summers we have taught a two-week "computer camp" for teachers of grades one through seven. The teachers, who were paired by grade level (one pair at each computer), spent approximately three hours each day in "class" at an Apple II$^+$ microcomputer and averaged another two hours a day in individual practice. Participants were selected on a first-come, first-served basis in response to announcements sent to area schools. The most recent camp session included eleven elementary and four middle school teachers and was taught by a university faculty member and four graduate assistants in the mathematics education department.

Activities for Programming in Logo

The teachers enjoyed working on programming assignments that helped them to apply their new knowledge. They seemed to appreciate using Logo in situations that would probably be interesting and instructional for children. Figure 1 displays a typical camp assignment, a sequence of four programming tasks. As a prerequisite, it was assumed the student could write a program to draw squares of any specified size. An example of such a program (or procedure) to generate squares is reviewed at the top of figure 1.

The four tasks were of increasing difficulty. In some cases ideas from the program for one task were used in subsequent tasks. For example, the three smallest squares in task one can be rearranged to complete task two. Then, the two smallest squares in task two form the body and cab of the truck in task three. This assignment was designed also to give practice in substituting values in a program containing a variable and to stress the importance of knowing the position and heading of the turtle at the completion of each portion of the required drawing.

Figure 2 shows a similar sequence of related tasks that involve a circle procedure. The solution to task one can be used three times in task two. Task four can be accomplished by annexing solutions to tasks two and three. Writing programs in such a sequence models a stepwise refinement of a problem for the student and might serve as an introduction to top-down design of a program.

Another programming assignment involved predicting the outcome of a given program, modifying that program, and continuing to predict and observe results. Imagine that you had a program that would create a drawing with these specifications:

> Follow these instructions until X exceeds 100
> Draw a segment of any length X
> Make a right turn of 60 degrees
> Increase your segment length by 5 and begin again

The program in Logo could be as follows:

```
TO SPIN :X
IF :X > 100 THEN STOP
FD :X      RT 60
SPIN :X+5
END
```

The left side of figure 3 gives the shape produced by the program when the initial segment has a length of 3. Modifications of this recursive program include different beginning and ending lengths, changes in the increment for X, and adjustments to the degree of turn. The right side of figure 3 gives such modifications—changing a turn of sixty degrees to ninety degrees and changing the increment of X to 10. This assignment was intended to foster a realization that even small changes in a program can create extreme changes in output. In addition, it was felt that making changes in programs and then imagining the consequences of these changes, before running the modified program, would help teachers develop skills that are essential in debugging programs.

Edward Davis is an associate professor of mathematics education, teaching courses to preservice and in-service teachers of grades K–12 at the University of Georgia, Athens, GA 30602. Jim Helms is a mathematics instructor at Waycross Junior College, Waycross, GA 31501, where he has worked with students and teachers in computer camps for the past two summers. F. A. Norman is a doctoral student at the University of Georgia. He has been an instructor in computer camps for children and teachers. Joao Ponte is a lecturer at the University of Lisbon who is presently pursuing a doctoral degree at the University of Georgia through a grant from JNICT-INVOTAN, Lisbon, Portugal.

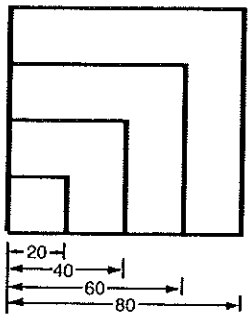
Fig. 1

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

All four pictures below can be drawn using SQ :X pieces of different sizes.

Task 1

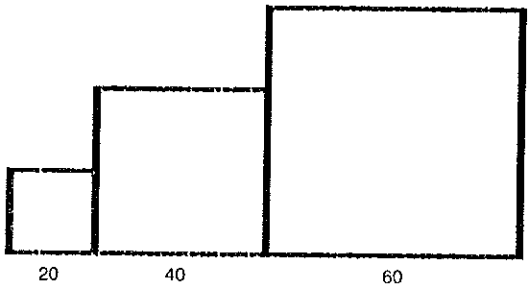
TO GROW
SQ 20



END

Task 2

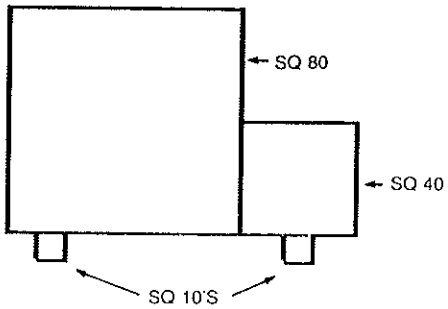
TO BLOCKS



END

Task 3

TO TRUCK



Task 4

TO FACE

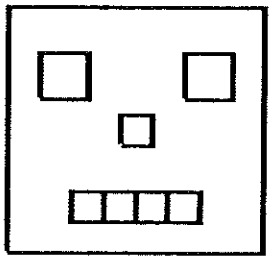
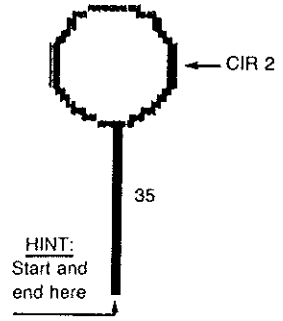


