ISSN 0319-8367

delta-k

JOURNAL OF THE
MATHEMATICS COUNCIL
OF THE ALBERTA
TEACHERS' ASSOCIATION

Volume 27, Number 1

June 1988

TECHNOLOGY AND MATHEMATICS

delta-k

JOURNAL OF THE
MATHEMATICS COUNCIL
OF THE ALBERTA
TEACHERS' ASSOCIATION

Volume 27, Number 1

June 1988

Contents

EDPRESS

delta-K is published by The Alberta Teachers' Association (ATA), for the Mathematics Council (MCATA). Editor: John B. Percevault, 2510 22 Avenue South, Lethbridge, Alberta T1K 1J5. Editorial and production services: Central Word Services staff, ATA. Copyright © 1988 The Alberta Teachers' Association, 11010 142 Street, Edmonton, Alberta T5N 2R1. Permission to use or reproduce any part of this publication for classroom purposes, except for articles published with permission of the author and noted as "not for reproduction," is hereby granted. Opinions expressed herein are not necessarily those of the MCATA or the ATA. Please address all correspondence regarding this publication to the editor. *delta-K* is indexed in the Canadian Education Index.

EDITOR'S COMMENTS

Welcome to the second issue of *delta-K* devoted tò the topic of technology and mathematics. In this issue, J. Dale Burnett's guest editorial discusses the importance of values in technology and mathematics. An article by Kate Le Maistre encourages teachers to use television, particularly the VCR, to enhance the teaching of mathematics. Marlow Ediger offers guidelines for using microcomputers to teach mathematics.

The articles by Francis Sommerville and Ron Taylor present microcomputer programs using BASIC and Logo respectively to solve the same mathematical problem. Scott Erickson discusses both the graphic art of M.C. Escher and the technology applied to the deformation of tessellations. J. Dale Burnett uses psychological theory to design practice sessions on the computer. George Cathcart shows how Logo programs can be used to perform arithmetic tasks. Barry McGuire explains how a computer simulates science studies and experiments in a school-designed scientific studies and computer course.

''Student Problem Corner'' features a mathematical exercise reprinted from *Resource Problems to Enhance the Teaching of Mathematics,* an algebraic solution and computer program published by the Department of Mathematical Sciences, University of Delaware. In addition to ''Student Problem Corner,'' a new section entitled ''Student Problem Solvers'' has been added. Will it become a regular component of *delta-K?* Can students develop an alternate algorithm or an algorithm using a different language? Solutions submitted to ''Student Problem Corner will be published in the next issue.

John B. Percevault

GUEST EDITORIAL

Technology and Mathematics: The Importance of Values

J. Dale Burnett

An example is worth a thousand pictures. Recently, I had the opportunity to observe a Grade 8 mathematics class in a computer lab environment. There were about 30 microcomputers, with a student at each one. The students were engaged in a math drill program requiring rapid responses to a series of computational problems of increasing difficulty. The screen was filled with fancy, colorful graphics as the students worked through the various levels of the scenario. About a quarter of the students were not involved. While the computer was meticulously measuring their response time to a thousandth of a second, they were actively engaged in conversation with their neighbor or close friend. (Imagine a researcher, removed from the classroom, interpreting those lengthy delays!) Most of the other students were giving the task their attention. For many, the importance of obtaining another 10,000 points was all-important. By most criteria, the latter three-quarters of the class would be considered good students. They were enjoying—and in a few cases, really enjoying—the activity set for them. This was what I found most upsetting.

We all have dreams. We all have visions of an ideal educational system. My visions do not include the above scenario. My comments are not directed to the teacher but to the students, particularly the majority who were enjoying the exercise. They shouldn't have! And though my comments are directed at students, my concern lies with the educational enterprise. I am concerned about an educational system in which students enjoy such activities. I am not opposed to drill. Far from it. There is clearly a place for such practice, and this classroom may well have been such a place.

Nonetheless, I am using the example to suggest something about the appropriate use of technology in education. The simple fact that a computer *can* be used in a certain manner is no reason to suggest that it *should* be used in that manner. More fundamental issues, pedagogic and conceptual, need to be addressed first. One of my criteria for a desirable educational activity, with or without computers, is that students should be actively thinking. Another is that students should have the opportunity to exhibit self-control and independence. A third is that students should work within a sharing environment, where ideas and suggestions are freely communicated among the participants.

I would like to contrast the above scenario with one that occurred in the same room during the next period. This time it was a Grade 5 class. Students were each working on a task of their own choosing within a Logo environment. To me, the differences were dramatic. First, the level of excitement and enthusiasm was contagious; the students were proud of their work and had good reason to be. Second, the students were actively engaged in a myriad of problems and subproblems—their minds were in high gear as they reasoned their way through the various implications of their actions. Third, although the students may not have realized it, they were doing mathematics, in contrast to remembering mathematics. In the case of Logo, the glitter was in the students' minds (and eyes) rather than on the screen. We need to distinguish between the enjoyment that comes from using a new device and the enjoyment that comes from understanding, for example, the pattern behind the concept of a square, or the relationships among a square, a hexagon and a circle. A Logo screen display is relatively barren, but it can be deeply satisfying.

Among the goals of the senior high school mathematics program are these:

1. To develop in each student a positive attitude toward mathematics
2. To develop the ability to use mathematical concepts, skills and processes
3. To develop the powers of logical analysis and inquiry
4. To develop an ability to communicate mathematical ideas clearly and correctly to others

The goals of the elementary mathematics program are similar. The program is intended to ''foster within the learner a sense of accomplishment and success, a positive attitude toward mathematics and a positive attitude toward learning.'' However, there is often a substantial gap between these goal statements and the skill objectives that are then formulated, avowedly to achieve these goals. Our concern for measurable skills and behaviors has the unfortunate tendency to pull us away from our goals—an embarrassing paradox.

I believe that these Grade 5 students were actively involved in a setting that is highly consistent with all of these goals. I am less confident about the activities of the Grade 8 class. The choice is ours. The critical issue is not technology but how the classroom teacher plans to use the technology. I would like to see the computer used more often in a way that is consistent with the goals just listed.

If This Is Television, Shouldn't My Intelligence Be Insulted?

Kate Le Maistre

Kate Le Maistre is mathematics and science consultant with Jerome-Le Royer School Commission, English Educational Services, Ville d'Anjou, Quebec. Le Maistre presented a paper to the NCTM Canadian Conference held in Edmonton in 1986.

The Freedom Machine

Those of us who have the responsibility of meeting a class of children up to 200 times a year and teaching them a subject like mathematics need help from time to time. My favorite aid is the videocassette recorder (VCR).

The VCR can free us from complicated technology and from network schedules. Even the manufacturers of VCRs have been surprised by the popularity of the flat box that so many of us store underneath the television set. Two factors that have contributed to this popularity are the convenience and ease of use of the VCR. We can ''time-shift'' or watch programs at a time convenient to us, not when the networks feel that we should watch them. This is important to a classroom teacher, who can show some or all of a program when it fits into the lesson plan, not necessarily when it is being transmitted.

If you have not yet bought a VCR for your home, the total inservice time needed to train the most unmechanical person to insert, play, rewind and eject a tape is about five minutes. In seven minutes, you can find how to fast forward, pause, put a transparency on the screen, trace the picture for later use on an overhead projector and probably come up with several innovative ideas of your own.

If all else fails, ask one of the children in your class to show the tape; after all, your students are growing up with VCRs as we did with radios.

Compare the ease of operating a VCR with the agony we used to go through in setting up a 16 mm movie. I thought life would be easy when my school bought a self-threading projector, but, although I didn't tear the sprocket holes from as many films, when I stopped the film to ask a question or to explain a point, someone near the projector was bound to call out: ''Miss! I can smell something burning!''

How Can I Compete with ''Miami Vice''?

A recent study found that ''young people between the ages of six and 11 watch, on the average, 27 hours of television a week'' (Merrow 1985). As teachers, we cannot control what children watch at home—and there are times when we all like to spend an hour or two looking uncritically at television. While our students are spending so much time using television as recreation, they are not reading books, yet our teaching methods remain textbook-based. I am not suggesting that because children do not read books for entertainment, we should not use books at school. Quite the contrary; part of our role is to encourage students to use as many sources of information as possible, and any good teacher will use as many strategies as possible to get through to students. We are hiding our heads in the sand if we do not use a strategy that is both familiar and attractive to students.

''Talk'' and ''chalk,'' closely followed by worksheets, seem to be the basics of the teacher's arsenal. Apart from the occasional overhead projector, the tools in many classrooms have not changed appreciably since the days of the slate and pencil. Yet we are trying to use these archaic tools with a generation of students who are often more technologically sophisticated than we are.

The most recent mathematics programs available on instructional television contain what the professionals call "production values," or what we might call "razzle-dazzle," attention-grabbers or—shades of teachers' college—motivators. These programs include action shots of children of the same age as the intended audience, animation, outside locations, simulations and computer-generated graphics. If anything is going to compete with the glitzy productions of the big networks, these programs stand a good chance.

I Thought We Were Supposed to Encourage Active Participation

Here is a typical scenario for using videotapes in a mathematics class: A Grade 5 teacher finds thai her classes on the comparison of decimals have not been as successful as usual. She looks at the teachers' guide to *Mathworks* and decides that the title of program 15 is promising. The program summary describes a sequence in which two girls are running a race, timed to hundredths of a second; an animation sequence involving the Three Bears; another short story in which two boys discover that 0.5 kg of meat is more than 0.33 kg; a gemologist explaining why decimals are important in his work; and a summary sequence in which two girls apply their knowledge of decimals.

The teacher then previews the tape to decide where to place it in her lesson plans. Should she use the whole program? Some scenes only? Should she do the suggested activities before showing the tape? At some point during the program? As a follow-up? Should she try to fit the entire 15-minute program into one class? Are there enough concepts to make several lessons?

Next, she prepares the program activities in the teachers' guide as well as any others that are appropriate, copies any assignment sheets and collects necessary materials. Again the teachers' guide is useful because it provides a black-line master and lists of simple materials used in the activities.

At this point, the only thing left is to show the tape and watch the children's faces. If you've never done this before, be prepared for a pleasant surprise.

I am not suggesting that playing videotapes should be the only, or even the major, activity in mathematics classes. To suggest as much would mean simply replacing one provider of information (the teacher) with another (the "box"). Rather, I am suggesting that a videotape can act as a motivator, as an introduction to a topic, as a unit review, as one more version of the same information, as a tutorial for an absent student—in fact, as one more tool in the teacher's toolbox.

After all, when you're a teacher in front of a class, isn't it reassuring to have a variety of tools in the toolbox?

Suggested Mathematics Programs

"Two Plus You"
"Math Patrol"
"Math Patrol 2"
"Math Patrol 3"
"It Figures"
"Landscape of Geometry"

Check the ACCESS listings for other mathematics programs available on instructional television.

Reference

Merrow, John. "Children and Television: Natural Partners " *Kappan*, November 1985, p. 212.

Mathematics Education and Technology

Marlow Ediger

Dr. Ediger is professor of education at Northeastern Missouri State University, Kirksville. He was the 1985 recipient of the Project Innovation's Merit Award for Excellence in Teacher Education. He is a regular contributor to delta-K.

Much has been written about using technology to teach mathematics. Such semi-concrete audiovisual aids as slides, films, filmstrips, transparencies and single-concept film loops have been used for several decades, and their use is increasing.

Prior to the advent of semi-concrete materials, mathematics educators advocated using concrete materials in the classroom. Objects and items used to teach stress the concrete. Students learned by relating the real world of objects to abstract ideas in mathematics.

In addition to using semi-concrete materials, the mathematics teacher should emphasize abstract learning. Reading materials and student-teacher interaction help students move from the concrete to the semi-concrete to the abstract.

Computers in the Curriculum

Computers are a relatively recent development in teaching mathematics. The principles of educational psychology should guide the teacher in selecting software for the curriculum. First, the software and computer should provide learning opportunities that are interesting and that secure the student's attention. Time on task is significant. Absent students waste their own time, as well as the time of the teacher. A major concern with technology is whether it attracts and maintains student interest.

Second, software should be meaningful to the learner. Students need to understand sequential content. The subject matter must make sense; if it is too complex, students will not be able to understand it. If it is too easy, students will learn nothing. Software must provide students with realistic goals.

Students need reasons for learning. They should find value in the content of computerized instruction. Although students perceive the content in mathematics to be worthwhile, they frequently fail to find value in the objectives emphasized in software content. Teachers need to evaluate new software to see if it contains striking content and reflects worthwhile objectives. The chosen objectives should guide software selection.

Slow, average and fast learners need to be respected for the abilities they possess and the levels of learning they have attained. The teacher must identify objectives, provide learning opportunities and select appraisal methods that assist each learner to achieve as much as possible. Software that provides for slow, average and fast achievers needs to be made available.

Regardless of abilities, students must achieve cognitive, affective and psychomotor objectives to the best of their abilities. First, cognitive process, which stresses intellectual tasks, is important. Students need to achieve structural ideas, major generalizations and higher-order thinking skills. Second, the affective or attitudinal category must also be emphasized. Students having healthy attitudes will likely enjoy and appreciate mathematics. A third objective is to improve psychomotor functions. Students require good eye-hand coordination, coupled with gross and fine-motor movement to construct models, designs and figures in mathematics.

Students need feedback on ongoing and completed work. Feedback informs the students about their progress, alerts them to what is incorrect and shows them how to remedy identified deficiencies. Computers and software provide students with immediate results to responses made sequentially. If students do not receive feedback, they may make identical or similar errors. Allowances should be made for trial and error—this learning process teaches students to generalize.

Students should be provided with both deductive and inductive learning opportunities. Teachers should select software that helps students to learn a new process through deduction. Students need opportunities to discover and find out on their own—the thrill and excitement of discovery are extremely valuable.

Students need opportunities to apply what they have learned. Objectives achieved by students using software should reflect the level of application. The applications can be made within the framework of sequential experiences contained in the software program. Applications can also be stressed by using mathematics textbooks, workbooks, worksheets, as well as actual problems in society. Mathematics is dynamic, utilitarian and functional in the day-to-day lives of students.

Selecting Software

Teaching materials should be evaluated before, during and after use; software should be thoroughly and continually evaluated. Which criteria for evaluation should then be applied in teaching-learning situations?

Software must assist students to achieve objectives and should be related to material taught in a lesson or unit. Content in software must relate to preceding subject matter. Unrelated facts, concepts and generalizations might well confuse students. High quality instruction and depth learning by students, rather than survey approaches, should be undertaken. Carefully chosen software directly related to objectives in the curriculum can aid students to understand subject matter in-depth. Depth teaching is not a process adding new content to previous subject matter. Rather, depth teaching forces students to relate the new ideas gained from using computers to their previously acquired learning. Depth learning focuses on helping students to grasp structural ideas and major generalizations more fully and comprehensively. Attitudes toward learning mathematics should become more positive as students attach increased meaning to acquired subject matter.

Software must present the content sequentially so that students perceive order and structure. Students in the classroom need to experience success, not failure. Properly sequenced software ensures that students have success and provides a foundation on which new subject matter can be learned. Therefore, writers of mathematics programs must test and develop sequence.

Software content should promote successful learning. Success reinforces what students have learned, and subject matter, in turn, is learned more effectively with reinforcement. Everyone desires to be successful in life—students are no exception.

As with sequence testing, the success rate of the program needs to be field-tested adequately. If the steps involved in learning are spaced too far apart, then the programmer should add items to the program so that students can respond correctly more frequently.

Students should be given the opportunity to respond frequently to questions and multiple-choice items within a program. Often, required reading leaves the student with very little responding time in a software program. Frequent interactions are recommended so the students receive feedback to their responses.

Drill and practice are necessary activities. Carefully selected software should guide students to review what was learned previously. If drill and practice are not stressed, students tend to forget the subject matter. Retaining information is important in mathematics.

