ED 302 231 IR 013 652

AUTHOR	Denenberg, Stewart A.
TITLE	Developing Reasoning Skills in College Freshmen Using Computer Programming, Collaborative Problem-Solving, and Writing.
PUB DATE	88
NOTE	18p.; Paper presented at the Annual Great Lakes/East Coast LOGO Conference (Cleveland, OH, May 6-7, 1988).
PUB TYPE	Reports - Descriptive (141) -- Speeches/Conference Papers (150)
EDRS PRICE	MF01/PC01 Plus Postage.
DESCRIPTORS	*Cognitive Processes; *College Freshmen; Discovery Learning; Higher Education; *Problem Solving; *Programing; Skill Development; *Teaching Methods; *Writing Processes
IDENTIFIERS	*LOGO Programing Language; State University of New York Coll at Plattsburgh

ABSTRACT

This paper describes part of a course for college freshmen entitled "Computation, Reasoning, and Problem Solving," which uses the LOGO programming language to integrate computer programming skills, collaborative problem solving skills, and writing skills. Discussion of the computer programming component includes two of the LOGO problem sets, which compare a priori and a posteriori reasoning, one at the beginning and one at the intermediate level. It is noted that a problem is presented at each level, the LOGO skills necessary to approach the problem are provided, and a guided interaction between the teacher and the student occurs. The two major techniques used to teach collaborative problem-solving are then discussed, i.e., the use of projects that facilitate collaborative learning via top-down design and the Whimbey and Lochhead's "Problem Solving and Comprehension" (1986). The types of writing assignments required by the course are also described, and a summary of the results achieved in teaching the course over the past five years concludes the report. The text is supplemented by two figures and five tables, and sample course materials and an evaluation questionnaire for students are appended. (5 references) (EW)

DEVELOPING REASONING SKILLS IN COLLEGE FRESHMAN USING COMPUTER PROGRAMMING, COLLABORATIVE PROBLEM-SOLVING, AND WRITING

Stewart A. Denenberg
Dept of Computer Science
SUNY, Plattsburgh

INTRODUCTION

College freshman need strong reasoning and logical thinking skills in three areas in order to be successful in most of their courses. They must be able to apply and follow logical reasoning, they must be able solve problems of the type that appear on college and job placement tests, and they must be able to express their ideas in writing. An additional useful skill is the ability to collaborate on a problem as a member of a group.

CSC111 "Computation, Reasoning, and Problem Solving" is offered by the Department of Computer Science to fulfill the Reasoning Skills component of the General Education Program at SUNY Plattsburgh. It is a "survival skills" type of course intended to provide freshmen with the attitudes and skills they need to have successful academic experiences in all of their other courses. Because it is a General Education course, the college requires that it contain a significant writing component and that, if possible, it utilize active and collaborative learning pedagogy (as opposed to straight lecture). To meet all of these goals, the course attempts to integrate skills in three areas: computer programming, collaborative problem-solving, and writing. Each of these areas will be discussed more or less independently in the next three sections, and the final section will summarize the results achieved from teaching the course over the past five years.

COMPUTER PROGRAMMING

When solving problems, many college freshmen have a great deal of difficulty determining when it is appropriate to use an experimental approach (or how to organize an experiment) and when it is appropriate to use the "armchair" method of simple deductive reasoning. Science courses develop the *a posteriori* or empirical approach (effect to cause) and mathematics courses develop the *a priori* or deductive (cause to effect) approach but there usually exist no courses to integrate the two methods or to develop the intuition of when it is appropriate to utilize one or the other, or both, in attacking a problem. Computer programming combines the methods of *a priori* and *a posteriori* reasoning and, as such, can provide the necessary integrative vehicle.

The Logo programming language was chosen for this part of the course primarily because it was designed to facilitate the development of problem solving skills and behaviors by providing a friendly,interesting and active environment for the learner to explore.[1] In fact, many of our computer science faculty were surprised to observe most of the students in CSC111 pick up the ideas of top-down design and recursion more quickly (albeit less substantively) than do our own majors.

The pedagogy is to present the student with problems of increasing difficulty which can be attacked by either the *a priori* or *a posteriori* approach, or both, and then, after the student has had an opportunity to solve the problem, to discuss the merits and tradeoffs between the two approaches. The Logo language is taught spirally where only the minimum command syntax and semantics are exposed to solve the problem at hand; if at a later time more details have to be added to solve a problem of greater difficulty, then we "spiral" back to do so. Thus, a problem is presented, the Logo necessary to approach the problem is covered, and a guided interaction between the teacher and student occurs whereby the efficacy of the two approaches (*a priori / a posteriori*) are compared. This cycle is repeated with a new problem of more difficulty, and that will usually entail learning either more Logo constructs or more depth to the ones already learned, or both. In parallel to this pedagogy the classic techniques of top-down design and debugging are taught.

Two of the Logo problem sets used in the course will be described here (others are available upon request). In order to appreciate the first example, we must first understand Turtle Graphics and five Logo commands.