Fig. 2

```
TO CIR :X
HT
REPEAT 36 [FD :X RT 10]
END
```

Task 1

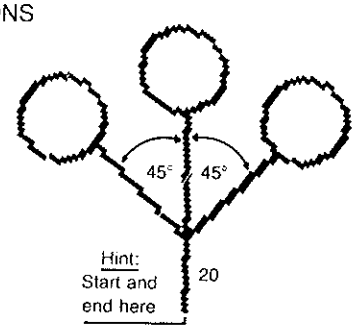
TO BALLOON
FD 35



END

Task 2
Now use BALLOON to draw BALLOONS

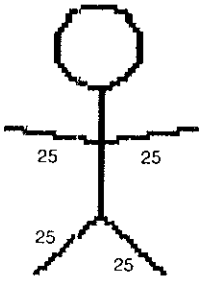
TO BALLOONS
FD 20



END

Task 3
Add arms and legs to BALLOON to make MAN

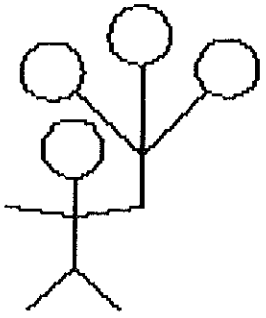
TO MAN



END

Task 4
Put MAN and BALLOON together for B.MAN

TO B.MAN



END

Fig. 3

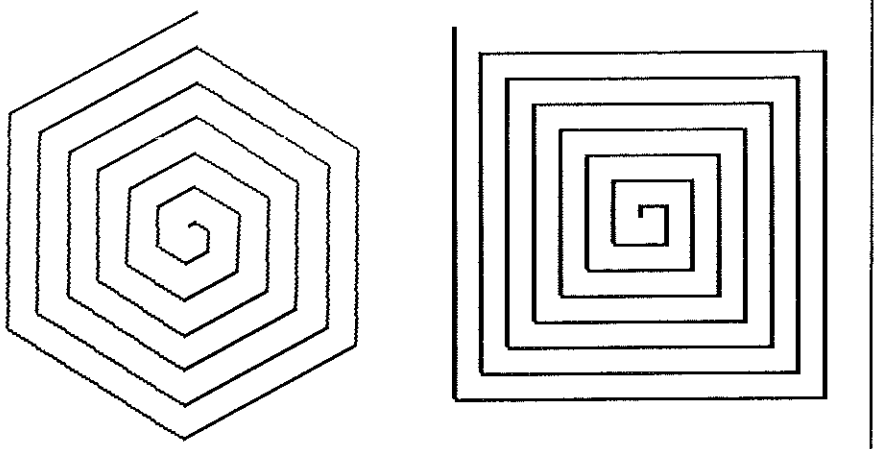


Figure 4 presents another drawing assignment. The house was presented in a way to encourage the use of a coordinate system. Students were encouraged to use a combination of previously constructed programs to draw the needed rectangles, squares, and circles, and supplement these with appropriate use of SETXY. Separate programs could be created for the door and windows.

The checkerboard shown in figure 5 contains a number of patterns. For a complex task a top-down analysis is particularly useful (see Appendix). Such an analysis would repeatedly break the task into less complex subtasks. For example, the checkerboard might be viewed as a stack of four identical blocks. Each block consists of two rows of alternating shaded and unshaded squares. Each of these rows can be further decomposed into four dominoes formed by one shaded and one unshaded square. These observations can help teachers break down the seemingly large and tedious task of composing sixty-four separate squares into the smaller task of writing procedures for two types of squares and devising a few building moves for constructing the successively larger pieces of the checkerboard. We anticipated and encouraged teachers to create original approaches to this challenging problem.

Curriculum and Materials

The training curriculum included introductions to the languages Logo and BASIC, familiarization with MECC Elementary Volumes 1–13 and Spelling Volume 1, word processing with Logo and Bank Street Writer, and consideration of various views of the potential roles of computers in education, with attention to Papert's (1980) ideas in *Mindstorms*.

Logo is designed to encourage a problem-solving approach.