Other software deals with games and gaming. In a game situation, two to four students can be involved in wholesome competition. Each student tries to gain as many points as possible by answering mathematics questions on the monitor. The student with the most points wins. Easier questions are worth fewer points than are more complex ones.

A fourth type of software covers simulations in which students play roles in solving real problems. Compared with day-to-day situations in school or home, simulations provide low-risk situations for students. Higher levels of thinking must be stressed in simulated programs.

Diagnostic and remedial programs may be employed in the mathematics curriculum. Diagnosis pinpoints specific difficulties that a student has in mathematics. Based on diagnosis, remedial programs follow.

Along with software related to ongoing lessons and units in mathematics, additional learning activities and resources must be available to students. These resources include textbooks, workbooks, filmstrips, slides, films, transparencies, overhead projectors, illustrations, videocassette tapes and single-concept film loops.

To help students, mathematics teachers must follow recommended principles of learning. These principles stress ways of guiding learners to attain as much as possible. They are applicable regardless of the aids used to teach mathematics. Aids include computers as well as textbooks and audiovisuals. Teachers should employ drill and practice, tutorials, games, simulations, and diagnosis and remedial techniques. Ultimately, each student's goal should be to achieve as much as possible in mathematics.

Programming: A Subset of Problem Solving

Francis Sommerville

Francis Sommerville teaches mathematic and computer courses at E.P. Scarlett High School, Calgary. Sommerville served as an executive member of MCATA and most recently as co-editor of the MCATA newsletter.

Computer programming effectively extends problem solving activities in the mathematics classroom. The computer forces a systematic approach, provides immediate feedback and appeals to most students.

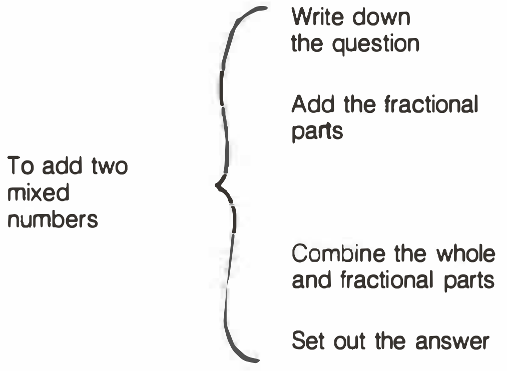
Unfortunately, many programming activities are only marginally related to the responsibilities of junior high math teachers. In his book *Mindstorms: Children, Computers, and Powerful Ideas*, Seymour Papert claims that when children program computers they ''establish an intimate contact with some of the deepest ideas from science, from mathematics, and from the art of intellectual model building'' (Papert 1980, 5). This sounds great, but will they pass the math final?

An approach is required that allows for group instruction, provides for individual differences and draws examples from the standard mathematics curriculum.

Consider the task of adding mixed numbers. Although not a problem in the usual sense, the task can be approached as a problem solving activity. Some students require extensive review while others are ready to solve the problem in the general sense using variables.

''How do you add mixed numbers?'' In response to this question, students might suggest following steps such as writing down the question, adding the fractional part, combining the whole and fraction part and setting out the answer. These responses reflect the usual heuristics of thinking of a similar problem or, dividing the problem into subproblems. These steps could be summarized on the board. (See Figure 1.)

Figure 1. Adding Mixed Numbers

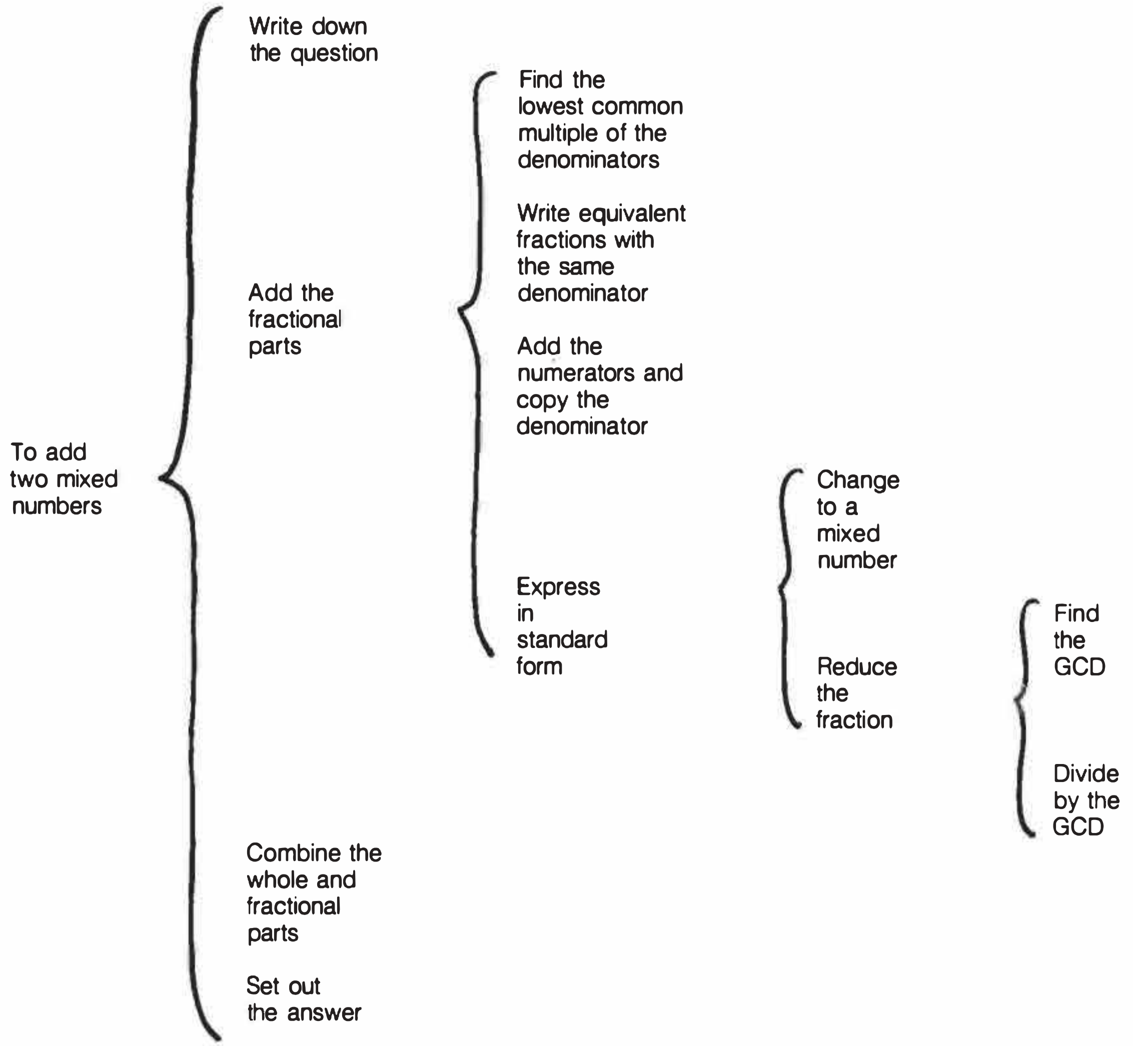


This style of representing the students' suggestions is described by Higgins (1979) and is called a Warnier/Orr diagram. Set notation shows the division of a task into its components. When read from left to right, the diagram explains how to add mixed numbers. The tasks are listed in order from top to bottom. Reading from right to left explains why a certain process is required. Students could be asked to expand on the more difficult aspects of the process to produce a diagram similar to Figure 2.

Figure 2 provides an overall picture and may help students identify areas that require review. As examples are completed, the teacher refers to the stage at which the students are working. This process also provides a sound basis for the student who is ready to solve the problem of adding mixed numbers in the general sense, using algebra.

A major thrust of the junior high mathematics program is to move students from solving specific examples to solving general problems. The general

Figure 2. Overall Picture of Process



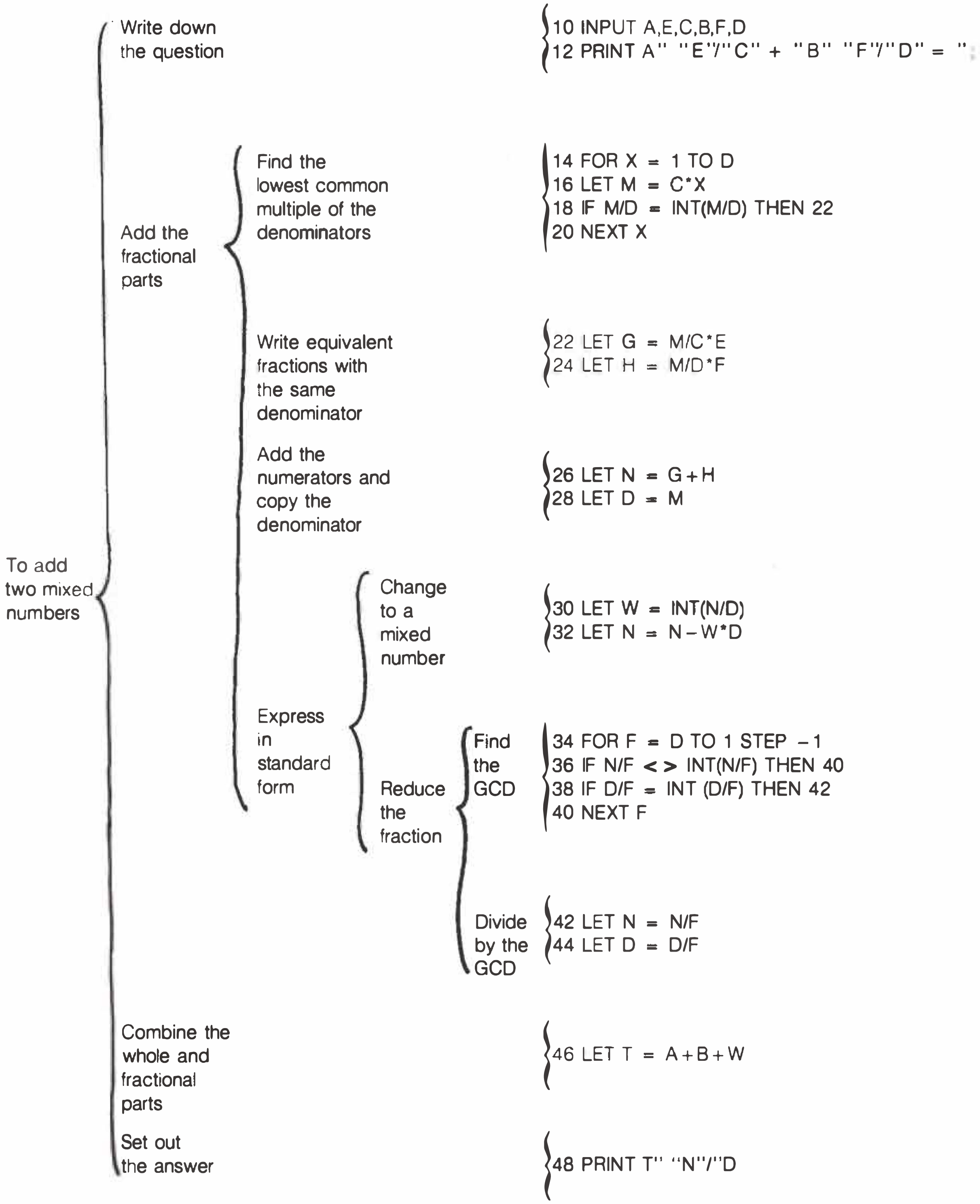
solution can be expressed verbally, algebraically or, more recently, as a computer program. Probably all three can be used effectively by some students; therefore a conscious effort to structure examples in a manner that promotes transfer to all three modes of expression is needed. Consider the subproblem identified previously, that of finding the Lowest Common Multiple (LCM) of the denominators.

Once students demonstrate the ability to find the LCM for specific examples, they might be led to a general solution by asking such questions as these (answers are given in parenthesis).

To find the LCM of C and D let us first consider multiples of C. What is the first multiple of C? (C or C times 1.) What is the second multiple of C? (C times 2.) What is the third multiple? (C times 3.) Now, what number is a multiple of both C and D? (C times D.) Is this the lowest common multiple? (Not necessarily.) How could you check to see if there is a smaller multiple of both C and D? (By testing each multiple of C to see if it is a multiple of D.)

This approach transfers to programing in BASIC in the following manner:

Figure 3. Warnier/Orr Solutions



To add two mixed numbers

Write down the question

Add the fractional parts

Find the lowest common multiple of the denominators

Write equivalent fractions with the same denominator

Add the numerators and copy the denominator

Express in standard form

Change to a mixed number

Reduce the fraction

Find the GCD

Divide by the GCD

Combine the whole and fractional parts

Set out the answer

```
10 INPUT A,E,C,B,F,D
12 PRINT A"  "E"/"C" + "B"  "F"/"D" = ";

14 FOR X = 1 TO D
16 LET M = C*X
18 IF M/D = INT(M/D) THEN 22
20 NEXT X

22 LET G = M/C*E
24 LET H = M/D*F

26 LET N = G+H
28 LET D = M

30 LET W = INT(N/D)
32 LET N = N-W*D

34 FOR F = D TO 1 STEP -1
36 IF N/F <> INT(N/F) THEN 40
38 IF D/F = INT (D/F) THEN 42
40 NEXT F

42 LET N = N/F
44 LET D = D/F

46 LET T = A+B+W

48 PRINT T"  "N"/"D
```

```
1. FOR × = 1 TO D
2. LET M = C*X : REM M is a multiple of C
3. IF M/D = INT(M/D) THEN 5 : REM Check
   to see if M is a multiple of D and, if it is, go on
   to line 5.
4. NEXT × : REM Repeat lines 2 and 3 with the
   next value of x.
5. PRINT M : REM M is the LCM of C and D.
```

Marcia Linn (1985) calls such programs "templates." She defines templates as "stereotypic patterns of code using more than a single language feature. They are employed as an entity in programs to perform commonly encountered tasks." In mathematics classes, templates are not necessarily programming code; they could be expressed verbally or algebraically. However, introducing programming code at this level reinforces verbal and algebraic templates.

To complete the initial problem, students must integrate such other templates as finding the greatest common factor and changing improper fractions to mixed numbers. The Warnier/Orr diagram (Figure 3) shows the complete solution. The BASIC program assumes that you are adding the mixed numbers of A E/C and B F/D. (Note that A and B are whole numbers with fractional parts, E/C and F/D respectively.) The program works on Apple or IBM computers. (The input and output commands on lines 10, 12 and 48 are potential problems if this program is attempted on other computers.)

Thus the student solves problems by dividing complex problems into sets of subproblems and solving them by relying on a previously developed repertoire of templates. If programming is part of this process, then students have to learn the features of the language, develop a repertoire of templates and develop the ability to use templates to solve more complex problems.

Clearly programming is not problem solving. It is, however, an exciting subset of the problem solving process as it exits in junior high mathematics classes.

References

Higgins, David A. *Program Design and Construction.* Englewood Cliffs, New Jersey: Prentice-Hall, 1979.

Linn, Marcia C. "The Cognitive Consequences of Programming Instruction in Classrooms." *Educational Researcher,* May 1985, pp. 14–16 and 25–29.

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

Using Logo to Solve Sommerville's Algorithm for Adding Mixed Numbers

Ron Taylor

Ron Taylor holds a B.Sc. in animal biology and a B.Ed. in secondary science. He taught Grades 1 through 12 in Rocky View School Division No. 41. He is currently studying for a master's degree in education at the University of Lethbridge and is interested in the phenomenology and pedagogy of play.

One of the important aspects of problem solving that we teach students is to look back at a successful solution and determine if the problem can be solved in another manner. What follows is just such an attempt. In this case, the essential algorithm, developed by Francis Sommerville (1987), has not been changed. The intention here is to show how the algorithm can be implemented in another language, Logo.