BACKGROUND: A VERY BRIEF INTRODUCTION TO LOGO

Turtle Graphics is a part of the Logo language that allows the user to draw pictures by moving and rotating an abstraction of a Turtle (a small triangle) on a video screen. The commands to move the Turtle are quite simple: FORWARD moves the Turtle in the direction it is currently pointing the number of "Turtle Units" specified by its argument (e. g. the Apple Logo screen is 279 by 139 Turtle Units) while BACKWARD moves it in the opposite direction. The commands to rotate the Turtle without moving it are RIGHT and LEFT, and they take degrees as arguments. All Turtle commands are relative to the current orientation of the Turtle and not the programmer. Thus, the command sequence to draw a square of side 50 would be:

```
FD 50   RT 90   FD 50   RT 90   FD 50   RT 90   FD 50   RT 90
```

where FD and RT are the Logo abbreviations for FORWARD and RIGHT respectively. Obviously this is cumbersome and so Logo provides a construct for repeating sequences of commands. The above sequence can be replaced with:

```
REPEAT 4 [FD 50  RT 90]
```

and the same square with side 50 will be drawn by the Turtle. The Repeat factor (in this case 4) is applied to the bracketed list of commands.

Beginners are often amazed that

```
REPE ᵀ 3 [FD 50 RT 60]
```

is not the correct ...ogo command to draw an equilateral triangle (because the Turtle turns through the outside or supplementary angles of a polygon) and that

```
REPEAT 360 [FD 1 RT 1]
```

is the Logo "equation" for a circle. Note that the Turtle acts more as a "growing tip" or a mathematical differential than it does a follower of Cartesian coordinates.

THE PROBLEMS

Beginner Level Problem Set

We now know almost enough to examine the first problem set which starts off as a writing assignment:

Type the following procedure definition into your workspace:

```
TO ARC :A :B :C
REPEAT :A [FD :B  RT :C]
END
```

Use both *a priori* and *a posteriori* reasoning to identify measurable outputs when ARC is run with different sets of inputs. Do not attempt to quantify the effects of the inputs on the Turtle's behavior, just write down what you see when you run ARC for systematically varied values of A, B and C.

Note that Logo uses colons (:) in procedure definitions to designate formal parameters (called Inputs in Logo) and so the above code is the definition of a procedure named ARC which has three Inputs named A, B and C. Thus the execution of:

```
ARC 360 1 1
```

is equivalent to:

```
REPEAT 360  [FD 1  RT 1]
```

and will cause a circle to be drawn.

Prior to the assignment, the distinctions between the deductive *a priori* and the experimental *a posteriori* methods of reasoning are discussed and examples of each are given. We are careful to point out that the methods are not mutually exclusive but. in fact, one nourishes the other: we generally begin by using *a priori* reasoning to define and delimit our initial experiment which might in turn uncover a pattern that can lead to a formal mathematical proof of an existing relationship. In this fashion one type of reasoning allows us to "leapfrog" into the domain of the other. We also cover the technique of systematic variation of variables in a scientific experiment so that the individual effect of a variable can be isolated. The student is asked to consider a question of the type, "If you wanted to determine the effect of the amount of salt in a recipe, how would you go about it?" However, actually using systematic variation of variables causes an interesting problem to occur. When we run:

```
ARC 360 1 1
```

we get the expected circle. Now if we change only the last parameter from 1 to 2 we would expect only one effect to change (e.g. the circle would change in size). However, when we run

```
ARC 360 1 2
```

in fact two things change:

1) the size of the circle decreases by half
2) the circle is traced out twice by the turtle!
as shown in Figure 1.

Most college freshmen are not sophisticated enough to recognize this as a nonlinear system, and it is just this fact that makes the problem interesting. Of course, most students investigate many sets of inputs in their attempt to discern changes in the pictures drawn by ARC.

Well, what do the students see as effects when they run ARC? Most see circles of various sizes. Some report squares or triangles even pentagons:

```
REPEAT 3  [FD 50  RT 120]  will draw a triangle
REPEAT 4  [FD 50  RT  90]  will draw a square
REPEAT 5  [FD 50  RT  72]  will draw a pentagon
```

Almost no one sees arcs as an effect although that was the name of the given procedure. Here we bump into a fundamental problem of the scientific method which we can informally state as follows: "Science cannot be completely objective because any theory you devise to explain what you see depends on what you are seeing." Even in a simple problem like this one, if you see circles and I see arcs then we're going to come up with different theories to explain what's happening. As a matter of fact seeing circles is precisely what caused us to be surprised when we changed only the last input in ARC from 1 to 2 as shown in Figure 1. The drawing on the right in Figure 1 is not actually a circle (who ever heard of a circle that goes twice around?) but an arc of exactly the same length in Turtle Units as the drawing on the left side of Figure 1. It differs from the drawing on the left in only one respect: the total angular rotation of the Turtle has doubled. We saw two effects change when we changed one cause only because we insisted on seeing circles rather than arcs.

Therefore the next assignment drops the pretense of perfect objectivity and forces the student to see the same effects I see:

Type the following procedure definition into your workspace:

```
TO ARC :A :B :C
REPEAT :A [ FD :B  RT :C]
END
```

Use it to determine the effects of each of the three inputs A, B and C on:

1. The length of arc as measured in Turtle Units; i.e. the total distance the Turtle travels.

2. The angular amount of arc as measured in degrees; i.e. the total amount the Turtle turns.

3. The time it takes the Turtle to trace the arc.

4. The resolution of the arc ---- is it coarse or is it fine?

Write up the results citing your conclusions and the *a priori* and *a posteriori* evidence for those conclusions. The best solution is to discover formulas for each of the four effects listed above e.g.