Teachers were required to complete a series of Logo lessons in a tutorial workbook produced by the staff, modify and extend BASIC programs presented in class, complete summaries and critiques of MECC's instructional materials, read sections from *Mindstorms*, and develop projects showing some ways they planned to involve their students with computers, using content from any subject area in the curriculum they followed.

Our Logo tutorial workbook first introduced drawing with turtle graphics. Next, teachers learned to write procedures (programs) using the turtle commands and later to include variables in these procedures. Disk-management commands were taught as a need to save programs developed. Recursive procedures were then taught so as to highlight the power of the language. Finally, word processing in Logo was presented. Apart from the tutorial workbook, lessons on interactive commands and music were given. By the end of the camp, teachers were using the components of Terrapin Logo shown in figure 6. These components are listed in order of their introduction.

On most days a portion of the instruction included applications of Logo for children. We designed programming activity sheets containing a series of increasingly difficult tasks that the teachers were to complete. Some of these activity sheets were shown in figures 1–5. Frequently we created sequential programming tasks—sequential in the sense that the more advanced tasks were much easier if the students used the results of previous tasks. The breaking down of complex tasks into a series of simpler tasks can be a viable problem-solving strategy, and it relates directly to the structure of Logo, which is designed to encourage such a problem-solving approach.

Originally, we intended to spend approximately 40 percent of the time allotted to computer-language instruction exploring BASIC and 60 percent exploring Logo. We gave this relative emphasis to Logo and BASIC during the first summer camp. However, presenting a Logo lesson and a BASIC lesson on the same day resulted in some confusion. Our plan for the second summer was to present only one language at first to forestall difficulties in learning two languages.

The enthusiastic reception for Logo during the first summer encouraged us, in the second summer, to spend all the language instruction time in the first week on Logo. Nevertheless, the delayed introduction of BASIC again resulted in considerable confusion about such system commands as READ in Logo and RUN in BASIC. Structural and editing differences between BASIC and Logo also caused concerns, and teachers began to lose much of the confidence they had ac-

Fig. 4

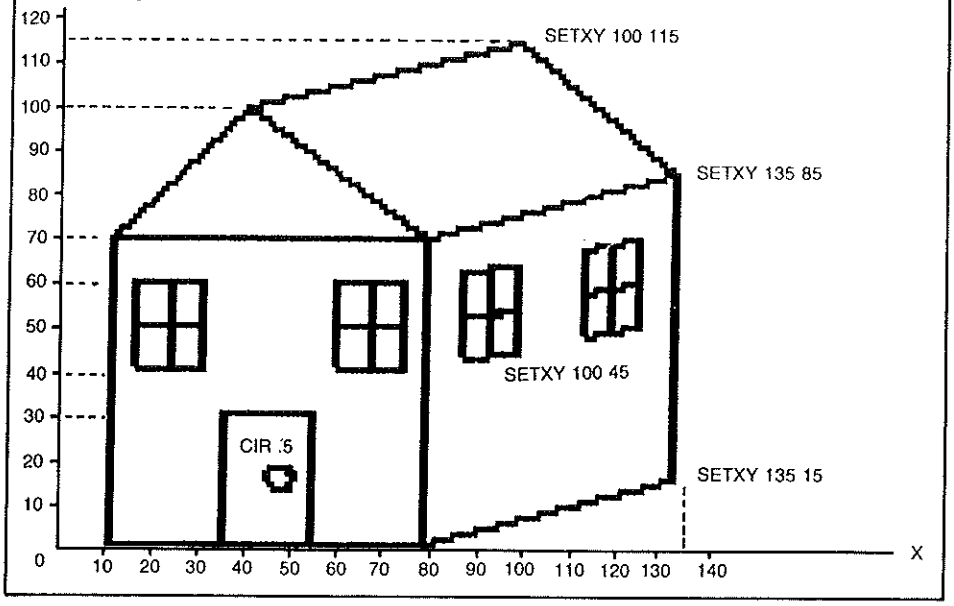
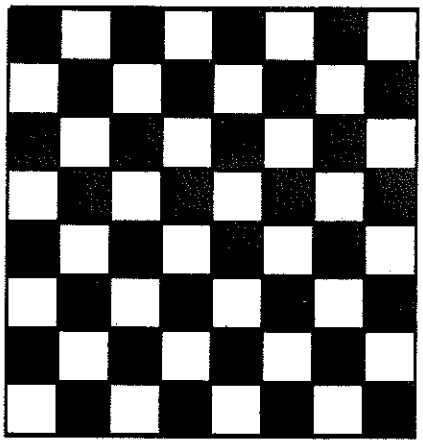


Fig. 5

Programming Challenge: Create a checkerboard procedure to meet the following specifications:



1. A standard 8 × 8 64-square board
2. Alternated shaded and unshaded squares
3. Outside dimensions of 160 turtle steps on each side

Use the space below for planning. This problem can be solved in many different ways. Planning in the beginning will save time in the end.