The Logo language has several advantages over traditional versions of microcomputer BASIC, among them extensibility and meaningful variable names. By extensibility, I mean that the language allows students to incorporate task names (see Figure 1) directly into the program. Using meaningful variable names helps students to avoid the confusion that sometimes results when many variables are used. The result is a longer program, but one that, with practice, may help the student to better understand the program or to develop alternate solutions.

Translating a program from BASIC into Logo may seem sacrilegious to many students and teachers familiar with the Logo language. Logo offers the particularly strong problem solving tool of recursion to emulate the loops used in Sommerville's program. The program does not fully exploit the power of the recursive loop but does retain Sommerville's original algorithm. Excellent articles on the use of the recursive loop can be found in the journal *The Computing Teacher*. Cathcart (1987) has recently published an article discussing the use of the recursive loop to generate factors. Readers may wish to develop Logo programs to add mixed numbers that use the Logo language to full advantage.

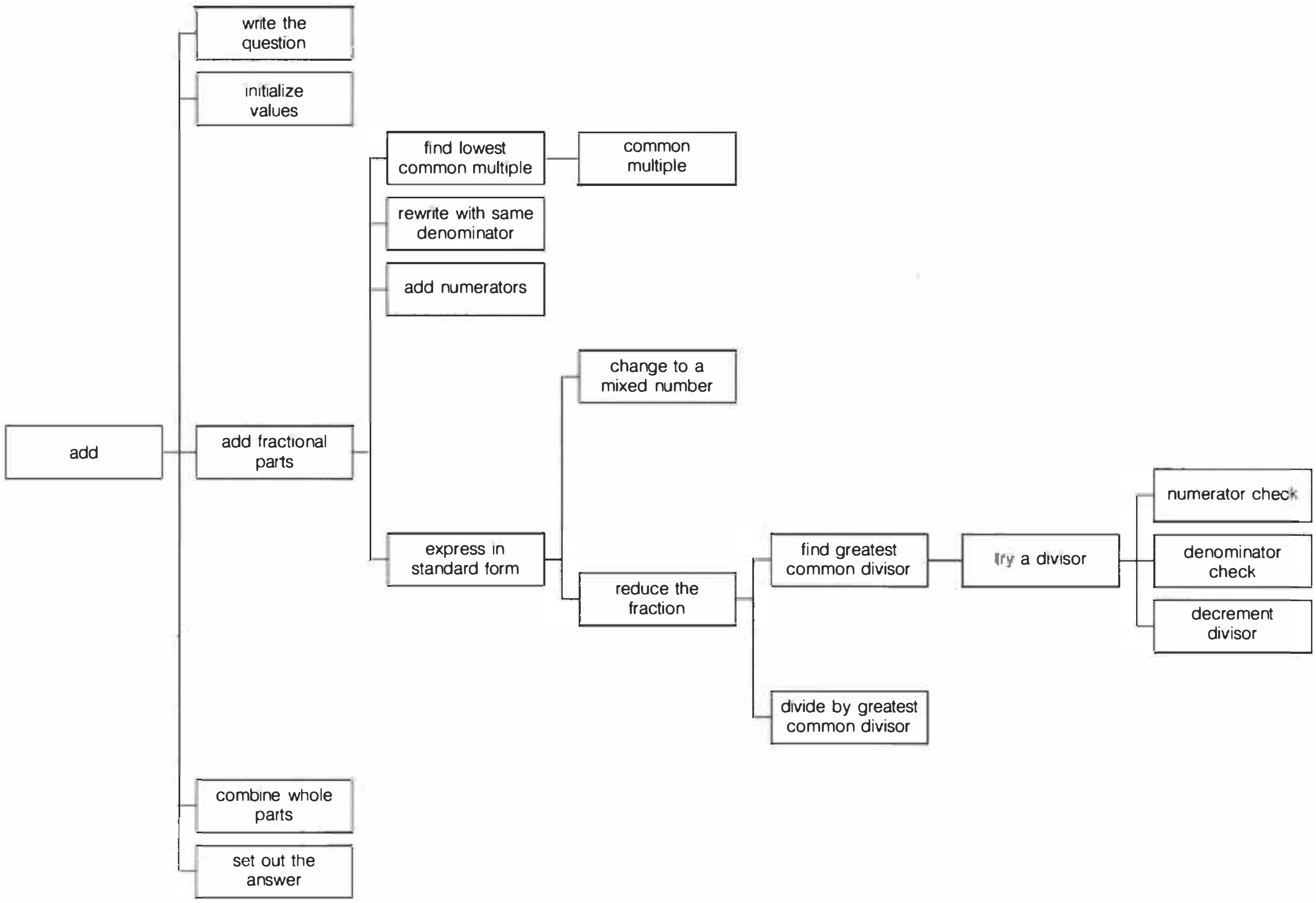
Logo Program

```
TO add :whole1 :numerator1 :denominator1 :whole2 :numerator2 :denominator2
        write.the.question
        initialize.values
        add.fractional.parts
        combine.whole.parts
        set.out.the.answer
END

TO write.the.question
        PRINT (SENTENCE :whole1 :numerator1 [/] :denominator1 [+] :whole2
            :numerator2 [/] :denominator2)
END
```

```
TO initialize.values
        MAKE "sum.of.numerators 0
        MAKE "multiplier 0
        MAKE "equiv.numerator1 0
        MAKE "equiv.numerator2 0
        MAKE "lowest.common.denominator 0
        MAKE "total.whole 0
        MAKE "reduced.denominator 0
        MAKE "check 0
        MAKE "derived.whole.part 0
        MAKE "reduced.numerator 0
        MAKE "not.reduced.numerator 0
        MAKE "test.divisor 0
END

TO add.fractional.parts
        find.lowest.common.multiple :denominator1 :denominator2
        rewrite.with.same.denominator
        add.numerators
        express.in.standard.form
END

TO find.lowest.common.multiple :denominator1 :denominator2
        common.multiple :denominator1 :denominator2
        MAKE "lowest.common.denominator :check
END

TO common.multiple :denominator1 :denominator2
        MAKE "multiplier :multiplier + 1
        MAKE "check :denominator1 * :multiplier
        TEST 0 = REMAINDER :check :denominator2
        IFFALSE [common.multiple :denominator1 :denominator2]
        STOP
END

TO rewrite.with.same.denominator
        MAKE "equiv.numerator1 :lowest.common.denominator / :denominator1 * :numerator1
        MAKE "equiv.numerator2 :lowest.common.denominator / :denominator2 * :numerator2
END

TO add.numerators
        MAKE "sum.of.numerators :equiv.numerator1 + :equiv.numerator2
END

TO express.in.standard.form
        change.to.a.mixed.number
        reduce.the.fraction
END

TO change.to.a.mixed.number
        MAKE "derived.whole.part INT :sum.of.numerators /
            :lowest.common.denominator
        MAKE "not.reduced.numerator :sum.of.numerators - :derived.whole.part *
```

```
               :lowest.common.denominator
END

TO reduce.the.fraction
         find.the.greatest.common.divisor
         divide.by.the.greatest.common.divisor
END

TO find.the.greatest.common.divisor
         MAKE " test.divisor :lowest.common.denominator
         try.a.divisor
         MAKE."greatest.common.divisor :test.divisor
END

TO try.a.divisor
         IF :test.divisor = 1 [STOP]
         IF NOT (AND numerator.check = "true denominator.check = "true)
              [decrement.divisor try.a.divisor]
         STOP
END

TO numerator.check
         TEST 0 = REMAINDER :not.reduced.numerator :test.divisor
         IFTRUE [ OUTPUT "true]
         OUTPUT ["false]
END

TO denominator.check
         TEST 0 = REMAINDER :lowest.common.denominator :test.divisor
         IFTRUE [ OUTPUT"true]
         OUTPUT ["false]
END

TO decrement.divisor
         MAKE "test.divisor:test.divisor - 1
END

TO divide.by.the.greatest.common.divisor
         MAKE "reduced.numerator. INT :not.reduced.numerator /
              :greatest.common.divisor
         MAKE "reduce.denominator INT :lowest.common.denominator /
              :greatest.common.divisor
END

TO combine.whole.parts
         MAKE "total.whole :whole1 + :whole2 + :derived.whole.part
END

TO set.out.the.answer
         PRINT (SENTENCE :total.whole :reduce.numerator [/] :reduced.denominator)
END
```

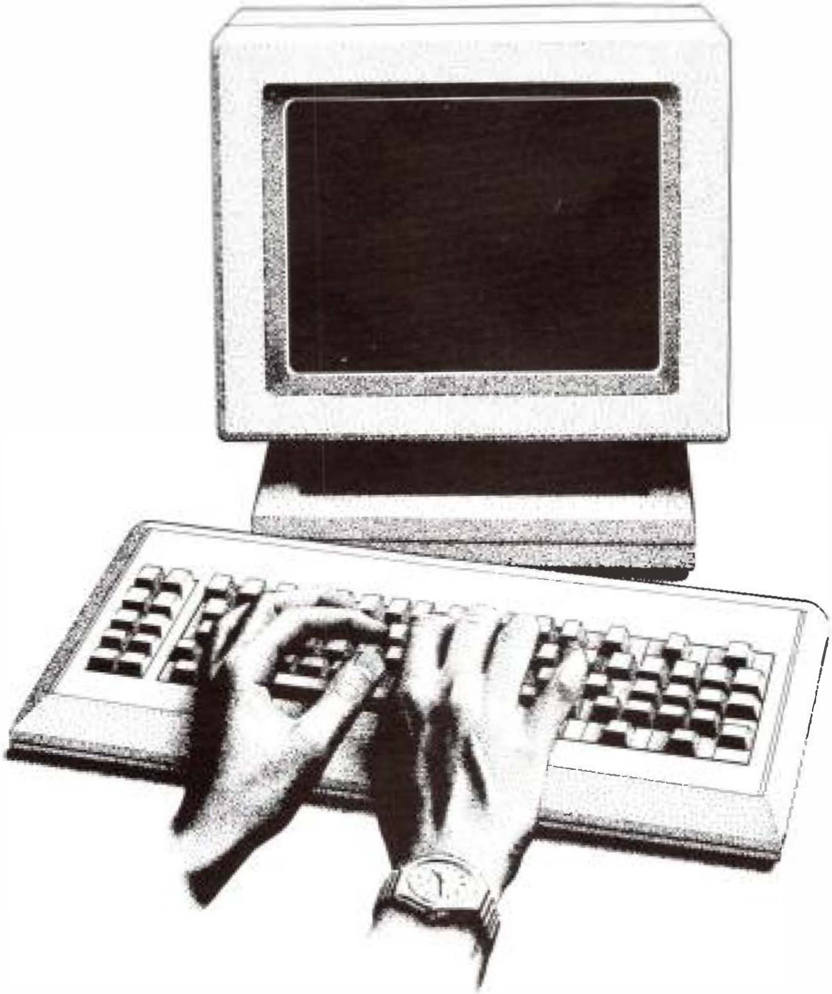
Figure 1. Warnier/Orr Diagram of Task Names



References

Cathcart, W.G. "Generating Factors." *Computers in Education*, February 1987, pp. 28–29.

Sommerville, F. "Programming: A Subset of Problem Solving." *delta-K*.



Escher Revisited: Modeling Gradual Deformations Using Logo

R. Scott Erickson

R. Scott Erickson is a computer specialist for Starland School Division. He graduated from the University of Lethbridge and served as a computer consultant for the City of Lethbridge, as a member of the University of Alberta PLATO Project Team and as a software technician for an accounting systems developer.

Introduction

M.C. Escher was a graphic artist who had little interest in traditional mathematics. However, the complexity of his compositions has interested many mathematicians. Escher used repeating patterns extensively and integrated them into very interesting drawings. The study of these geometric patterns is called "Escher Mathematics."

Many of Escher's drawings are composed using figures called "tessellations." A tessellating shape is one that, when repeated across a plane, will cover the plane entirely without leaving any space. It takes very little experimentation to discover that, when a square is repeated along both axes of a plane, the plane will be covered entirely. Hexagons and triangles are also tessellating shapes.

Escher had the ability to take basic tessellating shapes and modify them according to certain rules so that they become artistically fascinating, yet retain their properties of tessellation. Ranucci and Teeters (1977) explain Escher's rules in detail.

Logo plays an important role in the modeling of Escher-type drawings. Because Logo makes is possible to draw lines using few commands, it is a convenient tool in formulating computer simulations of Escher's work. Perhaps one of the greatest advantages of using Logo to model tessellations is that the mathematical component of the drawing becomes apparent while constructing the simulated drawing. Figure 1 shows three gradual deformations, adapted from Hofstadter 1983.

Rationale

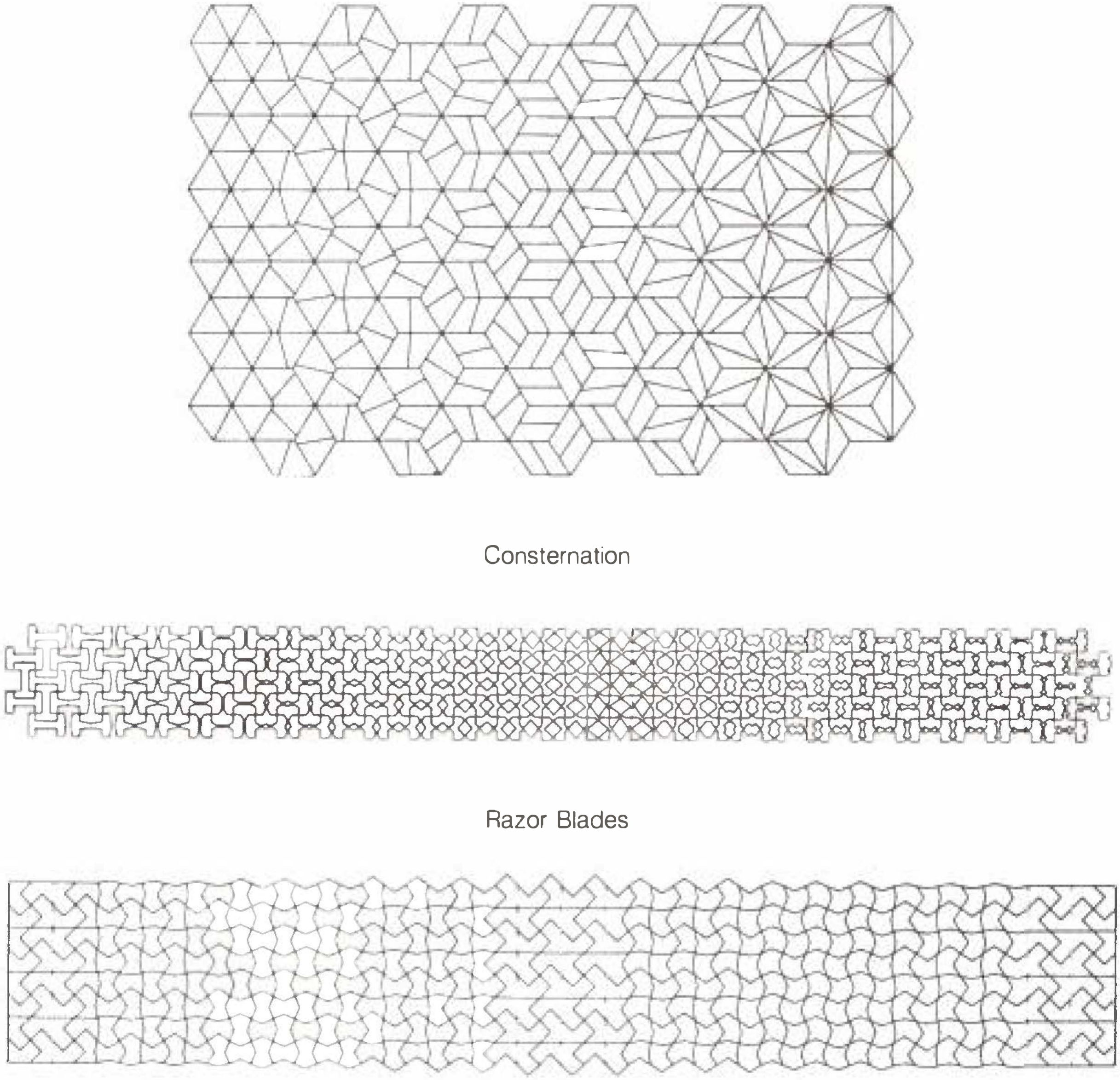
I undertook this research to explore some of the mathematics of gradual deformations in a simple form. Gradual deformations were implemented by Escher in many of his line drawings. I felt that the interesting mathematics of Escher-type drawings would make examining the principles involved an enlightening experience. Additionally, I hoped that the topic might provide enrichment material for advanced or gifted high school students. Having encountered students who are not challenged by the standard curriculum, I am always looking for interesting and demanding activities. As a result, one of my objectives was to evaluate the topic of gradual deformation to see if it could be implemented.