1. Length = A + B / 3.14159 (This is not the answer)

If you can't be that quantitative then a statement of the form:
"The Length of arc is somehow dependent on A and B. The bigger they are, the longer the arc drawn" is a good second choice.

It turns out that this Writing/Logo assignment is somewhat of hoax because it can be done pretty much without the aid of a computer, never having to leave the proverbial armchair, using mostly *a priori* reasoning as follows:

When the Turtle moves it does not turn, and when it turns it does not move, therefore we can examine the effects of the FD and RT commands independently. The total distance the Turtle travels is given by:

```
REPEAT :A [FD :B]
```

which, by definition, is equivalent to:

```
FD :A * :B
```

thus the formula for the length of arc (asked for in 1. above) is:

Length = :A * :B

Using similar reasoning we can deduce that

Angle = :A * :C

and the first two parts of the second assignment are solved.

As for part 3, we can also use *a priori* reasoning to deduce that the time to trace the arc is equivalent to the execution time of the ARC procedure which consists of the single Logo command:

```
REPEAT :A [ FD :B  RT :C]
```

Clearly the value of A is proportional to the execution time of the REPEAT command, but do the values of B and C also contribute? Yes and no. Surely it must take longer to execute FD 9999 than it does FD 9, but in fact most of the execution time is taken in the decoding and execution of the REPEAT command itself and not in the propelling of the Turtle. Of course, we could not expect a student to be so knowledgeable regarding the inner workings of a computer, but with very little experimentation (i.e., the *a posteriori* method), it is easy to "see" that it takes the Turtle little more discernible time to go FD 1 than it does to FD 100. Thus using a combination of *a priori* and *a posteriori* reasoning, we conclude that while B and C influence the time, they are "second-order effects" and that it is A which is by far the largest contributor to the running time. Ignoring second-order effects, we formulate this conclusion as:

```
Time = k * :A
```

where k is a proportionality constant that can be reckoned via *a posteriori* reasoning.
The fourth and final part of the assignment asks for the resolution. Here we must be careful to define exactly what we mean by the resolution of a drawing made by the Turtle. When we are discussing a TV screen or a photo in a newspaper, we mean the number of "dots" or pixels per unit area. However, a more informal way to look at it is: if we have two pictures of the same size then the one with the smaller dots will have the higher resolution because we can fit more small dots than large ones into a given fixed size area. Now with Logo, we don't have dots but lines in our pictures so that extending that reasoning: the shorter the lengths of the individual lines comprising the drawing, the higher the resolution, and vi :e-versa so the formula becomes:

```
Resolution = k / :B
```

where k is, once again, a constant of proportionality and thus the resolution is inversely proportional to the value of B.

Many students perceive a relationship between time and resolution and they are partially correct in this observation. For example, if we compare the times and resolutions for the two cases:

```
1) REPEAT 360 [FD 1  RT 1]
2) REPEAT 36  [FD 10  RT 10]
```

we see that the first case has 10 times the resolution of the second but that it runs 10 times more slowly, while just the reverse is true for the second case: it runs 10 times as fast as the first but has only 1/10 the resolution --- it draws a much coarser "circle" than the first. The other important observation to be made here is that both cases 1) and 2) draw exactly the same size "circle" because they both turn the Turtle thru exactly 360 degrees and they both have a circumference of 360 Turtle Units.
If time is not a problem and we want a pretty "circle" then we use 1); on the other hand, if we can live with something a little less exact, then the "circle" drawn by 2) runs much more quickly and it looks fine to most people. If this exercise doesn't get across the idea that Logo "circles" are just various polygons of differing resolutions, then the teacher can have the student try:

```
REPEAT 4  [FD 90  RT 90]
```

which is a very fast running "circle" with only 1/90 the resolution of case 1) but it has exactly the same circumference size (4 * 90 = 360 * 1). The fact that it appears to be a square is amusing but should not be confused with the fact that it is polygons and not circles we are drawing here.

A side benefit to this assignment is the realization by the student that there may be objective criteria by which to judge the value of a piece of text. Many college freshmen are at that stage of cognitive development where they "know" that all evaluation is subjective: "It's just the opinion of the evaluator and nothing more." While such student opinion has been reinforced by the grades they receive on their essays in various writing courses, they generally agree that if one program runs 10 times faster than another but produces the same output (i.e. draws the same picture), then it is better (and deserves a higher grade).
One of the most important lessons to be learned from this assignment is to use *a priori* reasoning first, for two important reasons:

1) *A priori* is a stronger form of reasoning than *a posteriori*. A mathematical or deductive proof is more compelling than experimental evidence not only because it is more of an abstraction and thus appears to be more generally applicable, but also because a new experiment is more likely to refute previous results than is a new mathematical proof likely to disprove a previous one (although this sometimes happens).
2) If the *a priori* method is successful then we need not even resort to an actual experiment to solve our problem; we will have succeeded with just a "thought experiment" whereas the reverse is generally not the case.

ust resist the temptation to hastily construct an experiment to answer the question, "How many married elors are there in Plattsburgh NY?" when a bit of carefully applied deductive reasoning will suffice.