Fig. 6

DRAW	HIDETURTLE	HEADING
FORWARD	SHOWTURTLE	SETHEADING
RIGHT	PENCOLOR	STOP
BACK	BACKGROUND	PRINT
LEFT	REPEAT	IF . . . THEN
TEXTSCREEN	Workspace Management	IF . . . THEN . . . ELSE
FULLSCREEN	Commands	CURSOR
SPLITSCREEN	Saving & Recalling	MAKE
HOME	from Disks	REQUEST
PENUP	Printer Interface	TEST
PENDOWN	Commands	IFFALSE
CLEARSCREEN	SETXY	IFTRUE

PLAY (from Utilities Disk)

quired during the initial week. Our response was to reduce considerably the emphasis on BASIC and to expand Logo instruction to deal with interactive programs and musical capabilities as well as turtle graphics.

Evaluation

We gathered evaluation data from the teachers on the first and last days of our second summer camp. We developed and administered tests to assess proficiency in the logic of programming (apart from the computer language), interpretation of programs written in Logo and BASIC, and knowledge of random numbers, variables, geometry, and coordinate systems. In addition, we developed an evaluation form designed to assess the teachers' attitudes toward the curriculum and the instruction they had received. Finally, we created an inventory of teachers' preferences for the availability of computers, beliefs about their skills in using computers, and reasons for teaching their students to write programs.

The responses to these instruments indicated statistically significant gains from pretest to posttest in mathematical knowledge of variables, coordinate systems, and random numbers. Teachers brought to the camp the ability to think in a mode required for writing computer programs. Also, teachers gained confidence in their computing abilities and changed their original preference for one computer in each cluster of classrooms to a preference for several computers in each classroom. The principal reasons expressed by the teachers for showing their students how to write programs shifted from the importance of computer literacy and the need to foster creativity to the need to teach logical thinking. Teachers' evaluations showed that they viewed the camp as a very positive experience, enjoyed working with a partner, and felt successful and confident in using Logo.

Evaluating each teacher's performance was a delicate problem. During the session we tried to instill an understanding in the teachers that a program is not completely wrong if it

contains a bug. We felt that the teachers gained confidence in diagnosing errors and correcting mistakes. At the same time, however, use of this approach to errors created a problem in the evaluation of the teachers' progress by traditional paper-and-pencil tests. Indeed, such tests may not be appropriate. With a computer, students (and teachers) can run a program and correct mistakes that they or the computer discover—there is no need to be perfect the first time. The computer is forgiving, and in Logo corrections can be made with relative ease.

Teachers and Logo

The most salient feature of the camp was the enthusiasm of the elementary school teachers for Logo. They liked the language and rapidly developed proficiency in using it. Our close interaction with teachers and the complexity of the programs they constructed for classroom use indicated their considerable competence and confidence with Logo. We were very impressed by the teachers' involvement in each day's lesson, their desire and requests to learn new commands, and their willingness to spend time on their work outside of class. Further evidence of their interest was shown by their attention to such details as improvements in their programs and the addition of new features.

Teachers were able to construct procedures involving both turtle graphics and words and lists (conversational features). They designed their own procedures for counting, keeping time, and scoring and embedded these in other procedures. Teachers' products included such games as golf and baseball, practice exercises and drills with graphics components in arithmetic and language arts, and demonstrations of concepts using textual and turtle graphics.

However, many teachers failed to appreciate readily the possibilities of Logo for programming by their students. The nature of the instructional Logo programs they produced indicated that the teachers still preferred to use computers essentially for the dispensing of information or for drill and practice. Most were not planning to have their students become actively involved in writing computer programs. This preference seems to imply that despite our emphasis on programming tasks, the teachers did not see much of Logo's potential—as Papert (1980) sees Logo. We suggest that some of the motivation of the teachers to use Logo in tutorial and drill contexts was a natural application of their newly acquired knowledge of this programming language to familiar instructional contexts. Teachers wanted to use Logo for graphics, explanation, and scoring of students' work—things they are used to doing. For teachers, creating Logo assignments for their students as explorations may have appeared to be a completely new venture.

Teachers rapidly developed a proficiency with Logo.

What are some strategies to help teachers see new opportunities in having their students write programs? One strategy would be to respond to their questions about particular programs and programming techniques in ways that encourage trial and discovery. At the same time, this kind of response could be identified so that, in turn, teachers would interact with students in the same manner. To the extent that teachers value discovery learning, they could see that programming in Logo is a fertile environment for discovery.