The project was unique; whereas tessellations have received considerable attention since the advent of Logo, gradual deformations have been largely ignored. Therefore, the project provided a takeoff point for studying Escher Mathematics further.

The Problem

The project was to take a square and deform it by having the midpoints of the top and right sides migrate directly away from the centre of the square. The mathematics of this deformation is easier to understand if the problem is broken down and the top side looked at first. The same principles can then be applied to the other axis.

Figure 1. Three Gradual Deformations



Consternation

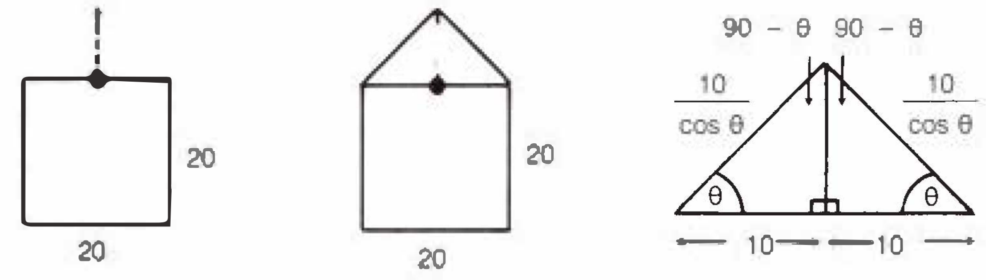
Razor Blades

Parquet Deformation Titled Fylfot Flipflop

The result is the "addition" of a triangle on top of the square (Figure 2). Interesting relationships are established if this triangle is divided into two equal parts.

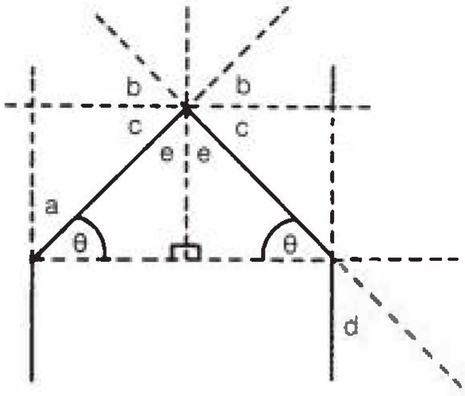
There are two important points on the path that the turtle must take in tracing the deformed shape. The angle the turtle turns at each corner and the distance it travels between corners must be found. Both values can be calculated with trigonometry (Figure 2). Turning the turtle right at the upper left corner of the square at an angle of $90 - \theta$ will point the turtle along the "right" edge of the triangle.

Figure 2. Gradual Deformation of the Top of a Square



The turtle will then travel 10/cos θ units in that direction until it reaches the next corner, which is the apex of the triangle.

Figure 3. Similar Triangles Arising from the Deformation



At this point, a construction of several similar triangles can be used to determine how much the turtle must turn. Figure 3 illustrates the relationships of several triangles. The measure of angle e is 90 − θ, since θ + e = 90. Applying the principles of similar triangles establishes that angles a and e are equal. As well, angle b has the same measure as θ. Angles b and c are vertically opposite, so they must be equal. Finally, angle d can also be proven to be 90 − θ. The result is as follows:

angle a = 90 − θ
angle b = θ
angle c = θ
angle d = 90 − θ
angle e = 90 − θ

Now, it is relatively simple to determine that, at the apex of the triangle, the turtle must turn right at an angle of 2 × θ, move a distance of 10/cos θ and turn right 90 − θ. Turtle will then be pointing straight down, ready to draw the next side.

The Solution

Programming the solution in Logo entailed a minimum of "dirty work." Figure 4 lists the completed program.

To begin with, an arbitrary square size of 20 by 20 units was chosen, as well as an arbitrary grid size of 8 squares by 10 squares. Three short routines were then written to draw a grid, starting at the lower left corner. FILLTILES fills the screen with 10 strips of squares (Figure 4), and STRIPTILES is the procedure that draws each strip. SQUARE is a procedure that draws out each individual distorted square. The program, when executed, draws an undistorted square in the lower left corner of the screen and gradually deforms the shape until the upper right square is drawn. This corner would be the most deformed in both dimensions. The result of running the program with the command FILLTILES 8 6 can be seen in Figure 5.

In programming the solution, I introduced a number of variables and counters to keep track of the angles and increments required for the gradual deformation. XDEG and YDEG are representations of

Figure 4. The Procedures FILLTILES, STRIPTILES and SQUARE

```
TO FILLTILES :XINC :YINC
CS
PU
SETPOS [-100 -60]
PD
MAKE "XDEG 0
REPEAT 10 [STRIPTILES :XDEG :YINC RT 90 FD 20 LT 90 MAKE "XDEG :XDEG + :XINC]
END

TO STRIPTILES :XDEG :YINC
MAKE "YDEG 0
REPEAT 8 [SQUARE :XDEG FD 20 MAKE "YDEG :YDEG + :YINC]
BK 160
END

TO SQUARE :XDEG :YDEG
FD 20
RT 90 - :YDEG
FD 10 / ( COS :YDEG )
RT 2 * :YDEG
FD 10 / ( COS :YDEG )
RT 90 - :YDEG - :XDEG
FD 10 / ( COS :XDEG )
RT 2 * :XDEG
FD 10 / ( COS :XDEG )
RT 90 - :XDEG
FD 20
RT 90
END
```

θ (Figure 2), one for each of the X and Y axes. XINC and YINC are variables that hold the increment of θ for the gradual deformation along each axis.

Implications

The most striking observation was that Logo programming language played a very minor role in the project. I paid much more attention to solving the actual mathematical problem, a result that pleased me because I had intended to concentrate on mathematics rather than on computer programming.

I was somewhat disturbed that the relationship between this type of gradual deformation and trigonometry was not immediately apparent. I did not realize the connection until I had examined the problem more closely. (Perhaps this indicates a general lack of mathematical awareness in society.) Once discovered, the connection is obvious, and I wonder how long it would take someone with a weaker background to recognize the relationship.

Before, during and after the investigation, I considered implementing this and similar problems in a class. The problem would make an interesting and challenging enrichment exercise, implemented in the framework of the mathematics curriculum. Such an activity would do equally well as a culminating activity, to wrap up a unit on trigonometry and similar triangles. Whether this type of activity can be used for direct instruction needs to be examined.

rather than midpoints, are chosen? Also, what happens when the direction of the migration is altered? When the point migrates at X degrees to the right (or left) of perpendicular? And what would happen if both ideas were combined?

The ultimate investigation of this problem would involve three dimensions and repeating all of the above in the context of X, Y and Z axes. A cube would replace the square, and new variables would be introduced. Concepts such as stellate polyhedra could be examined, as well as many other topics. I can't imagine where this might lead.

There exists a vast unexplored area of mathematics. Rich rewards await those prepared to press forward and push their personal limits to the edge of what we now understand. See you there!

References

Burnett, J.D. "Tinkertoys, Tangrams, and Tessellations." *delta-K* 25, no. 3 (1986): 3-9.

Escher, M.C. *The Graphic Work of M.C. Escher.* New York: Ballantine Books, 1967.

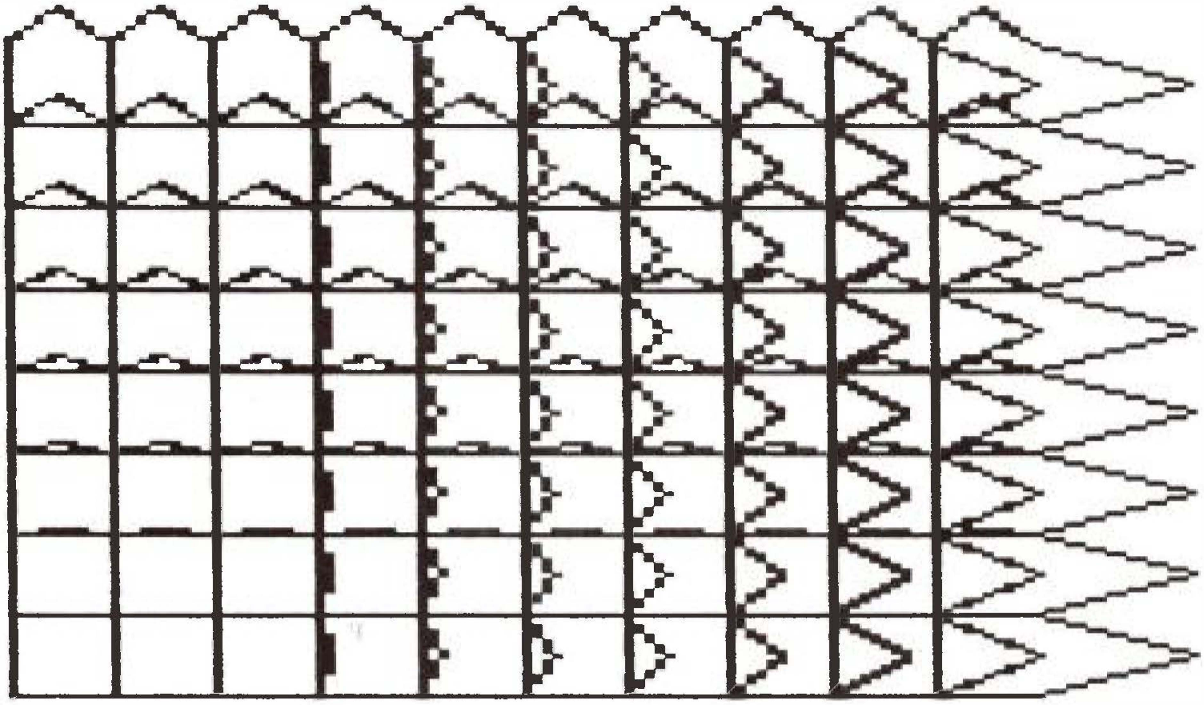
Hofstadter, Douglas R. "Metamagical Themas." *Scientific American* 249 (July 1983): 14 - 20.

———. *Godel, Escher, Bach: An Eternal Golden Braid.* New York: Random House, 1980.

Lochner, J.L. *The World of M.C. Escher.* New York: Harry N. Abrams, Inc., 1971.

Ranucci, Ernest R., and J.L Teeters. *Creating Escher-Type Drawings.* Palo Alto, California: Creative Publications, Inc., 1977.

Figure 5. The Result of FILLTILES 8 6



A number of questions come to mind. For example, wouldn't it be interesting to begin with a square in the middle of the screen and to deform all four sides toward the perimeter of the screen? Or to examine the limits of deformation? How much can the square be distorted? What happens when negative numbers are used? Are there values that will not work? Why don't they work?

What happens when one performs similar deformations of figures such as triangles, pentagons and hexagons? Is there a difference in the behavior of tessellating and non-tessellating polygons? What special problems are encountered? Are there cases in which the deformation of one shape in a particular way gives rise to a new shape? Under what conditions does this occur? These and similar questions can be posed and conclusions drawn from the answers.

Other, advanced topics could include the deformation of irregular and curved shapes. The programming would involve a fair amount of dirty work, but the mathematics would indeed be interesting.

This project involved deformations in which the midpoint of a side migrates perpendicular to, and away from, the side on which it is located. It would be interesting to examine the behavior of deformations when a point other than the midpoint is chosen. Does something special happen when corners,

Logo: An Opportunity for Synthesis, Self-Control and Sharing

J. Dale Burnett

J. Dale Burnett is an associate professor in the Faculty of Education, the University of Lethbridge. Dr. Burnett teaches courses on applying computers in education. This paper was presented at the Seventh National Congress of the Council for Exceptional Children held in Regina, Saskatchewan, in October 1986.

Introduction

Logo is a computer language specifically designed for children. I am not implying that it is a ''kiddie language'' (Logo is suitable for graduate work in computing science) but rather that the syntax and the domains of inquiry are readily accessible to young children. Logo is successful with very young children (Lawlor 1985), with physically disabled students (Goldenberg 1979), with students with learning disabilities (Weir and Watt 1981), as well as with students in regular classrooms and in gifted programs (Carmichael et al. 1985).

What are some of the reasons for Logo's success? *Synthesis, self-control* and *sharing* (the three *Ss*), plus the teacher, are key factors in Logo's success.

Synthesis refers to the natural necessity to build on one's previous knowledge (the Piagetian concept of constructivism), using both real world knowledge as well as a growing understanding of the rules of the Logo language. Self-control flags the value of permitting the learner to have a substantial degree of autonomy in what tasks are set and in the method of approaching them. Sharing refers to the social context in which much Logo activity occurs. Students helping students and feeling good about it (and about themselves) are common features of many Logo settings (Carmichael et al. 1985).

Synthesis

I will discuss the nature of synthesis at two levels. After reviewing how the concept of synthesis fits into current psychological theory, I will show how synthesis can be applied to the situation of an individual facing his or her first exposure to Logo. The second subsection then shows how this theory might apply to the situation of an individual learner, faced with their first exposure to Logo.

Synthesis and Psychological Theory

The educational community owes an enormous debt of gratitude to a self-proclaimed non-educator: the Swiss psychologist-epistemologist Jean Piaget. When one hears Piaget's name, one immediately thinks of children and of stages. The first association is a good one, the latter misleading. His substantial contribution to current psychological perspectives was not the idea of stages (which suggests that development is discrete rather than continuous) but rather that of development. Development implies change and growth. The purpose of education is to facilitate development. Change and growth are our mandate.

Equilibration is the term used to describe the underlying process of mental development by which individuals organize their ideas into noncontradictory wholes. This process occurs through the complementary subprocesses of assimilation and accommodation. Because individuals draw heavily upon what they already know (their present cognitive structure), the label constructivism can be used to describe theoretical perspective.

Traditionally, developmental literature has treated only a particular subset of the total picture: cognitive development, physical development, social development, moral development and emotional development are all familiar terms. The thesis is that such exclusions or restrictions, while well intentioned ("let us control for all possible sources of variation except one, and then observe the effect of this one remaining factor"), are fundamentally misguided. The resulting information is misleading because individuals never find themselves in such controlled situations outside of the research environment. Classroom practices or curriculum guidelines that fail to take this natural complexity into account are inappropriate.