The next assignment in this Beginner's Level Problem Set asks the student to write four quarter-circle procedures that accept either the radius or circumference as Inputs and either run fast at low resolution or slowly at high resolution. If the student has had difficulty absorbing the lessons of the previous assignment, this is a good opportunity for the teacher to model another very useful and powerful problem-solving technique: "Solve a simpler Problem"

In this technique (made famous by Polya [2]) we attempt to solve a simpler version of the problem in the hope that some of the insights gained will transfer back to the original problem. If we can generalize our solution process, perhaps we will be in a better position to re-attack the more general original problem. It is important for the student to realize this strategy is inherently different from top-down design which attempts to simplify a problem by breaking it down into pieces but with no loss of generality.

In this particular instance, the simpler problem would be to:

> Write a quarter-circle procedure that accepts the circumference as an Input (don't worry about the resolution and the time just yet)

After the simplification of ignoring resolution and execution time has been made, we can begin our attack on the simpler problem of determining values for X, Y and Z in the following unfinished procedure definition (the values of X, Y and Z are to be determined):

```
TO QCIRCLE  :CIRCUMFERENCE
REPEAT X  [FD Y  RT Z]
END
```

Since the problem specifies that only a quarter of a circle be drawn and that its arc length be equal to the value of the Input CIRCUMFERENCE, the question becomes how to translate those specifications into formulas or relationships between X, Y and Z ? Using results gained from the prior assignment:

The first specification that a quarter circle be drawn translates to:

1) X * Z = 90 (Degrees)

The second specification that the arc length of the drawn figure be equal to the Input translates to:

2) X * Y = :CIRCUMFERENCE

We see that the above two relationships indicate that once a value for X has been chosen, the values for Z and Y are then fixed; e.g. a natural choice for X might be 90 and that fixes Z to 1 by 1) above and Y to :CIRCUMFERENCE/90 by 2) above so that the solution to our simplified problem becomes:

```
TO QCIRCLE  :CIRCUMFERENCE
REPEAT 90  [FD :CIRCUMFERENCE / 90    RT 1]
END
```

Next we examine our solution to the simplified problem in the hope we will be able find insights that will allow us to generalize and eventually solve the original harder problems. We begin by asking ourselves, "Exactly what problem is the above procedure definition a solution to?". Because the value of X is high (90) it is clear that this solution will run slowly and because the Input to the FD command is the total arc length divided by a large number (90) we can see that the resultant resolution will be very high (as each line segment will be very small). Thus the above procedure definition represents a solution to the high resolution, long time situation.

Next, how can we adjust the values of X, Y and Z to speed up the execution and lower the resolution? From the previous assignment we know that X is the prime determiner of execution time so if we make X smaller by a factor of 10 then we should speed up the subsequent drawing of the quarter circle by that same factor of 10. This hypothesis leads us to the unfinished command:

```
REPEAT 9  [FD Y  RT Z]
```

Now, if we reapply relations 1) and 2) we can determine the values for Y and Z above so that our faster, lower resolution version of the procedure becomes:

```
TO QCIRCLE :CIRCUMFERENCE
REPEAT 9 [FD :CIRCUMFERENCE / 9   RT 10]
END
```

In this fashion, the appropriate procedure definitions can be derived when the radius rather than the circumference is the Input argument; the only caution would be to make sure everyone understands that the arc length for a quarter-circle is: 2*PI*RADIUS/4. Advanced students can be given the more general problem of accepting the Resolution as well as the Radius as Inputs.

This problem set culminates in a large Picture Project where the student is asked to write a set of Logo procedures to draw a house, tree(s), flower(s), bird(s), cloud(s) and a sun arranged as a pretty picture on the screen. Inputs to the main or driver procedure must specify the scale of the drawing as well as the number of trees, flowers, etc. The problem of preserving relative distance between objects as the scale changes allows the teacher to restrict the problem solution to the relative "coordinates" of Logo's preferred mode; i.e., allowing only the LT, RT, FD and BK commands. The pedagogical advantage to this approach is that it forces the student to view the solution from the Turtle's point of view and hence it can be argued that this mode of graphics moves the problem-solver into or at least closer to the problem itself. Alternatively, the teacher can use this assignment as an opportunity to teach the Logo commands that address the screen via Cartesian coordinates (see reference [3]). Of course, the advanced student can be asked to do both and compare the resulting two programs in terms of ease of writing and debugging as well as memory space and execution time tradeoffs.

The Picture Project also presents an opportunity to combine the top-down design technique (this is a large program with a moderate degree of complexity) with the insights gained from the previous Quarter-Circle procedures. Most of the objects to be drawn (trees, flowers, clouds, birds and sun) can be composed of quarter circles (an amazing coincidence...) and the student can be made to see the value of a fast-running QCIRCLE procedure: a drawing that previously took 5 minutes to execute can be made to run in 30 seconds with very little loss of visual resolution. It is indeed a powerful idea when the student realizes that all of the procedures (BIRD, BIRDS, etc.) rest like an inverted pyramid on the QCIRCLE subprocedure and that a well-designed program (a collection of independent modules) will easily allow a slow QCIRCLE module to be "unplugged" and replaced with a speedier version. After the students have gained an appreciation of exactly how difficult it is to actually write a large program of independent modules, the politically sensitive teacher can point out that is precisely the claim made for the SDI (Star Wars) software which will be debugged just once: under actual battle conditions.

Intermediate Level Problem

The next level of problem also starts off as a writing assignment:

Investigate the effects of the two Inputs to SHAPE on the figure drawn:

```
TO SHAPE :MULT :SIDES
REPEAT :SIDES [FD 50   RT 360 * :MULT / :SIDES]
END
```

State your conclusions first, then follow with your *a priori* and *a posteriori* evidence to support those conclusions.