A second strategy to encourage students' programming would be to have teachers identify and examine areas in the curriculum where students' learning would be enriched by writing programs. For example, students could write programs to draw symmetric, congruent, or similar figures, thereby enriching their understanding of these concepts. The concepts of fractions, decimal numbers, and percentage might also be reinforced by having students write programs to display examples of them (see fig. 7). If a physical model for a mathematics concept exists, then the turtle ought to be able to draw a picture of this model. Thus, students should be able to write programs depicting such mathematical concepts as mixed numbers, equivalent fractions, bar and circle graphs, isosceles triangles, altitude, centimeters, and area.

In the camp, teachers often began working on projects requiring a good grasp of very complex Logo capabilities. They did not redesign their projects to fit the limitations of their skills but held onto the original design. Doing so forced them either to learn more advanced Logo commands in a very short time or often to request help without always fully understanding what was happening in the program. Also, in working with the tutorial workbook, teachers sometimes went through the lessons without fully exploring and reflecting on the material. The "student syndrome" could have caused them to want to advance as rapidly as possible without reflecting on what they were learning.

To counteract the misunderstandings that can result from the teachers' using advanced commands or working through material too quickly, instructors need to provide many examples of how commands operate and give applications of all new procedures. It seems advisable to look again to the school curriculum for these examples and applications.

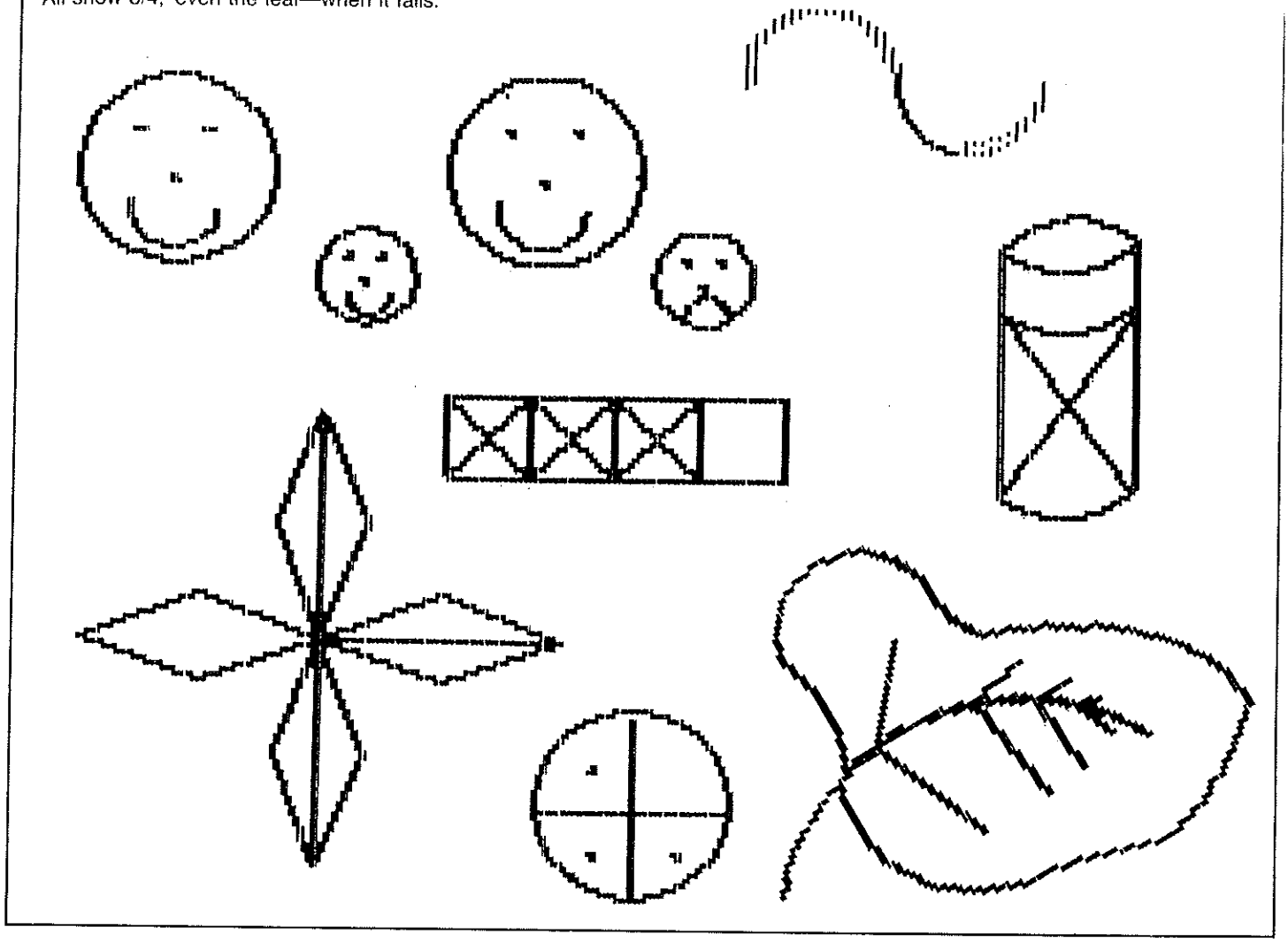
Recommendations

Two summer experiences of intensive work with teachers and the analysis of their products, achievement, and reactions led to the following recommendations:

1. During a two-week session, instruction in one computer language appears to be "enough." Introduction of a second language is likely to cause confusion among participants.

2. Given this constraint, and because of its potential as a structured programming language with graphical and editing features, Logo is the most appropriate language for introducing elementary and middle school teachers to computers. Logo is a good starting point for teaching teachers to

Fig. 7
All show 3/4, even the leaf—when it falls.



write programs and is a computer language they feel confident in teaching their students.

3. Because the teachers already brought with them considerable knowledge of programming procedures (expressed in ordinary language) and were very successful in learning Logo, curriculum supervisors should take advantage of Logo's wide accessibility. Logo dialects exist for many brands of microcomputers, including Apple, Atari, Commodore, Franklin, Radio Shack, and Texas Instruments.

4. The two-week session worked well, but to avoid an information overload, especially during the school year, the session could be broken into two one-week sessions. During the first week turtle graphics, procedure writing, and disk management could be taught and complemented by an examination of existing educational software such as MECC's Elementary Volumes. Several small projects could be assigned between the sessions, which could involve interactive programming, words, and lists, to be followed by a larger project or several smaller projects. An alternate schedule could be used during the school year that would involve a series of two-day workshops on weekends.

5. Teachers' enthusiasm for a language and their readiness for implementation of it in the classroom must be balanced by the application of this newly acquired knowledge to a variety of different tasks and must be accompanied by the teachers' reflecting on what they have learned. Also, various instructional uses of computing should be considered, with special emphasis on programming by students. Special assignments could require the teachers to produce sample programming tasks for students involving the material that is usually taught.

References

Abelson, Harold. *Logo for the Apple II*. Peterborough, N.H.: BYTE/McGraw-Hill, 1982.

Davis, Edward J., Jim Helms, and Robert Jensen. *Logo—a Primer for Elementary School Teachers*. Athens, Ga.: University of Georgia, Department of Mathematics Education, 1983.

Markuson, Carolyn, Joyce Tobias, and Tom Lough. "Logo Fever: The Computer Language Every School Is Catching." *Arithmetic Teacher* 31(1983):48–51.

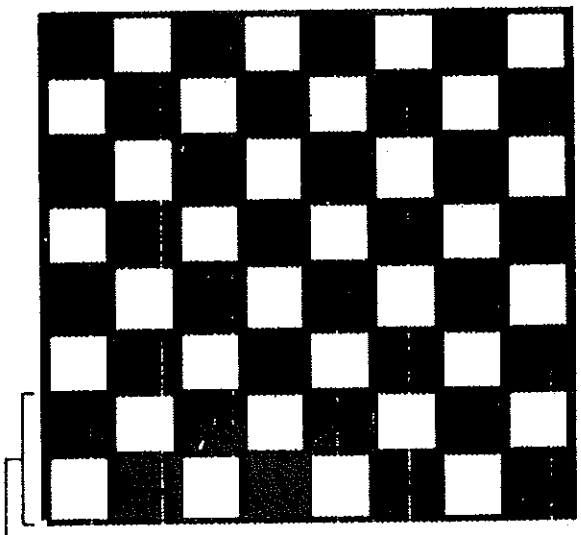
Papert, Seymour. *Mindstorms: Children, Computers, and Powerful Ideas*. New York: Basic Books, 1980.

Watt, Daniel. *Learning with Logo*. New York: McGraw-Hill, 1983.

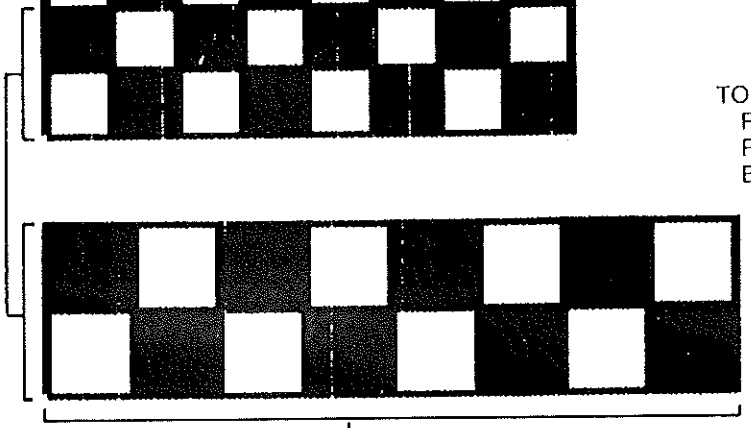
(Continued on next page)