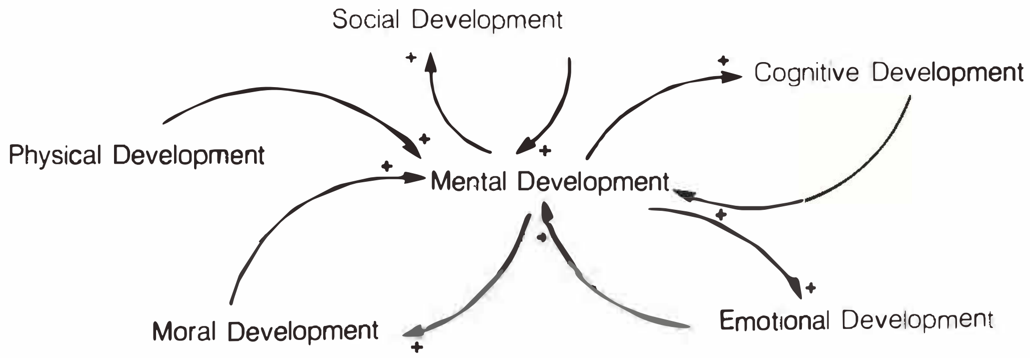
Figure 1 illustrates the interaction between mental development and other developmental factors. The arrows indicate a posited causal effect. Thus an increase in emotional development causes an increase in mental development. Similarly, an increase in mental development causes an increase in emotional development. The positive signs beside each arrowhead indicate that the relationship between the two nodes is in the same direction (for example, an increase in one causes an increase in the other, or a decrease in one causes a decrease in the other). Second, a distinction is made between mental development and cognitive development. The intention is to distinguish between overall mental development, which might include feelings and intuitions, and the more restrictive conceptual domain of cognitive development. The charting conventions follow those outlined by Roberts et al. 1983.

The psychological literature of the last decade has increasingly focused on cognitive approaches and the literature on learning emphasized individual's building upon their previous knowledge and experience. The cognitive emphasis has also expanded to encompass not only strict rational and logical perspectives but also emotional, affective and social components. Psychology is becoming both more holistic and more philosophic (for example, is knowledge constructed or discovered?) as professionals (for example, Solomon 1986) begin to reflect on the conceptual underpinnings of many of their ideas.

In addition to receiving contributions from philosophy, cognitive science (as the new discipline is called), has been strengthened by ideas from computing science. At first glance, this development seems remarkable, since one field is concerned with human ideas and nature while the other is ostensibly interested in machines and electricity. The term "artificial intelligence" is familiar to most people (Winston 1977; Haugeland 1985), and the term "expert system" is beginning to appear in the educational literature (Hayes-Roth et al. 1983; Van Horn 1986). However, lest the novice become enamored too quickly with these new ideas, cautionary notes have also appeared (Weizenbaum 1976; Dreyfus and Dreyfus 1986).

Thus cognitive science is practising what it preaches: the discipline itself is synthetic, building on any relevant bit of knowledge. One branch of computing, system dynamics, has taken the idea of modeling and simulation, combined it with the biological concepts of feedback and used the idea to construct conceptual as well as computer-based

Figure 1. Causal Loop



models of phenomena. One of the first applications of this approach to reach the public's attention was the Club of Rome's famous publication *Limits to Growth* (Meadows et al. 1972), which attempted to construct a model of the world showing interactions among population, agriculture, industry, pollution and natural resources. The same approach clarified the complexities of mental development. Diagrams and models should not be viewed as right or wrong but as appropriate or inappropriate for a given purpose or function. As the function changes, so may the model. Thus the previous model is explicated to reveal various kinds and interactions of development.

Synthesis in the Individual Learner

Figures 2, 3 and 4 illustrate a particular perspective on learning.

Figure 2

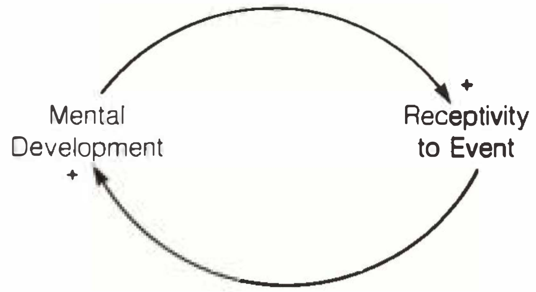


Figure 2 shows how a person's knowledge is positively related to the individual's receptivity to new events, which in turn is positively related to their knowledge. The net effect is a "constructive circle" in which learning begets more learning. "Receptivity to an event" is closely related to Vygotsky's (1962, 1978) concept of "zone of proximal development."

Both nodes clearly require amplification, and new nodes and causal arrows need to be identified. One might assume that knowledge consists of knowledge about using computers and about the specific language of Logo, as well as "other Knowledge" that may have a bearing on the present situation. The Other Knowledge may be very important. Existing knowledge about Logo may be zero: the individual may never have heard or seen it before. Existing knowledge about computers is not likely to be zero (most people have at least heard of them and have seen pictures of them) but it may be very limited (the person may not have actually touched one or watched someone else use one). (See Figure 3.)

We now have three knowledge nodes or "containers," plus one receptivity node. Thus the Logo node contains the amount of Logo knowledge that the user brings to the task (assumed to be zero). We now insert a brief exogenous variable: the instructional event. This event consists of a brief demonstration of the Logo commands "FORWARD" and "RIGHT."

An alternative representation may place more focus on the centrality of the Logo experiences. Consider Figure 4.

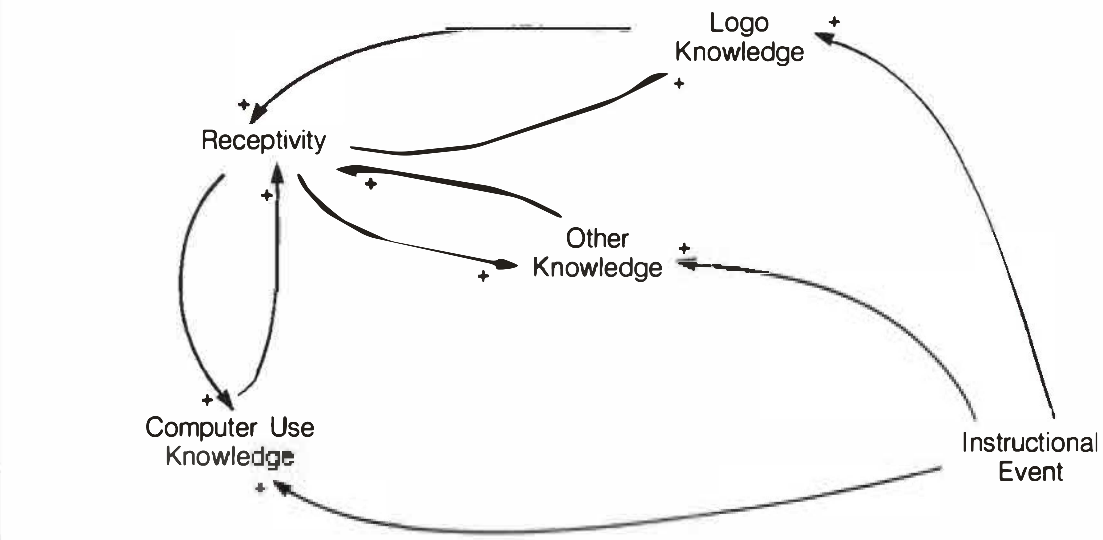
The effectiveness of this instruction depends upon (1) what the student already knows, (2) the student's attitude toward Logo and (3) the student's generic ability to learn. All three factors are "within the student." External factors include (1) the teacher's attitude toward Logo, (2) the teacher's attitude toward the student, (3) the teacher's understanding of Logo and (4) the actual instructional sequence. The addition of these nodes further complicates the situation, but the nodes may be important. Constructing a pleasing diagram is less important than constructing an adequate explanation.

Clearly, the effectiveness of instruction should not be viewed as a simple topic. We immediately realize that instruction is enhanced when

1. students can relate the instruction to what they already know,
2. students are positively disposed toward the topic,
3. students are positively disposed toward learning,
4. teachers are positively disposed toward the topic,
5. teachers are positively disposed toward the student,
6. teachers have a firm understanding of the topic, and
7. the instructional sequence takes the above into account.

The preceding summary is important not because it is particularly novel or complex but because it permits us to grasp the nature of the complexity "at a glance." Many people will consider the model incomplete, but additional information can be added. Another difficulty with the above approach is the ease with which we can construct alternative representations with little basis for choosing among them. Then

Figure 3



again, it may be that the alternative representations are equally appropriate.

Let us consider some of the details that might indicate that the first experience with Logo will be successful. First, the chances are quite good that novices (preschoolers or adults) will relate the commands FORWARD and RIGHT to their existing real-world experience of moving about. Indeed, this approach is no accident and was specifically built into the design of the original Logo. Drawing is also an early experience for virtually all children. The particular terminology and certainly the syntax may be new, but the general context should strike a responsive chord in most learners. Thus the first condition is likely to be met, at least to some degree. However, the student's initial attitude toward Logo is more difficult to estimate and is likely related to whatever attitudes the student may possess regarding computers. Attitudes may be positive, negative or neutral. A strongly negative attitude may well affect the outcome. The student's attitude toward learning is also important. A positive attitude (''learning something new is fun'') is a substantial asset; on the other hand, a negative attitude (''school is boring'') is a handicap. With young children, all three factors are often positive, perhaps explaining the level of success of introducing young children to Logo.

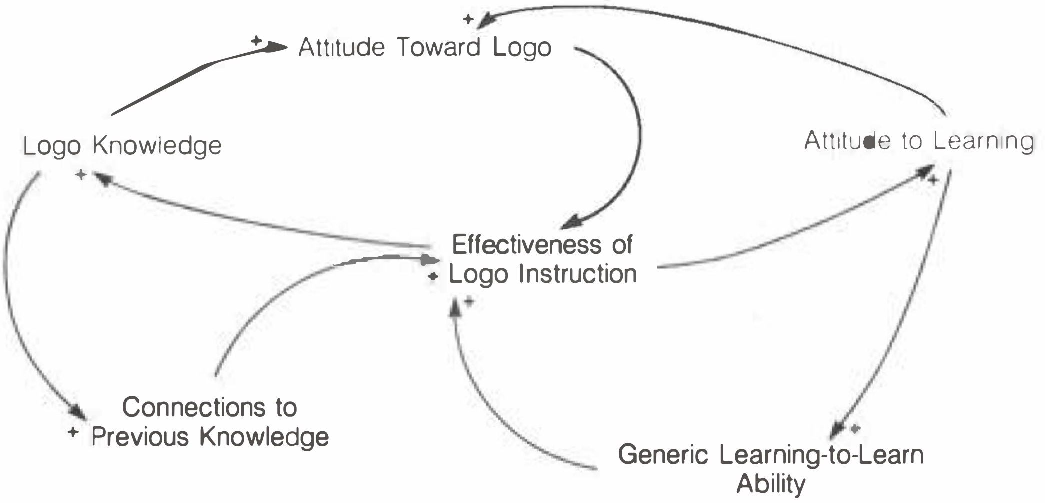
Three teacher factors were also identified. The teacher's attitude toward Logo is important. A skeptical or negative rating may be a powerful determiner of the outcome. We should not assume that the computer or Logo is a positive enough factor to compensate for a teacher who does not believe in using it. Similarly, teacher attitudes toward particular students should not be ignored. Sometimes teachers may feel that they are teaching the whole class, but this is not the case. Students filter the information as though it were directed at them. If previous events indicate that the teacher does not respect or value a particular student or students, then the teacher's impact is diminished, if not eliminated. A lesson that ''looks good on videotape'' may be entirely negated by an event that occurred two weeks earlier in the classroom.

The teacher's understanding of Logo is also important. Teaching a subject that one does not understand is indeed difficult; this applies to Logo. Finally, the particular instructional sequence is important and will be discussed later.

Self-Control

I have already acknowledged the contributions of Jean Piaget: I will now do the same for Carl Rogers. Although Rogers has written many books, I will quote from one, *Freedom to Learn for the Eighties.* The introduction contains this statement:

Figure 4


Attitude Toward Logo
Logo Knowledge
Attitude to Learning
Effectiveness of Logo Instruction
Connections to Previous Knowledge
Generic Learning-to-Learn Ability
+

> It [the book] appears in a peculiar time in our history when many are saying that we must teach only the "basics," that we must tell children what is right and wrong, that we must teach them to obey and follow. . . . They hold that students are in school to be taught, not to discuss problems or make choices. (Rogers 1983, 1–2)

Rogers (1983, 18) states that "the primary task of the teacher is to permit the student to learn." He then distinguishes between meaningful learning and learning that has no personal meaning and only occurs "from the neck up." Rogers says that meaningful learning has five characteristics: it has a quality of personal involvement, is self-initiated, is pervasive, is evaluated by the learner, and has meaning as its essence.

Other authors have noted this distinction between meaningful and meaningless learning. I wish to highlight for a moment the second characteristic—that of self-initiation. Noss (1984) focused on the related issue of ownership. Either by design or by default, many students engaged in Logo activities have had opportunities to ask their own questions, to set their own tasks and to explore their own ideas. Such events are rare—a sad reflection of our present educational system. As a result, we have little information on what occurs in such situations. However, the findings of a number of studies in which this was allowed to occur (Watt 1979; Noss 1984; Lawlor 1985; Carmichael et al. 1985) all point in a positive direction.

Sharing

Maslow (1970) acknowledged the importance of sharing in his hierarchy of motives by placing it just after basic physiological and safety needs. Yet, my experience shows that most educators view Maslow's hierarchy as something to be memorized for a psychology exam rather than as something to consider in designing the curriculum. The Logo community may, in part, be responsible for resurrecting the idea of sharing. As a result, many of the exciting events surrounding Logo experiences have a highly social flavor to them. Researchers gathering data on this dimension are impressed by its richness (Carmichael 1985). Others have failed to look for it, have not noticed it or have set up an environment to prevent it from happening (since it might contaminate the results).

The issue has broader implications. What is the role of sharing in the school environment? What are the relationships between sharing and individualized instruction and cheating? How much of the school curriculum explicitly gives students an opportunity to share? If this number is low, then why would we expect our graduates to be adept at sharing or working together? The potential for sharing occurs at many levels as well: there can be sharing among classmates working on the same task. There can be sharing among students of different classes or grade levels. For example, Grade 6 students could work with Grade 3 students, or gifted students could work with students with learning disabilities. It is naive to assume that the primary domain of learning is at the level of the subject matter. Finally, there is the sharing between student and teacher. One of my favorite anecdotes from Papert's *Mindstorms* is that of a student who, working with his teacher on a problem, suddenly says, "You mean you really don't know!"

Teacher

What is the primary function of education and what is the role of the teacher in facilitating this function? The first question is dangerous because it appears to imply a single answer. Perhaps a more appropriate question is "What are some of the principal functions of education?" This question at least leaves the door open for new ideas that may have been missed in an earlier formulation. One such idea is "learning how to learn." Novak and Gowin (1984) asked "How can we help individuals to reflect upon their experiences and to construct new, more powerful meanings?" (p. xi). They go on to say:

> Whereas training programs can lead to desired behaviors such as answering math problems or spelling correctly, educational programs should provide learners with the basis for understanding why and how new knowledge is related to what they already know. . . . (p. xi)

Perhaps we have passed through an era when American behavioristic perspectives have held sway (What can students do? What are your behavioral objectives? What are the scores on standardized tests?) and are entering an era, also with a strong American flavor, when we ask "What do students understand?" The difference in perspectives is fundamental. Unfortunately, much of our current practice is based on a perspective that may be outmoded philosophically, psychologically and educationally.

The shift toward understanding is not as simple as learning a new instructional technique. What is learning? How do students learn? What is the proper relationship of teacher to student? Alternative teacher roles vie for attention: distributor of facts, organizer of drill programs and seatwork, facilitator, encourager, and fellow explorer.

The teacher has an important role to play. For example, it is an error to assume that the synthesis within the mind of a student first exposed to Logo occurs naturally and spontaneously. The teacher can facilitate learning by bringing some of these potential connections into explicit awareness. Thus FD 100 may be related to FD 50 or to BK 100 or to FD-100. Relationships to movement commands in English or to another language may be observed. How would you tell someone from Japan to go forward? How would you tell Logo turtle to go forward? Exploring relationships among numbers may provide an excellent introduction to mathematics and the relative magnitudes of different numbers. Turtle steps and metric units may be viewed as analogous. How would you tell a robot to go forward? Now the class can discuss robotics for a while.