The SHAPE problem (which is discussed in Thornburgh [4]) is a better example of combining a priori and *a posteriori* reasoning in order to come to a solution. His treatment is almost exclusively *a posteriori* and so I will concentrate here on what can be deduced via a priori reasoning.

As with the earlier ARC problem, the first thing we have to do is decide on what effects we are observing as we vary the values for MULT and SIDES. After the students have had a chance to report what they see, I subjectively decide (as does Thornburgh also) that only two effects are visible: we either get a drawing of a simple polygon with :SIDES sides or a star with :SIDES points. Thus the problem may be restated as, "What is the relationship between MULT and SIDES that determines which kind of picture will be drawn?" If I know that relationship and you tell me the values you will be using for MULT and SIDES when you run SHAPE, then I should be able to tell you exactly what kind of picture (a star or a polygon) will be drawn.

We begin by using *a priori* reasoning to determine and delimit the experiment we will run using *a posteriori* reasoning. If the student is allowed to plunge into indiscriminate experimentation starting off by examining, say, 15 values each of MULT and SIDES then 225 cases would have to be investigated for possible patterns. We can show the student how, using *a priori* reasoning, we can cast out most of these cases making for a much more efficient experiment. As shown in Appendix 1, once a value for SIDES has been chosen, we can delimit MULT to the range:

2 <= MULT <= SIDES - 1

The next phase of investigation is via *a posteriori* reasoning. Although it is possible to further constrain MULT using pure *a priori* argumentation, most students (or teachers for that matter) would not think to make that argument without a simple experiment which indicates a symmetry for many combinations of MULT and SIDES. For example, the patterns for SHAPE 1 7 through SHAPE 6 7 are shown in Fig. 2. Thus, by observation, we note that SHAPE 1 7 is congruent to SHAPE 6 7, SHAPE 2 7 is congruent to SHAPE 5 7 and SHAPE 3 7 is congruent to SHAPE 4 7 so that we only have to examine half of the possible cases. Once this insight has been obtained using *a posteriori* reasoning, we work to formalize it using *a priori* reasoning (shown in Appendix 2) and this leads us to the conclusion that MULT can, with no loss of generality be further constrained in its upper range:

2 <= MULT <= SIDES/2

Now we are in a position to use the results of our *a priori* labor by running the experiment with a vastly reduced number of values for MULT and SIDES. We can structure this *a posteriori* exercise by providing the students with a table of the form shown in Table 1. We usually encourage the student to run the value of SIDES up to at least 13 to be sure the pattern becomes visible. To further automate the process we encourage the student to write a DRIVER program that given a value for SIDES will run MULT thru its range from 2 to SIDES/2 with appropriate WAIT commands so that Table 1 can be filled in as the program runs.

The interesting conclusion to this problem is that we only get a star with :SIDES sides if MULT and SIDES are relatively prime; i.e. have no common divisors. The most important lesson to be gained from this problem is that even though *a priori* reasoning should be applied before considering *a posteriori* reasoning, they are not antithetical --- in fact one process "primes the pump" for the other and they provide a mutual nourishment that leapfrogs the problem-solver toward the solution: a formal proof suggests an experiment and the experiment evokes a pattern that suggests a new formal proof...

COLLABORATIVE PROBLEM-SOLVING

Many of the classes are held in our Microcomputer Laboratory where the students work on Logo problems of the type discussed in the previous section. Collaborative learning and group problem-solving occurs almost automatically in this environment. Some students choose to work alone, and some form groups (I limit the size of these groups to two for logistic as well as pedagogic reasons) but a sense of "community" quickly develops as individuals and groups share their solutions and bugs (which are oftimes more interesting than the correct solution) . It is easy and natural for the teacher to encourage and facilitate this type of learning in the laboratory environment.
Another positive feature of this environment is that it causes the students to ask questions; in fact, I spend most of my time scurrying around the lab trying to answer the student's questions (having to give them numbers as they do at a bakery or meat market to insure a first-come first-served queue discipline).

Contrast this to a lecture pedagogy where the teacher proposes mind-stretching questions which promptly fall into an abyss of student silence while the teacher attempts to appear cheerful but underneath the heart sinks and the stomach churns. In any case, I firmly believe that a student only listens when the student and not the teacher has asked the question. When I am asked a question by a student in the Lab he or she is "ripe" to hear me and learn from my answer to their question; when I lecture I am not nearly so certain that learning is occurring.

As mentioned previously, the PICTURE project facilitates collaborative learning via the technique of top-down design. Groups naturally form where individuals take responsibility for pieces of the total picture. Another collaborative project sometimes assigned is the design and implementation of an ALPHABET: the end result is to send large letters to the screen as the associated keys on the keyboard are pressed. While I usually supply the driver program, the students work together to determine a reasonable set of core shapes that can be used to assemble each letter of the alphabet. Some lively discussions develop around the ideas of primitives and parsimony that are reminiscent of an upper-level computer science class in program design.

The second major technique for developing collaborative problem-solving skills is provided by the Whimbey and Lochhead Workbook [5]. The Workbook concentrates on the development of skills useful for solving the types of problems found on college and job placement (and even "IQ") tests so that if nothing else measurable happens in the course at least most students IQ will "increase" as they become more proficient test takers.