Appendix 1 Top-Down Programming from Figure 5

In this example of top-down analysis, one mentally decomposes the checkerboard into successively smaller pieces until its simple, basic components are discerned. The programming process begins with these basic components—procedures for SQUARE and SHADESQUARE. These components are then repeated, reoriented, and fitted into more complex components until the checkerboard is complete.



```
TO BOARD
  REPEAT 4
  [BLOCK PU FD 40
   RT 180 PD]
  END
```



```
TO BLOCK
  ROW RT 180
  ROW
  END
```



```
TO ROW
  REPEAT 4
  [DOMINO]
  END
```

```
TO DOMINO
  SHADESQUARE SQUARE
  RT 90 FD 20 LT 90
  END
```

```
TO SQUARE
  REPEAT 4  [FD 20 RT 90]
  END
```

```
TO SHADESQUARE
  REPEAT 10  [FD 20 RT 90 FD 1 RT 90 FD 20
                          LT 90 FD 1 LT 90]
  END
```

Appendix 2

Programs for Figure 1

```
TO SQ : X
  REPEAT 4 [FD :X RT 90]
END
TO JR :X
  RT 90 FD :X LT 90
END
TO GROW
  HT SQ 20 SQ 40 SQ 60 SQ 80
END
TO BLOCKS
  HT SQ 20 JR 20 SQ 40 JR 40 SQ 60
END
TO TRUCK
  HT SQ 80 JR 10 RT 90 SQ 10 LT 90 JR 70
  SQ 40 RT 90 FD 15 SQ 10
END
TO FACE
  HT SQ 80
  PU FD 50 JR 10 PD SQ 15
  PU JR 45 PD SQ 15
  PU JR ( - 20 ) BK 15 PD SQ 10
  PU BK 25 JR ( - 15 ) PD REPEAT 4 [SQ 10
      JR 10]
END
```

Programs for Figure 2

```
TO CIR :X
  REPEAT 36 [FD :X RT 10]
END
TO BALLOON
  HT FD 35 LT 90 CIR 2
END
TO BALLOONS
  HT
  REPEAT 3 [BALLOON RT 90 BK 35 RT 45]
  LT 90 BK 20
END
TO MAN
  BALLOON LT 90 FD 35 RT 45 FD 25 BK 25
      LT 90 FD 25 BK 25 LT 135 FD 20 LT 80
      FD 25 BK 25 RT 160 FD 25
END
TO B.MAN
  MAN LT 80 FD 20 LT 45 BALLOONS
END
```

Programs for Figure 3

```
TO SPIRAL :X
  FD :X RT 90
  SPIRAL :X + 5
END
TO SPIRALH :X
  FD :X
  RT 60
  SPIRALH :X + 3
END
```

Programs for Figure 4

```
TO CIR :X
  REPEAT 36 [FD :X RT 10]
END
```

```
TO RECT :C :L
  REPEAT 2 [FD :C RT 90 FD :L RT 90]
END
TO DOOR
  HT RECT 30 20 PU SETXY 45 15 PD CIR .5
END
TO WINDOW
  RECT 20 15
  FD 10 RT 90 FD 15 BK 7.5 LT 90 FD 10 BK
      20
END
TO WINDOW2
  REPEAT 2 [FD 20 RT 80 FD 12 RT 100]
  FD 10 RT 80 FD 12 BK 6 LT 80 FD 10 BK 20
END
TO HOUSE
  HT PU SETXY 10 0 PD
  SETXY 10 70 SETXY 40 100 SETXY 100 115
      SETXY 135 85
  SETXY 135 15 SETXY 80 0 SETXY 80 70
      SETXY 135 85 SETXY 80 70
  SETXY 40 100 SETXY 80 70 SETXY 10 70
      SETXY 10 0 SETXY 80 0
  PU SETXY 15 40 PD WINDOW
  PU SETXY 60 40 PD WINDOW
  PU SETXY 35 0 PD DOOR
  PU SETXY 88 42 PD WINDOW2
  PU SETXY 115 48 PD WINDOW2
END
```