Connections abound. The secret is to look for them. Another example of looking for connections occurs at the meta level of problem solving when the teacher suggests that a student "play turtle" in order to figure out how to draw a particular figure with a sequence of Logo commands. Other suggestions such as breaking a complex problem into a number of simpler subproblems or developing an overall structure to the solution can be related to other non-Logo activities such as writing a term paper, baking a cake or studying for a history exam.

The teacher should be aware of numerous potential connections: the relating of Logo commands to one another, the relating of Logo to other non-Logo environments, the relating of Logo problem solving to generic problem solving. Everyone should be alert for connections between the specific situation and other knowledge. As a result, the student should see that learning Logo is much like learning anything else. The synthesis should include not only low-level activities such as learning how to use Logo language, but higher meta-level activities such as debugging, planning, organizing, problem solving, attitude awareness, communicating and sharing approaches and strategies. The basics of education may be at the other end of the continuum from where we have been looking. It may be very difficult to show some of these connections empirically. That does not necessarily mean they do not exist but that our current research procedures are at fault.

Education also benefits from a little faith. On the other hand, researchers must continue their efforts to provide further insights into our understanding of the learning process.

References

Carmichael, H.W., J.D. Burnett, W.C. Higginson, B.G. Moore and P.J. Pollard. "Computers, Children and Classrooms: A Multisite Evaluation of the Creative Use of Microcomputers by Elementary School Children." Toronto: Ontario Ministry of Education, 1985.

Dreyfus, H.L., and S.E. Dreyfus. *Mind Over Machine: The Power of Human Intuition and Expertise in the Era of the Computer.* New York: Free Press, 1985.

Goldenberg, E.P. *Special Technology for Special Children.* Baltimore: University Park Press, 1979.

Haugeland, J. *Artificial Intelligence: The Very Idea.* Cambridge, Massachusetts: MIT. Press, 1985.

Hayes-Roth, F., D.A. Waterman and D.B. Lenat, eds. *Building Expert Systems.* Reading, Massachusetts: Addison-Wesley, 1983.

Lawlor, R.W. *Computer Experience and Cognitive Development.* Chichester: Ellis Horwood, 1985.

Maslow, A.H. *Motivation and Personality.* 2nd ed. New York: Harper and Row, 1970.

Meadows, D.H., D.L. Meadows, J. Randers and W.W. Behrens. *Limits to Growth.* London: Pan Books, 1974.

Noss, R. "Children Learning Logo Programming." Chiltern Logo Project. Interim report No. 2, 1984.

Novak, J.D., and D.B. Gowin. *Learning How to Learn.* New York: Cambridge University Press, 1984.

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

Roberts, N., D. Anderson, R. Deal, M. Garet and W. Shaffer. *Introduction to Computer Simulation.* Reading, Massachusetts: Addison-Wesley, 1983.

Rogers, C. *Freedom to Learn for the Eighties.* Columbus, Ohio: Charles E. Merrill, 1983.

Solomon, C. *Computer Environments for Children.* Cambridge, Massachusetts: MIT Press, 1986.

VanHorn, M. *Understanding Expert Systems.* Toronto: Bantam Books, 1986.

Vygotsky, L.S. *Thought and Language.* Cambridge, Massachusetts: MIT Press, 1962. (Originally published in Russian in 1934.)

——. *Mind in Society.* Cambridge, Massachusetts: Harvard University Press, 1978.

Watt, D. "Final Report of the Brookline Logo Project Part III: Profiles of Individual Student's Work." MIT Logo Memo No. 54, 1979.

Weir, S., and D. Watt. "Logo: A Computer Environment for Learning-Disabled Students." *The Computing Teacher,* 1981, pp. 11-19.

Weisenbaum, J. *Computer Power and Human Reason.* San Francisco: W.H. Freeman, 1976.

Winston, P.H. *Artificial Intelligence.* Reading, Massachusetts: Addison-Wesley, 1977.

Integrating Mathematics and Logo: A Four-Step Approach

W. George Cathcart

Dr. Cathcart is a professor of education at the University of Alberta. He served on the executive of MCATA and is a former editor of delta-K. *Dr. Cathcart has presented papers at the annual meeting of MCATA and at NCTM general meetings.*

Programming in Logo develops problem solving skills and affects cognitive processes in young children (Clements and Gullo 1984). Having middle school children write Logo procedures that simulate certain mathematical algorithms and processes may also be a worthwhile endeavor.

First, the process sharpens a child's understanding of the mathematical concept or algorithm. Before a procedure can be written, the mathematics are analyzed or broken down into small "mind-sized bites."

Second, programming broadens the child's understanding of Logo and leads to a greater appreciation of the power of Logo. Children discover that Logo can do many things besides drawing designs, geometric figures and graphs.

A Developmental Sequence

Using Logo to enhance a mathematical concept involves four steps. The first two steps focus on the mathematics and reinforce and clarify the mathematics for students; Steps 3 and 4 focus on Logo. To complete these steps, students may need to consult a Logo manual or ask for guidance. The experience will both develop their skill in programming and broaden their understanding of Logo's capabilities.

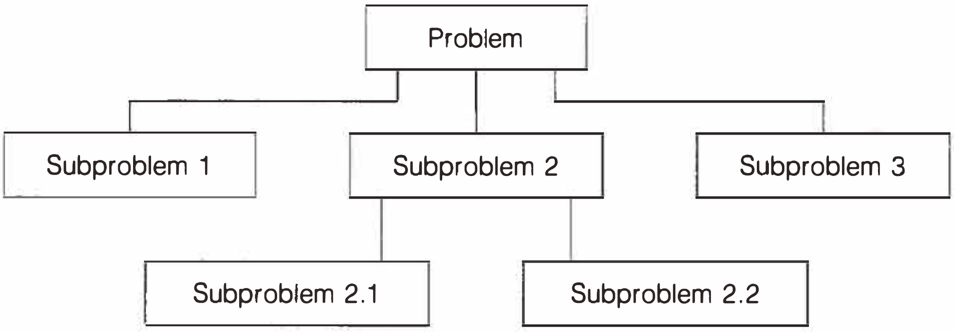
Focus on Mathematics

Step 1. Analyzing the Mathematics

Before students write a Logo procedure to perform a mathematical task, they must be able to break the mathematical process down into its components. Computer scientists call this process "stepwise refinement." Figure 1 illustrates the process.

For example, to find the mean set of numbers, students might analyze the problem by

Figure 1. Stepwise Refinement



1. calculating the sum of all numbers in the list,
2. dividing the sum by the number of elements in the list,
2.1. counting the number of elements in the list, and
3. printing the results.

Figure 2 represents this process using the stepwise refinement chart.

Step 2. Write a Pseudo Code

After analyzing all the components, students should be encouraged to write a pseudo code corresponding to the mathematical steps. Based on the example of the mean, a pseudo code would look something like the following:

```
TO MEAN
  • get sum
  • count number of elements
  • divide sum by the count
  • output the result
END
```

This is just one example of a possible format for a pseudo code. You may prefer a different style. What is important is that students structure their analysis of the mathematical process into a logical step-by-step algorithmic-like statement. The focus here is still on the mathematics. The step is an attempt to state in a succinct form the mathematical process. Additionally, the step serves as a transition to focus on Logo.

Focus on Logo

All the Logo procedures in this article are written in Apple Logo, a product of Logo Computer Systems (LCSI) Inc. It is assumed that the students have a reasonably good understanding of Logo including tail recursion.

Step 3. Logo Match to Pseudo Code

The first attempt at writing a Logo procedure to perform the mathematical task could be a relatively straightforward translation of the pseudo code into Logo. The pseudo code developed in Step 2 can be translated as follows:

```
TO MEAN :ALIST
MAKE "S ADDUP :ALIST
MAKE "C COUNT :ALIST
MAKE "R :S / :C
OUTPUT :R
END
```

Note that ADDUP is used as an operation in line 2. ADDUP, from the context, appears to be an operation that calculates the sum of a list of numbers. ADDUP is not a Logo primitive and should be given to students as a tool procedure, perhaps included in a STARTUP file. There is one version of ADDUP:

```
TO ADDUP :ALIST
IF EMPTYP :ALIST [OUTPUT 0]
OUTPUT SUM FIRST :ALIST ADDUP BUT-
  FIRST :ALIST
END
```

Figure 2. Stepwise Refinement for Finding Means

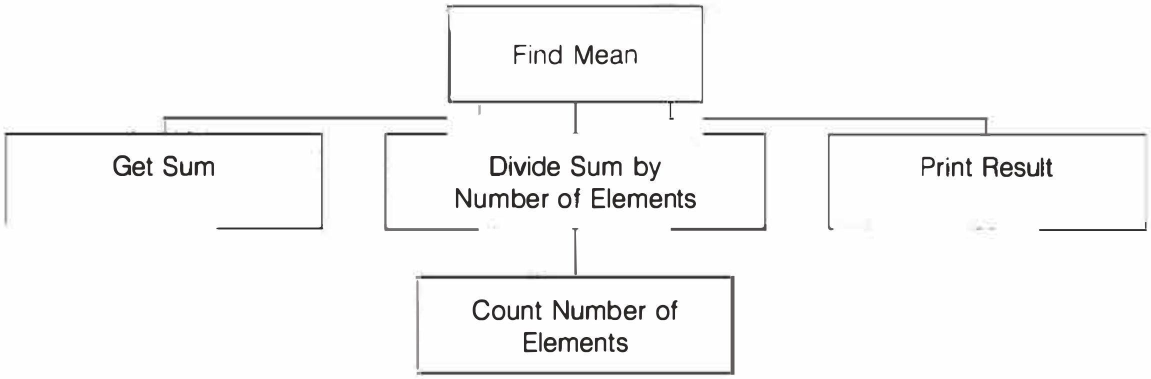


Figure 3. Pseudo Code and Logo Match in Parallel

Step 2. Pseudo

```
TO MEAN
  • get sum
  • count number of elements
  • divide the sum by the count
  • output the result
END
```

Step 3. Logo Match

```
TO MEAN :ALIST
MAKE "S ADDUP :ALIST
MAKE "C COUNT :ALIST
MAKE "R :S / :C
OUTPUT :R
END
```

A worthwhile exercise for students is to do Steps 2 and 3 side-by-side on a page (see Figure 3).

The pseudo code and the Logo match may not always correspond as closely as in Figure 3. Minor adjustments to format may be required for some tasks. However, if the pseudo code is a detailed statement of the mathematical process, Logo statements can usually be written to match it step-by-step.

Logo "purists" will be dismayed by the extensive use of global variables in this approach. The situation was a trade-off, I wanted a strategy that would make the transition from mathematics to Logo a small step. Steps 2 and 3 seem to do this. I am prepared to trade some purism for simplicity. Step 4 restores some of the "pure" Logo.

Step 4. More Elegant Logo

In Step 4, encourage students to ask "Can I write this procedure in a better way?" The purpose is to write a more elegant procedure. Students should consider shortening statements, combining statements, using different commands or operations, taking a completely different approach and so on.

A more elegant procedure for our example of the mean might combine all of the steps into one line:

```
TO MEAN "ALIST
OUTPUT (ADDUP :ALIST) / COUNT :ALIST
END
```

In writing a more elegant Logo procedure, students should be encouraged to eliminate as many MAKE statements as possible. (Note the removal of all MAKE statements in the final MEAN procedure.) After students work through this four-step process with different mathematical concepts, you may wish to explain the difference between local and global variables in Logo and the advantages and disadvantages of each.

With some experience, students may translate the pseudo code (Step 2) into a Logo procedure that is somewhat "better" than a straight line-by-line translation. However, Step 4 still needs to be emphasized because a search for a more elegant procedure should never end. (To illustrate this point, refer to the example of the median in the following section.)

Students soon discover that the four-step process does not always result in a Logo procedure that will do exactly what was expected. A bug may have crept in during the transition from pseudo code to Logo. The pseudo code itself may contain errors, such as missing steps. Possibly the analysis of the mathematics was incorrect. Therefore, a continual evaluation or monitoring of each step and the total process must be practiced. Figure 4 illustrates the process of monitoring.

More Examples

To illustrate the outlined four-step process of integrating Logo into mathematics, two additional nongraphic examples are outlined. One deals with divisors (factors), the other with the median.

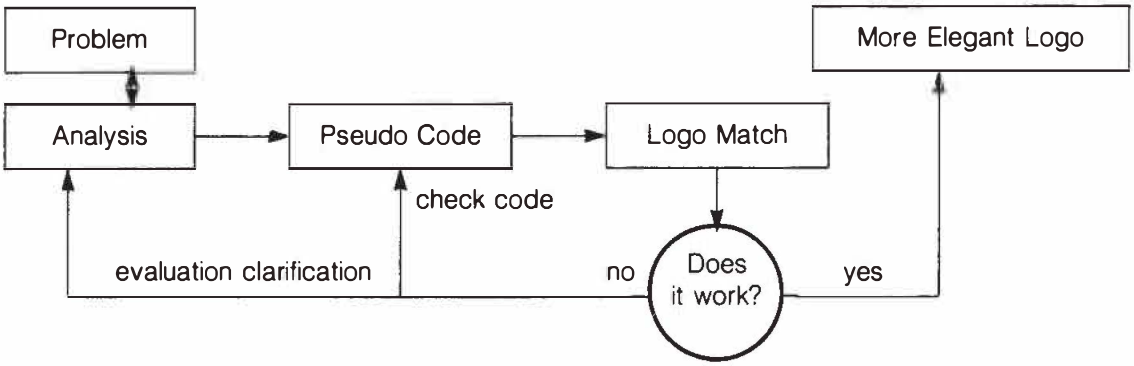
Factors

The problem is to find all the factors of a whole number, n.

Step 1. Analysis

A factor of n is a whole number that divides evenly into n. One method of obtaining all the divisors of a number is to check for divisibility (remainder = 0) by all whole numbers less than or equal to n. If divisibility occurs, list the divisor as a factor; otherwise try the next whole number. Except for n itself,

Figure 4. Dynamics of the Four-Step Process



check divisors of whole numbers up to n/2 because no number between n/2 and n will divide evenly into n. A child thinking through the concept of a factor in this way clarifies the concept in his or her mind.

Step 2. Pseudo Code

There are several ways the analysis could be translated into succinct logical steps. Here is one possibility:

```
TO FACTORS of n
   • start with divisor of 1
   • if divisor > n/2, print n as a factor and stop
   • otherwise get remainder when n/divisor
   • if = 0, list divisor as a factor
   • otherwise, repeat with divisor 1 greater
END
```

Step 3. Logo Match

```
TO FACTORS :NUM :DIV
IF :DIV > :NUM / 2 [PRINT :NUM STOP]
MAKE "R REMAINDER :NUM :DIV
IF :R = 0 [PRINT :DIV]
FACTORS :NUM :DIV + 1
END
```

The pseudo code and the Logo match do not correspond as closely as in the example of the mean described earlier. The initial divisor, 1 (line 2 of the pseudo code), is incorporated as the second input to the procedure. A sample execution of FACTORS might be: FACTORS 36 1. Different versions of pseudo code may result in a greater or lesser degree of correspondence in the Logo match.

Step 4. More Elegant Logo

The first improvement would be to combine lines 3 and 4 and remove the MAKE command:

```
TO FACTORS :NUM :DIV (assign :DIV the
   value 1)
IF :DIV > :NUM / 2[PRINT :NUM STOP]
IF REMAINDER :NUM :DIV = 0 [PRINT :DIV]
FACTORS :NUM :DIV + 1
END
```

This version places the factors in a vertical format. A more elegant procedure should print them horizontally with a space or a comma between each factor. Changing the THEN portion of line 3 to [(TYPE :DIV CHAR 32)] would cause this to happen.

As it stands, FACTORS is rather limited in its usefulness. To be used in another set of procedures, say for finding common factors, FACTORS needs to be an operation or outputting procedure. Unfortunately, PRINT or TYPE cannot simply be replaced by OUTPUT since OUTPUT also stops the procedure. This is a case in which students will likely need your assistance.