The basic technique is to divide the class into pairs of students, one of whom plays the role of the problem-solver and the other the role of listener. The problem-solver's job is to engage the problem in the Workbook by verbalizing the problem-solving process as much as is possible in order to bring the process up to a conscious level where it can be examined. The job of the listener is to keep the problem-solver vocalizing and to keep him or her honest by continually checking the accuracy of the reasoning process and, if need be, stopping the problem-solver and making her or him explain any unclear ideas. The listener is not allowed to suggest how to solve the problem, or offer any hints, or even be thinking of an alternative solution. When the process is working, the classroom is full of activity (and noise) and the main job of the teacher is to walk around coaching the students how to play their respective roles.

I have found it to be useful pedagogic technique to bring in a tape recorder to the second or third class and, as a class demonstration, record the process of two students as they play their roles of listener and problem-solver. If they are both performing their jobs effectively, I should be able to reconstruct their solution on the blackboard just from their dialogue.

Following each problem in the Workbook is a transcription of the problem-solving process of an experienced problem solver (a graduate student or a professor) which problem-solver and listener can only read after they have produced a solution to the given problem. The problem-solver and listener then switch roles and begin the next problem in the Workbook. The problems are graduated in difficulty to develop confidence in the learner and are in the categories of Word Problems, Analogies, and Analysis of Trends and Patterns (Sequence problems). The Whimbey and Lochhead Workbook teaches some very concrete and useful problem-solving techniques such as how to convert a word problem into a table of data and how to draw diagrams expressing complex relationships that may exist. The student is shown how to translate a problem from its English language statement into a format that organizes and "remembers" the information so that relationships are more clearly discerned and deductions can be made more easily . It is a very pragmatic approach that helps to develop a fearless, positive attitude towards problem-solving; the transcriptions clearly show that even experienced problem-solvers are not afraid to use "baby" methods to solve the problem at hand (such as counting on fingers, talking thru the problem of discerning one's left hand from one's right. Anything that works that is neither illegal nor immoral is OK). The students invariably comment that they wish they had been introduced to the Whimbey and Lochhead Workbook before they came to college.

As an example of converting a word problem into a table of data that organizes and "remembers" information, consider the following problem from the Workbook:

> Three fathers ---Pete, John and Nick--- have a total of 15 children of which 9 are boys. Pete has 3 girls and John has the same number of boys. John has one more child than Pete, who has 4 children. Nick has 4 more boys than girls and the same number of girls as Pete has boys. How many boys each do Nick and Pete have?

facts and therefore it is difficult to organize them. However, a table of the type shown as Table 2 can help in both respects. Once the structure of the table is provided, we can mechanically map each sentence of the problem statement into data that is stored and thus rememoered by the construct of the table. For example, the first sentence of the problem can be mapped to the result shown in Table 3. Then, using simple deductive reasoning we can produce Table 4. The point here is that our minds are small and our problems are big so we need some sort of construct that helps us remember and organize information (i.e. know it) so that we can then apply simple deductive logic to what we know. The rest of the sentences in the problem statement also map easily onto the table structure convincing most students that they can solve complicated problems of this type with a simple and powerful technique; thus not only their problem solving skills but their very attitude (which is a prerequisite to acquiring skills) is improved.

WRITING

Writing has long been recognized as an effective way to organize and examine one's thoughts and thus its practice can make one a better thinker and a better problem-solver. Unfortunately writing courses may lack focus in a particular content area so that the exercise of writing is not relevant to the student. As discussed in a prior section, writing assignments are coupled to the Logo projects and are meant to integrate and digest the learning of the a priori and *a posteriori* reasoning methods. Several of the Logo projects require a written analysis using both *a priori* and *a posteriori* reasoning. Typically, a Logo program is given to the students to examine and determine what effects the Inputs (causes) will have on the output pictures (effects). The students must present their conclusions in quantitative terms and back up their claims with evidence clearly identified as *a priori* or *a posteriori*.

Additionally, several writing assignments require the student to create a problem and a detailed description of the solution process of the type studied in the Whimbey and Lochhead Workbook. In the subsequent class the students grade each others' papers (teaching two sections expedites this strategy) and they learn firsthand how difficult it is to create and show the solution to problems so that they are clear and understandable. The criteria that they must use to grade another student's paper is shown as Table 5. A student has the opportunity to resubmit the paper to the teacher if they disagree with their peer grade --- in fact, this resubmition seldom occurs even though the students are much harsher and more unforgiving graders than I would ever be .
All writing assignments are required to be on a word processor. Software such as the Bank Street Writer is taught early in the course using the excellent "flip" side self-tutorial. The first writing assignment is to recount the frustrations and glories of the process of finding the Microcomputer Lab, interacting with its bureaucracy, and finally, learning the Bank Street Writer and writing the essay. The experience involves writing, reflection and, of course, problem solving. I stress the value of a positive attitude in this and any type of problem-solving activity, reminding them that no matter how weird or frustrating this assignment becomes, it can be turned to the student's advantage by writing an entertaining paper about it. Thus the paper integrates four activities: it teaches the rudiments of word processing, it provides the first problem for the student to solve, it is a writing assignment, and it acts as a catharsis for what is usually a stressful experience.