Programs for Figure 5 Are in Appendix 1

Programs for Figure 7

```
TO CIRCLE.DOT
  HT
  REPEAT 36 [FD 4 RT 10]
  RT 90 FD 46 BK 23 LT 90 FD 23 BK 46
  FD 13 PU RT 90 FD 10 PD DOT
  PU BK 20 PD DOT
  LT 90 PU FD 20 PD DOT
END
TO BOXES
  REPEAT 2 [FD 20 RT 90 FD 80 RT 90]
  REPEAT 3 [RT 45 FD 28.28 LT 45 BK 20 LT
      45 FD 28.28 BK 28.28 RT 45]
END
TO FAN
  SETH 0 REPEAT 4 [D RT 90]
  REPEAT 3 [FD 55 BK 55 RT 90]
  SETH 0
END
TO ROPE
  PC 5
  REPEAT 18 [FD 3 RT 10]
  REPEAT 18 [LT 10 BK 3]
  RT 90 FD 2 LT 90
  REPEAT 18 [FD 3 RT 10]
  PC 1
  REPEAT 9 [FD 3 LT 10]
  PC 5
  REPEAT 9 [FD 3 LT 10]
  REPEAT 9 [RT 10 BK 3]
  SETH 0 FD 2 RT 90
  REPEAT 9 [FD 3 LT 10]
  SETH 0
END
TO BIG.SMILEY
  HT
  REPEAT 36 [FD 4 RT 10]
  RT 90 PU FD 22 PD
  DOT PU SETH 0 FD 12 RT 90 FD 8 PD DOT
  PU BK 16 PD DOT SETH 0
  PU BK 18 LT 90 FD 2 PD SETH 180
  REPEAT 18 [FD 1.8 LT 10]
  SETH 0
END
TO CAN
  FD 60 RT 45
  CARC RT 90
  CARC
  LT 135 FD 15 SETH 45
  PU CARC PD RT 90 CARC
  LT 135 FD 45 SETH 45
  PU CARC SETH 0 FD 60 PD RT 180
  FD 60 RT 45 CARC SETH 0
  SETXY 32.33 45 PU SETH 180 FD 45 PD
  SETXY 0 45 PU SETH 0 BK 45 PD HT
END
TO CARC
  REPEAT 9 [FD 4 RT 10]
END
TO D
  LT 20 FD 30 RT 40 FD 30
  RT 140 FD 30 RT 40 FD 30
  RT 160
END
TO HC
  OUTDEV 1
  ( PRINT1 CHAR 9 "G "E "R "D CHAR 13 )
  OUTDEV 0
END
TO STEM
  HT
  LT 25 REPEAT 12 [FD 2 RT 2]
  REPEAT 90 [FD 1 RT 1]
  REPEAT 20 [BK 1 LT 1]
  LT 25 FD 7 BK 7 RT 50 FD 12 BK 12
  LT 25
  REPEAT 10 [BK 1 LT 1] LT 30 FD 9 BK 9
  RT 80 FD 22 BK 22 LT 50
  REPEAT 15 [BK 1 LT 1]
  LT 30 FD 15 BK 15
  RT 90 FD 30 BK 30 LT 60
  REPEAT 25 [BK 1 LT 1]
  LT 45 FD 25 BK 25 RT 120 FD 35 BK 35
  LT 75 REPEAT 10 [BK 1 LT 1]
END
TO LEAF
  STEM
  LT 45 REPEAT 7 [FD 2 LT 9 FD 2 RT 2]
  RT 5
  REPEAT 10 [FD 1 RT 4]
  REPEAT 15 [FD 1 RT 10 FD 2 LT 1]
  REPEAT 9 [FD 1 RT 2]
  REPEAT 15 [FD 1 LT 3]
  LT 25
  REPEAT 10 [FD 1 RT 1]
  REPEAT 15 [FD 2 RT 1]
  REPEAT 10 [FD 3 RT 5]
  RT 45
  REPEAT 10 [FD 2 RT 1]
  REPEAT 10 [FD 2 RT 3]
  RT 25
  REPEAT 15 [FD 2 LT 1]
  REPEAT 20 [FD 3 RT 5]
END
TO FROWN
  HT
  REPEAT 36 [FD 2 RT 10]
  RT 90 PU FD 11 PD
  DOT PU SETH 0 FD 6 RT 90 FD 4 PD DO
  PU BK 8 PD DOT SETH 0
  PU BK 15 PD SETH 0
  REPEAT 18 [FD .9 RT 10]
  SETH 0
END
TO DOT
  REPEAT 36 [FD .1 RT 10]
END
TO SMALL.SMILEY
  HT
  REPEAT 36 [FD 2 RT 10]
  RT 90 PU FD 11 PD
  DOT PU SETH 0 FD 6 RT 90 FD 4 PD DO
  PU BK 8 PD DOT SETH 0
  PU BK 9 PD SETH 180
  REPEAT 18 [FD .9 LT 10]
  SETH 0
END
```