One way around the dilemma is to store each factor in a list, and output the list when all the factors have been determined. The following procedure, while longer, is more elegant; results can be used as input to other procedures in which factors are needed.

```
TO FACTORS :NUM
MAKE "FACTS [ ]
GET.FACTORS :NUM 1
```

```
OUTPUT SENTENCE :FACTS :NUM
END
TO GET.FACTORS :NUM :DIV
IF REMAINDER :NUM :DIV = 0 [MAKE
  "FACTS LPUT :DIV :FACTS]
IF :DIV < :NUM / 2[GET.FACTORS :NUM
  :DIV + 1]
END
```

Sample Output

```
?PRINT FACTORS 16
1 2 4 8 16
```

LPUT is the operation storing each factor as it is generated into the list, FACTS. Notice that, since a list was created, the output is automatically in horizontal form.

Median

I dealt with the concept of the median in a previous issue of *delta-K* (Cathcart 1986). Some of the ideas presented in that article can be incorporated into the four-step process.

Step 1. Analysis

For the median to be found, the data needs to be sorted. The number of elements in the data list must be determined. If this number is odd, the middle number is the median. Otherwise, the median is the average of the middle two numbers.

Step 2. Pseudo Code

```
TO MEDIAN
  • sort input list
  • count number of elements in input list
  • check if number of elements is odd or even
    • if odd, pick middle element
    • otherwise find mean of middle two numbers
  • output result
END
```

Step 3. Logo Match

```
TO MEDIAN :ALIST
MAKE "SL SORT :ALIST
MAKE "C COUNT :ALIST
TEST (REMAINDER :C 2) = 0
IFTRUE [EVEN]
IFFALSE [ODD]
OUTPUT :R
END
```

This procedure calls two subprocedures, EVEN and ODD. SORT is a tool procedure that sorts data into ascending order. This procedure should be given to students as a tool. (See Cathcart 1986, for a listing of a sort procedure.)

```
TO EVEN
MAKE "F ITEM :C / 2 :SL
MAKE "K ITEM :C / 2 + 1 :SL
MAKE "R MEAN LIST :F :K
END
TO ODD
MAKE "R ITEM :C / 2 + .5 :SL
END
```

Step 4. More Elegant Logo

A first attempt to write a more elegant procedure may result in the following:

```
TO MEDIAN :ALIST
MAKE "C COUNT :ALIST
IF (REMAINDER :C 2) = 0 [OP EVEN] [OP
  ODD]
END
TO EVEN
OP MEAN LIST ITEM :C / 2 :ALIST ITEM :C
  / 2 + 1 :ALIST
END
TO ODD
OP ITEM :C / 2 + .5 :ALIST
END
```

With these procedures, SORT would be used as an input to MEDIAN. That is, PRINT MEDIAN SORT :ALIST. This example shows how trying to modify a procedure, while retaining the basic strategy, may blind a programmer to a far more elegant solution. By determining the median with pencil and paper, it is possible to simply strike out the first and last elements of the sorted data. Continue this process until only one or two elements are left. If one element remains, it is the median. If two elements remain, the average of these two is the median. Actually, if only one element remains, the median is still the average of the number. To illustrate:

Case 1: ~~1~~ ~~3~~ ~~9~~ 16 ~~25~~ ~~31~~ ~~33~~
16 is the median

Case 2: ~~1~~ ~~3~~ 9↓16 ~~25~~ ~~31~~
12.5 is the median

This suggests a recursive procedure. A much more elegant procedure for calculating the median, then, would be as follows:

```
TO MEDIAN :ALIST
IF OR ((COUNT :ALIST) = 1) ((COUNT :ALIST)
   = 2) [OP MEAN :ALIST] [OP MEDIAN BF BL
   :ALIST]
END
```

Line 2 (IF–THEN portion): if there are one or two elements in the data list, output the mean and stop.

Line 2 (ELSE portion): if there are more than two elements, strip away the first and last and repeat the process.

Summary

A four-step process for integrating Logo programming into mathematics consists of

1. analyzing the mathematics,
2. writing a pseudo code for the mathematical process,
3. writing a Logo code to correspond to the pseudo code, and
4. writing a more elegant Logo procedure.

Some steps may be repeated or revised as the process develops. This action may be needed to correct the code or to re-analyze the problem due to initial misconceptions or omissions.

If students follow the four steps, they will likely sharpen their understanding of the mathematics involved, broaden their knowledge of Logo, increase their appreciation of Logo and hone their programming skills.

References

Cathcart, W.G. "Logo and Measures of Central Tendency." *delta-K* 25, no. 3 (1986): 27–31.

Clements, D.H., and D.F. Gullo. "Effects of Computer Programming on Young Children's Cognition. *Journal of Educational Psychology* 76, no. 6 (1984): 1051–58.

Of Students, Computers and Learning

Barry McGuire

Barry McGuire is a physics teacher and science department head at Western Canada High School, Calgary. McGuire holds a B.Sc. and B.Ed. from the University of Saskatchewan, and a M.Sc. from the University of Calgary. He has contributed articles to Alberta Science Education Journal *published by the ATA Science Council.*

In the mid-1970s, like many other teachers, I became excited about the educational possibilities of the then-new microcomputer technology. This interest led, in 1979, to an Innovative Learning Project designed to explore the uses of the microcomputer in a high school physics class. In 1984, I designed a curriculum for a locally approved course called Scientific Studies and Computing. The groundwork for the curriculum was taken from those aspects of the Innovative Learning Project that offered the most interesting and constructive learning experiences.

Scientific Studies and Computing is, in all senses of the words, a science course. All of the objectives defined in the curriculum emphasize the nature, the knowledge and the processes of science. However, the curriculum displays one major difference from the regular science course: students in this course become the teachers; their pupils are the computers.

To fulfill the requirements of the course, students complete two science projects. The student chooses a topic from science and develops a computer application for science within that topic. Since the student is placed in one long problem solving situation, the meta-lessons become the most important learning experiences.

After selecting a topic, the student prepares a project proposal. The proposal outlines the science content that the student expects to learn. The proposal also outlines the nature of the computer involvement in the project. Upon approval, students select the type and sequence of activities necessary to complete the project. My role is mainly that of a resource person who ensures that the students are on task while in the classroom. Neither task is particularly easy.

First, students often choose topics outside my range of expertise. In such cases, if the problem is complicated, I may do quite a bit of background reading. Second, what appears to be an unproductive approach on the surface may have several extremely beneficial long-term effects. If students give the impression that there is some design behind their activity, then I usually let them pursue it. On the other hand, students often perceive that the six to ten weeks allowed for a project is a long time and that to catch up on a little homework during class would not be a serious misuse of time. I do not permit this.

Initially, I was quite uncomfortable with the structure, or lack thereof, and to some extent I remain uneasy. Subconsciously, I want the students to be involved in activities that produce tangible results of their efforts everyday. Consciously, I realize that problem solving techniques evolve from a wide variety of behaviors. Therefore, I refrain from contributing when it is not essential. Indeed, students often request help, not because they are unable to solve a problem, but because they want reassurance that they are on the right track. In such cases, it would be easy to steer them in the direction I think their project should go. I try to give them the courage to proceed, although there is some doubt as to the outcome.

However, not all activities are unstructured. Students must have taken a 20-level science course as a prerequisite to ensure that they have some science background. During the first weeks of class, I establish the groundwork for science. Most students are rather naive when it comes to this activity. I first establish the nature of science by discussing such activities as hypothesizing, interpreting, classifying, analyzing and problem solving.

Second, I introduce students to the Apple IIe and computer programming. While experience with computers and programming is recommended, it is not compulsory. To ensure that students have the fundamentals of BASIC programming, I give short programming assignments at the time that I discuss several of the science processes. For example, when discussing data and data analysis, I give students a parallel programming problem. This program requests the user to "input" several pieces of parametric data, find the mean of this data, find the standard error and display the results in a prespecified fashion on the video display terminal.

Obviously, students will have a wide range of programming experiences. Problems, such as the one discussed earlier, will challenge some students and be extremely simple for others. Peer tutoring is encouraged and students regularly consult each other on programming techniques. Mini-lessons on programming (which most students require), are presented throughout the course. Books on programming are available in class, and students quickly become familiar with the resources and devise solutions to most of the programming tasks specific to their projects.

Selecting a project topic is very difficult (anyone who ever selected a thesis topic can relate to this task). Moreover, students inevitably perceive their project as a program that will teach the user all the neat information they have accumulated in preparing their project, a sort of computer assisted instructional program (CAI). On the contrary, their task is to create an application or utility program, a program, in other words, that makes the computer a useful tool to a scientist in the area of science from which the project originates.

The application of computers as scientific tools is a difficult concept for students to grasp. The most effective way to teach this concept is to use past projects that demonstrate several successes and failures. Even so, continued reinforcement is required to remind students that the nature of the computer application decides the ultimate validity of their projects. A computer application that attempts to do interesting things in science may have several programming flaws and yet be viewed much more favorably than a slickly programmed application that has a less scientifically valid application.

Here are some examples. The first time the course was offered, a student with considerable programming experience who was quite fluent in BASIC wanted to carry out a titration simulation. He wrote a good proposal describing a simulation of titration using graphics and a considerable amount of user interaction. Several discussions throughout the development of his project suggested that he was going not in the direction of his original proposal but more in the direction of a tutorial. When the project was presented, most of the graphics were found in a beautifully prepared title screen. The main routines of the program were merely a titration calculation sequence and a problem generation sequence. User interaction was limited to entering data to complete the calculations for the pH value of the unknown acid or base.

What of this student's learning objectives? Since he never studied titration, he certainly had to extend the base of his scientific knowledge to carry out the project. The output to the computer screen was extremely well-designed and the data handling routines were excellent, but the scientific application of his program was weak. The project is typical of programs in which students misperceive the nature of computer applications in science.

Another student in the same class, however, completed the best project to date. The student undertaking the project had virtually no computer experience but possessed a great love of astronomy. His previous scientific experience was limited to project work and astronomy projects in the science fair. All of the programming skills used in his first project were picked up in developing his project. What made this project succeed? First, the student formulated a hypothesis that he wanted to test. Thus, he had a clear image of the science involved in his project. Second, there was a definite role for the computer to search the data for relationships verifying his hypothesis. Third, he had a clear image of what the final product should do.

The essence of his hypothesis was that the periodicity of the fluctuations in size of red giant carbon stars was related to the periodicity of the fluctuations in the intensity of the light given off in certain areas of their spectrum. Data for the periods of fluctuation in the size and light intensity in each area of the star's spectrum for about 100 red giant carbon stars were entered into a data file. The program grew as the student's programming experience increased.

First, he programmed the search sequence to look through the data to find stars that gave light in the regions of the spectrum specified by the user. Then the program plotted the period of the size of the star against the period of the light. If the graphed data resulted in an approximately straight line, a relating constant was calculated. In this way, he could explore relationships not only between the periods of the star's fluctuations in size versus light intensity but also between the periods of the intensities of various areas of the spectrum. Many of the programming techniques were quite sloppy, mostly due to the inexperience of the programer, but the science of it was quite exquisite.

Another excellent program was carried out by a student who bred dogs as a hobby. The student was quite interested in the genetics of sex-linked characteristics. She proposed to develop a program that could track sex-linked characteristics throughout a breeding sequence of five generations. She had absolutely no previous computer experience and several times got stuck trying to debug quite convoluted programming sequences. In the end, she learned about programming as well as about preplanning problem solving approaches.

The program enabled the user to specify the characteristic that was sex-linked to specify the genetic structure on which the characteristic was found. Then the user could choose the genotype of the male and female. The computer generated the genotypes of the offspring and allowed the user to specify which offspring was to be bred for the next generation and to specify the genotype of the new breeding partner. The computer repeated the process until all five generations were traced and then presented a summary of the breeding sequence.

What students found most interesting during the in-class presentation was when the program traced the incidence of hemophilia in the royal families of Britain, Germany and Russia in the late 19th and early 20th centuries.

In some cases, previous programming seemed to be a drawback. Many students with programming experience viewed the class as a computer class rather than a science class. As a result of this confusion, several students dropped the class.

The best programmer in the class almost dropped the course. The student could program more proficiently in assembly language than most students could in a higher level language. What kept this student from dropping the class was a discussion about the nature of gravity. We discussed the book *Flatland,* and the possibility that gravity could be a distortion of our three-dimensional world into the fourth-dimension. The following day, the student presented a proposal for his project. He wanted to design a program that would allow the user to enter the data points (ordered triplets) for a "wire frame" diagram of an object in the third-dimension. The program would then permit the user to rotate the object to any view in that space and then translate that view into one from the second-dimension or extrapolate it to a higher-order space. For example, if a cube was entered in the third-dimension, it could be rotated about any of the three orthogonal axes. Then the cube could be viewed as it would appear to a person living in the fourth-dimension.

Two factors almost stopped me from approving his proposal. First, it lay on the fringes of science and really was a project in pure mathematics. Second, and more critically, the project was very complex. However, since the proposal was so clearly presented, I decided to allow it.

The student immediately plunged into researching the mathematics of drawing three-dimensional projections on a two-dimensional space—the display screen. (The techniques for programming this are found in BASIC.) The key is the matrix; a cube, for example, is a three-by-eight matrix containing the ordered triplets for its corners. To make the cube undergo a realistic rotation, a matrix multiplication with another matrix (when the trigonometric functions occupy the cells rather then numerical data), was required. Since this program ran so slowly in BASIC, the student translated the entire program to machine language. For weeks on end, he was immersed either in books on matrix mathematics or in books on spatial projections.

The next problem was that of dimensional translation; it proved much more difficult than he first imagined. In the case of matrix mathematics and spatial projections, all previous analysis was by mathematicians. All the student had to do was figure out what they were talking about and translate it into computer language. Because he had to create the mathematics before he could begin programming, he had limited success. Nonetheless, during the four months he spent on the project he did learn an immense amount not only about mathematics but also about problem solving.

His presentation included the rotation of a wire frame diagram of a simple car. However, the extrapolation of the car into the fourth-dimension did not succeed entirely: portions disappeared in the translation because the matrix generated by the mathematical operations to translate even a simple solid from the third- to the fourth-dimension was so large that it required more memory than the computer had available. Therefore, much of the data generated by the program was lost. As a footnote, about a year

and a half later I received a disk and the documentation for the completed version of this programming that he was preparing to market.

Students with sufficient programming background can undertake projects that involve interfacing the computer with laboratory equipment. The gameport on the Apple is fairly simple to access from any type of program. Several sensing devices are available and quite easy to use. Phototransistors, photoresistors, thermistors and transistors can be connected to the gameport, thus providing a relatively safe (for the computer) and easy (for the student) connection between the computer and the outside world.

One student's initial proposal was to develop an interface so that the computer, in conjunction with a spectroscope, could do spectral analysis. Unfortunately, the difficulty was that the light levels from the spectroscope were too low to activate the phototransistor. After several unsuccessful attempts to develop an amplifier to make the system more sensitive, the student changed the direction of his project. As a result, the new project tried to use the phototransistor as a light metre.

Although this project didn't have quite the romance of the original, it still had considerable merit. The student researched light intensity and luminance. He researched how the computer interacted with the phototransistor in order to tell his program how to read the phototransistor. Then, to translate the value the computer read from the phototransistor into an intelligible number, the student had to understand the nature and importance of instrument calibration. As a final step, the computer collected and stored the data in a format that made it intelligible to a commercially purchased graph analysis program. Using his own program, the student then collected data for luminance versus the separation of light emitted from both a point source (a bulb) and a rod source (a florescent tube) of light. Subsequent graph analysis of the data showed the inverse square law for the point source of light and the inverse first power law for the rod source.