The word processing *of* writing assignments not only produces legible papers, it allows the teacher to enforce rigorous writing standards. If a paper has any spelling errors or grammatically incorrect structures a failing grade is given with the option to resubmit a corrected version for a grade up to a B. Most students learn very quickly to live up to the high standards asked of them.

CONCLUSIONS

The last five years of experience teaching CSC111 "Computation, Reasoning, and Problem Solving" clearly shows mixed results on the development of the students' reasoning skills. A few never get it at all (these students seem to need a remedial course to learn even more basic skills such as reading,multiplying fractions, etc.), a few swallow the course whole, stretching their minds and confidence in themselves as effective problem-solvers, while most gain an excellent grasp on the Whimbey and Lochhead problem-solving techniques as well as a fairly good understanding of the two types of reasoning and the power of applying them within the context of a computer programming language such as Logo.

I get very good feedback from the students on the value of the course despite the fact that they typically have to spend about 10 hours per week outside of class in the Microlab. More importantly, it's fun for me to teach. Logo and the English language are rich enough to keep me continually intrigued, and I really enjoy watching the intellectual growth in most of the freshmen who take this course. As the last half of the course is spent in the Microlab, the students are solving problems pretty much on their own. When I tell them at the end of the semester that they have become autonomous learners, that they don't need me, that they can learn by themselves, and that was the real goal of the course, the silence is stunning as the realization sinks in. They finally believe that they can solve problems, that they can reason and learn on their own, and that result is very gratifying.

REFERENCES

1. Papert, S., Mindstorms, Basic Books, 1980
2. Polya, G., How to Solve It, Princeton University Press, 1945
3. Davidson, L., Apple Logo Reference Manual, Logo Computer Systems Inc., 1982
4. Thornburgh, D., Apple Logo, Addison Wesley, 1983
5. Whimbey, A., & Lochhead, J., Problem Solving and Comprehension, Lawrence Erlbaum Assoc., 1986

APPENDIX 1: PROOF THAT 2 <= MULT <= SIDES - 1

1. When :MULT = 0 we have REPEAT :SIDES [FD 50 RT 0]

2. When :MULT = SIDES we have REPEAT :SIDES [FD 50 RT 360] so that cases 1 and 2 above are equivalent to FD 50 * :SIDES and are degenerate cases not worthy of examining further.

3. When :MULT = 1 we always get a closed polygon with :SIDES sides.

4. When :MULT > :SIDES the drawings will start to repeat themselves.

For example, when :MULT is one more than :SIDES we have

```
    REPEAT :SIDES [FD 50 RT 360 * (:SIDES + 1) / :SIDES]
```

which is easily shown to be equivalent to the case when :MULT = 1 (we use the identities RT A + B = RT A RT B and RT 360 = RT 0 = NO OPeration and we need examine only the Input to the RT command):

Thus we start by examining:

```
     RT 360 * (:SIDES + 1) / :SIDES =
     RT 360 * (1 + 1 / :SIDES) =
     RT 360 + 360 * (1 / :SIDES) =
     RT 360   RT 360 * (1 / :SIDES) =
     RT 360 * (1 / :SIDES)
```

which is the angle when :MULT = 1. Thus we have shown that the drawing we get when :MULT = :SIDES + 1 is the same one we get when :MULT = 1. This result can be easily generalized (and makes a good homework assignment) from 1 to N to prove that when :MULT = :SIDES + N we get the same drawing as when :MULT = N

Therefore, given a value for SIDES, we can limit our investigation of :MULT to 1 <= :MULT <= :SIDES - 1 because we can predict all other values for MULT.

APPENDIX 2: PROOF THAT 2 <= MULT <= SIDES/2

We can begin our *a priori* investigation by using an *a posteriori* crutch: examining a few special cases in the hope that the general principle will shine through (another useful problem-solving technique used by scientists and mathematicians alike).

a) Prove SHAPE 1 7 is symmetric to SHAPE 6 7
This is equivalent to proving the drawing produced by
REPEAT 7 [FD 50 RT 360 * 1 / 7]
is congruent to the drawing produced by
REPEAT 7 [FD 50 RT 360 * 6 / 7]
Clearly the crucial part is the examination of the angle:
RT 360 * (1/7) turns right through 1/7 of a full turn and
RT 360 * (6/7) turns right through the other 6/7 --- but this is equivalent to LT 360 * (1/7) so we see that SHAPE 6 7 is just SHAPE 1 7 with RT replaced by LT.

b) A similar argument can be made for proving SHAPE 2 7 is symmetric to SHAPE 5 7.

Now we try to generalize any insights acquired. For any value of SIDES, we want to show the congruence in the drawings when we have MULT values of:
1 and (:SIDES - 1)
2 and (:SIDES - 2)
3 and (:SIDES - 3)
etc. until :MULT = :SIDES / 2 because the pairings exhaust all cases at this point (in our example .SIDES = 7).

Let us examine the first symmetrical pair of 1 and (:SIDES - 1)
Here we must prove that
REPEAT :SIDES [FD 50 RT 360 * 1 / :SIDES]
is congruent to
REPEAT :SIDES [FD 50 RT 360 * (:SIDES - 1) / :SIDES]
which is equivalent to proving that
RT 360 * (:SIDES - 1) / :SIDES is equivalent to LT 360 * (1 / :SIDES) i.e., the congruency is merely due to LT replacing RT in
REPEAT :SIDES [FD 50 RT 360 * 1 / :SIDES]
This is straightforward because (we use the identity: RT -X = LT X):
RT 360 * (:SIDES - 1) / :SIDES = RT 360 * (1 - 1 / :SIDES) =
RT 360 - 360 / :SIDES = RT 360 RT (-360 / :SIDES) =
LT 360 / :SIDES = LT 360 * (1 / :SIDES).

Similar arguments can be made for values of MULT = 2,3 ... up to and including :SIDES/2. It is also possible to furnish a geometrical argument (proof by pictures) to show
RT 360 * (A / N) = LT 360 * (B / N)
iff A + B = N and A, B, N are positive integers. This means that, because of symmetry, we can limit our test cases to the upper range:

:MULT <= :SIDES / 2.

Therefore we can further constrain our testing range to:

2 <= :MULT <= :SIDES / 2

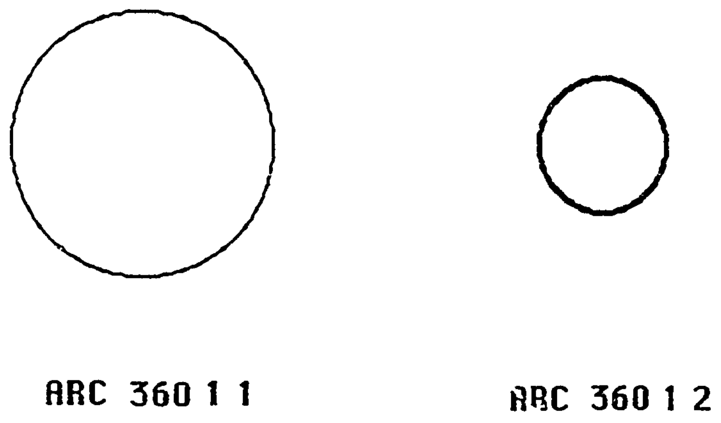

Figure 1. Changing one Cause Produces two Effects

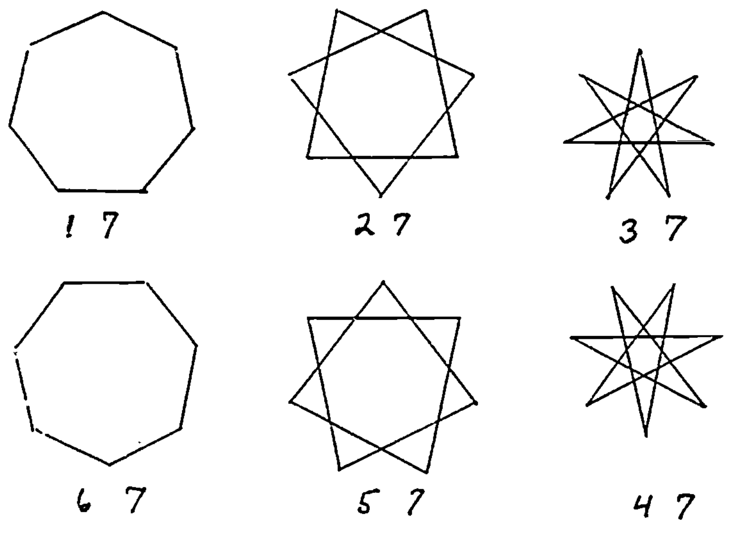

Figure 2. Six Runs of the SHAPE Procedure Drawings have Inputs (MULT and SIDES) Displayed Below them.

SIDES	MULT	STAR with :SIDES Points ? (Yes or No)
4	2	?
5	2	?
6	2	?
6	3	?
7	2	?
7	3	?
8	2	?
8	3	?
8	4	?
.	.	etc.. .
.	.	.
.	.	.

Table 1. Experiment to Determine the Relationship Between MULT and SIDES

	BOYS'	GIRLS	TOTAL
PETE			
JOHN			
NICK			
TOTAL			

Table 2. A Table Provides the Structure to Organize and Save Information

	BOYS	GIRLS	TOTAL
PETE			
JOHN			
NICK			
TOTAL	9		15

Table 3. Mapping the First Sentence Onto the Table

	BOYS	GIRLS	TOTAL
PETE			
JOHN			
NICK			
TOTAL	9	$\overset{\text{DEDUCED}}{\boxed{6}}$	15

Table 4. Deducing 6 Girls

EVALUATION SHEET FOR WRITING ASSIGNMENTS

The grade you assign should be based on five factors:

1. Clarity. Is the paper easy to read and to understand? Is it clear what the author is trying to say?

2. Organization. Does the paper flow nicely? Are the ideas organized so that you can easily follow them?

3. Writing Standards. Does the paper meet the course Writing Standards? Is it neatly typed and double-spaced? Are the grammar and spelling correct?

4. Accuracy. Does it meet the requirements of the assignment? Are there any logical or reasoning errors ?

5. Creativity. Does it show imagination? Does it make you smile?

CHECK THE APPROPRIATE BOX BELOW:

	POOR E	FAIR D	AVG C	GOOD B	OUTSTANDING A
CLARITY					
ORGANIZATION					
WRITING STANDARDS					
ACCURACY					
CREATIVITY					

COMMENTS:

1. What you liked about the paper:

2. What you didn't like:

3. Suggestions for improvement:

FINAL GRADE IS ______

Table 5. Criteria for Peer Grading