At some point in developing any project, the original excitement of the project wears off. When that happens, persistence in problem solving becomes extremely important. Some students have confidence in their ability and do not need much encouragement; others need regular shots of enthusiasm. Once the first project is successfully completed, the difference in the students' approach to their second project is quite remarkable. Although the second project promises to be longer and more difficult than the first, students are much more confident of their ability to handle the challenge. They are more independent and flexible in their approaches to problem solving. They are less threatened by peer criticism than they were initially.

How students react to peer criticism is especially noticeable during the evaluation stage of the project. Projects are evaluated on the basis of three criteria. The science aspects of the project comprise 50 percent of the final mark. The nature of the computer involvement and the final program contribute 30 percent to the grade. Finally, an in-class presentation of the project, adjudicated by the students, earns 20 percent. Not only do the students respond more positively to peer evaluation on their second project, but, having all been through a peer evaluation, they are much more perceptive and constructive in their criticisms of others the second time around.

I would eventually like to have several students cooperate in a major project. Each student would develop a separate segment of the computer program, which would then be merged with students' sections. Considerable group planning in both the scientific and the computer aspects of project development would be realized. Perhaps the potential for disaster in this approach looms too large.

In the meantime, students benefit from their experiences in several ways: they increase their problem solving ability, develop persistence, come to understand the nature and process of science and learn to appreciate the symbiotic relationship between science and technology.

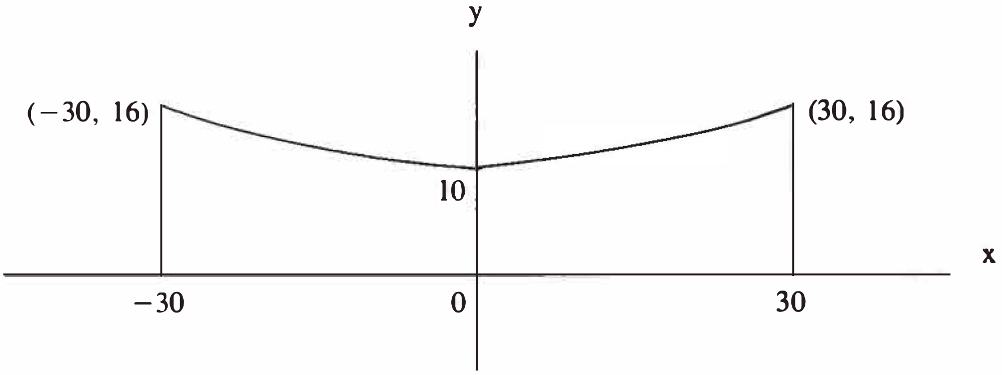
Student Problem Corner

Editor's Note: The problem presented here is suitable for students studying pre-calculus. It explores the concepts of exponentional function and quadratic formula. The problem is reprinted with permission from Resource Problems to Enhance the Teaching of Mathematics, *University of Delaware, Newark, Delaware. This collection of senior high school mathematical problems is available for $4 U.S. from Willard E. Baxter, Department of Mathematical Sciences, 501 Ewing Hall, Newark, Delware 19716. Phone: (302)451-2653.*

Problem

A wire hanging symmetrically across a road is anchored at the edges of the road on standards. The standards are 16 feet high. The road is 60 feet wide, and the height of the wire at the centre of the road is 10 feet above the road. Furthermore, after establishing the coordinate system shown, it is observed that the graph of the wire is

$$y = \frac{B^x + B^{-x}}{2} + 9$$



1. Find B as an algebraic expression.

2. What special relationship do the two values of B have?

3. Give the first five decimals in the decimal approximation of B.

4. Can a truck 10 feet wide and 14 feet high pass under the wire?

Solution

FORMULA

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}, \text{ where } ax^2 + bx + c = 0$$

STRATEGY

Since (30, 16) is a point on the curve, substitute the coordinates into the formula. Express the resulting equation in quadratic form and solve with the quadratic formula.

Question 1

ARGUMENT

$$y = \frac{B^x + B^{-x}}{2} + 9$$

$$16 = \frac{B^{30} + B^{-30}}{2} + 9$$

$$32 = B^{30} + B^{-30} + 18$$

$$B^{30} + B^{-30} = 14$$

$$B^{60} - 14B^{30} + 1 = 0$$

$$B^{30} = \frac{14 \pm \sqrt{14^2 - 4(1)(1)}}{2}$$

$$B^{30} = \frac{14 \pm \sqrt{192}}{2}$$

$$B = \sqrt[30]{\frac{14 \pm \quad 192}{2}}$$

ANSWER

$$B_1 = \sqrt[30]{\frac{14 + \quad 192}{2}} \quad \text{and } B_2 = \sqrt[30]{\frac{14 - \quad 192}{2}}$$

Question 2

Since $B^x + B^{-x} = (1/B)^x + (1/B)^{-x}$, we see that $B_2 = 1/B_1$.

JUSTIFICATION

$$B_2 = \sqrt[30]{\frac{(14 - \quad 192)\,(14 + \quad 192)}{2\,(14 + \quad 192)}}$$

$$= \sqrt[30]{\frac{2}{14 + \quad 192}}$$

$$= 1/B_1$$

ANSWER
$B_2 = 1/B_1$

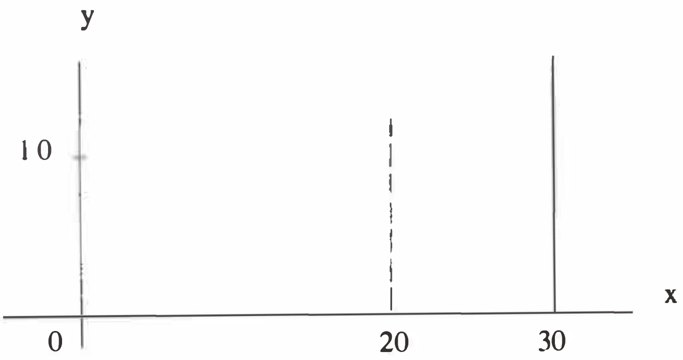
Question 3

The appended program allows one to approximate the values of B using a bracket-and-halving algorithm. We first estimate

$$n = \frac{14 + \sqrt{192}}{2} = 13.92820$$

ANSWER
Using the algorithm, we find that
$B_1 = 1.09176$ and $B_2 = 0.91601$

Question 4



ANSWER
We wish to approximate y when x = 20. Using the accompanying algorithm

$$y = \frac{B^{20} + B^{-20}}{2} + 9 = 11.98097$$

Since the truck is 14 feet high, it *cannot* pass under the wire. Therefore, the answer is no.

Program for the Algorithm

```
10     'Intent: Approximate a root of a number.
20              Evaluate y, when x is 20, in the formula from the problem.
30     'Variables:
40     '       A$ is the user's answer to input questions.
50     '       B is the maximum amount of error to be allowed in the calculation.
60     '       E is the index of the root.
70     '       I is the counter for calculating the power by repeated multiplications.
80     '       L is the lower bound for the root.
90     '       M is the midpoint of L and R.
100    '       N is the real number whose root is to be calculated.
110    '       P is the E'th power of T.
```

```
120   '     Q is the reciprocal of P.
130   '     R is the upper bound for the root.
140   '     T is the current approximation of the root that is being tested.
150   '
200   'BEGIN
210     CLS
220     GOSUB 500        'Enter values and test boundary for root
230     L = 1
240     IF R - L < B GOTO 290
250         GOSUB 700    'Calculate an approximation for the root
260         GOSUB 800    'Calculate the power of the approximation
270         If P > N THEN R = M ELSE L = M
280         GOTO 240
290     'ENDLOOP
300     PRINT
310     PRINT "The root is ";R
320     PRINT
330     PRINT "Do you want to evaluate the formula?"
340     A$ = " "
350     INPUT "Type Y for yes, N for no";A$
360     IF A$ = "Y" OR A$ = "y" THEN ELSE 400
370         GOSUB 900 'Calculate y value in formula, given x as 20
380         PRINT
390         PRINT "When x = 20 in the formula, y = ";Y;" feet."
400       'ELSE
410       'ENDIF
420   END
430   '
500   'SUBROUTINE-Enter values and test boundary for root
510     INPUT "What is the real number N whose root you want to find? N = ";N
520     INPUT "What is the index of the root ";E
530     INPUT "What is the maximum error, you want to allow in the answer ";B
540     A$ = " "
550     IF A$ = "Y" OR A$ = "y" GOTO 660
560         INPUT "Name a value that is an upper (or lower) bound for N";R
570         PRINT "The computer will now test the value you just entered."
580         PRINT
590         T = R
600         GOSUB 800
610         PRINT P
620         PRINT
630         PRINT "Is the value printed above larger (or smaller) than N?"
640         INPUT "Type Y for yes, N for no. ";A$
650         GOTO 550
660     'ENDLOOP
670   RETURN
700   'SUBROUTINE - Calculate an approximation of the root
710     M = (L + R)/2
720     T = M
730   RETURN
800   'SUBROUTINE - Calculate the power of the approximation
810     P = 1
```

```
820      FOR I = 1 TO E
830          P = P*T
840      NEXT I
850   RETURN
900   'SUBROUTINE-Calculate y value in formula, given x as 20
910      T = R
920      E = 20
930      GOSUB 800
940      Q = 1/P
950      Y = (P + Q)/2 + 9
960   RETURN
```

This program was run on an IBM-PC.

```
RUN

What is the real number N whose root you want to find? N = ? 13.92820
What is the index of the root? 30
What is the maximum error you want to allow in the answer? .000001
Name a value that is an upper (or lower) bound for N? 1.5
The computer will now test the value you just entered.

191751.1

Is the value printed above larger (or smaller) than N?
Type Y for yes, N for no.? y

The root is 1.091767

Do you want to evaluate the formula?
Type Y for yes, N for no? y

When x = 20 in the formula, y = 11.98086 feet
```

Student Problem Solvers

Students who solve problems presented in *delta-K* will have their solutions and names published. The students listed below submitted solutions to the problems appearing in Volume 26, Number 2, June 1987. The solution reaching the editor's desk first was submitted by Craig Langston, a Grade 6 student at Parkmeadows School in Lethbridge. Congratulations to Craig and all of the problem solvers.

Craig Langstron
John Quantz
Shannon Boutland
Lindy Provost
Cynthia Wolf Child
Denise Sawchyn
Cobey Farmer
Jim Nelsson
Suzanne Strachan
Jennifer Edwards
Tania Janzen
G.R. Davis
Lennie Melvin
Trenna Waldie
Tany Wadswoth
Amber Dersch-Schneider
Heather Moultan
Carrie Scout
Tyson Voik
Jason Myers
Christie Welsh
Julie Magson

For readers who missed the previous issue, here are the original problems, together with Craig Langstron's solutions.

Final Clearance

—by Kevin J. Sherratt

Problem

Near closing time on the last day of a sporting goods liquidation, only $800 worth of equipment was yet to be sold: 1 canoe @ $160, 3 tents @ $80, 5 sleeping bags @ $40, 6 camp stoves @ $20 and 8 bush knives @ $10.

The next five customers each spent $160, clearing out the last pieces of equipment. From the given clues, find the items that each customer bought.

CLUES

- Brad picked up neither bush knives nor tents.
- Doris bought at least one piece of four different kinds of equipment.
- Andy and Brad each bought 5 items.
- Carla bought at least one knife.

Solution

—by Craig Langstron

- Andy has 1 tent @ $80, 1 sleeping bag @ $40, 1 camp stove @ $20, 2 bush knives @ $10 = $160.

- Brad has 3 sleeping bags @ $40, 2 camp stoves @ 20 = $160.
- Carla has 1 tent @ $80, 2 camp stoves @ $20, 4 bush knives @ $10 = $160.
- Doris has 1 tent @ $80, 1 sleeping bag @ $40, 1 camp stove at @ $20, 2 knives @ $10 = $160.
- Eric has 1 canoe @ $160 = $160.

1 canoe	= $160
3 tents	= $ 80
5 sleeping bags	= $ 40
6 camp stoves	= $ 20
8 bush knives	= $ 10

Tutors

—by Karen M. Gibling

Problem

From the following clues, determine on which day of the week each student tutors.

CLUES

- Jane tutors later in the week than Tony does.
- Jane's day is earlier in the week than Bob's day.
- Frank will tutor on a day that is later in the week than Cathy's.
- Frank will tutor earlier than will Bob.
- Frank, Cathy and Bob will not tutor on Monday.
- Frank cannot tutor on Thursday.

Solution

—by Craig Langstron

Monday	Tuesday	Wednesday	Thursday	Friday
Tony	Cathy	Frank	Jane	Bob

Erratum

In R. Scott Erickson's article "Escher Revisited: Modeling Gradual Deformations Using Logo," published in *delta-K,* volume 27, number 1, June 1988, the three designs appearing on page 20 are improperly credited. "Razor Blades" was created at Carnegie-Mellon University, "Consternation" by Scott Grady and "Fylfot Flipflop" by Fred Watts are designs that originated at the Basic Design Studio of William S. Huff, professor of architectural design at the State University of New York at Buffalo. The designs originally appeared in "Metamagical Themas" written by Douglas R. Hofstadter and published in *Scientific American,* volume 249, number 1, July 1983.

R. Scott Erickson, John Percevault (editor of *delta-K*) and the Mathematics Council of The Alberta Teachers' Association apologize to William S. Huff and students at the Basic Design Studio and Carnegie-Mellon University.

Permission to reprint or reproduce in any way the abovementioned designs must be obtained from *Scientific American* and William S. Huff.

1987–88 MCATA Executive

President
Louise Frame Res. 251-5841
#36, 2323 Oakmoor Drive SW Bus. 278-3633
Calgary T2V 4T2

Past President
Robert Michie Res. 246-8597
149 Wimbledon Crescent SW Bus. 230-4743
Calgary T3C 3J2

Vice President, NCTM Representative and Newsletter Editor
Art Jorgensen Res. 723-5370
Box 2619
Edson T0E 0P0

Secretary
Mary-Jo Maas Res. 553-4848
Box 484
Fort Macleod T0L 0Z0

Treasurer
Dick Kopan Res. 271-5240
23 Lake Crimson Close SE Bus. 271-8882
Calgary T2J 3K8

***delta-K* Editor**
John Percevault Res. 328-1259
2510–22 Avenue S Bus. 329-2185
Lethbridge T1K 1J5

Monograph Editor
Thomas Schroeder Res. 284-3979
3703 Unity Place NW Bus. 220-6173
Calgary T2N 4G4

1988 Conference Chairman and Faculty of Education Representative
Al Olson Res. 435-5427
Dept. of Secondary Ed. Bus. 432-5860
Room 338, Education Bldg. S
University of Alberta
Edmonton T6G 2G5

Mathematics Representative
Dennis Connolly Res. 329-4568
Dept. of Math Sciences Bus. 329-2476
University of Lethbridge
4401 University Drive
Lethbridge T1K 3M4

PEC Liaison
W. Bruce Hopchin Res. 452-0993
13024–104 Avenue Bus. 426-6933
Edmonton T5N 0V8

ATA Staff Adviser
Bill M. Brooks Bus. 265-2672
#200, 540–12 Ave. SW or 1-800-332-1280
Calgary T2R 0H4

Directors
Diane Congdon Res. 527-8978
124 Shaw Crescent SE Bus. 548-7516
Medicine Hat T1B 3P5

Bill Davidoff Res. 627-4283
P O Box 574 Bus. 627-4414
Pincher Creek T0K 1W0 or 627-4415

George Ditto Res. 282-6682
2713–17A Street NW Bus. 294-6309
Calgary T2M 3S9

Nancy Hope Res. 532-5915
8511–96 Street Bus. 539-0333
Grande Prairie T8V 3C8

Jim Johnson Res. 481-0373
#301, 17729– 64 Ave. Bus. 487-0550
Edmonton T5T 2J9

President, South West Regional
Dennis Burton Res.
3406 Sylvan Road Bus. 328-9606
Lethbridge T1K 3J7

PRINTED AT
BARNETT HOUSE