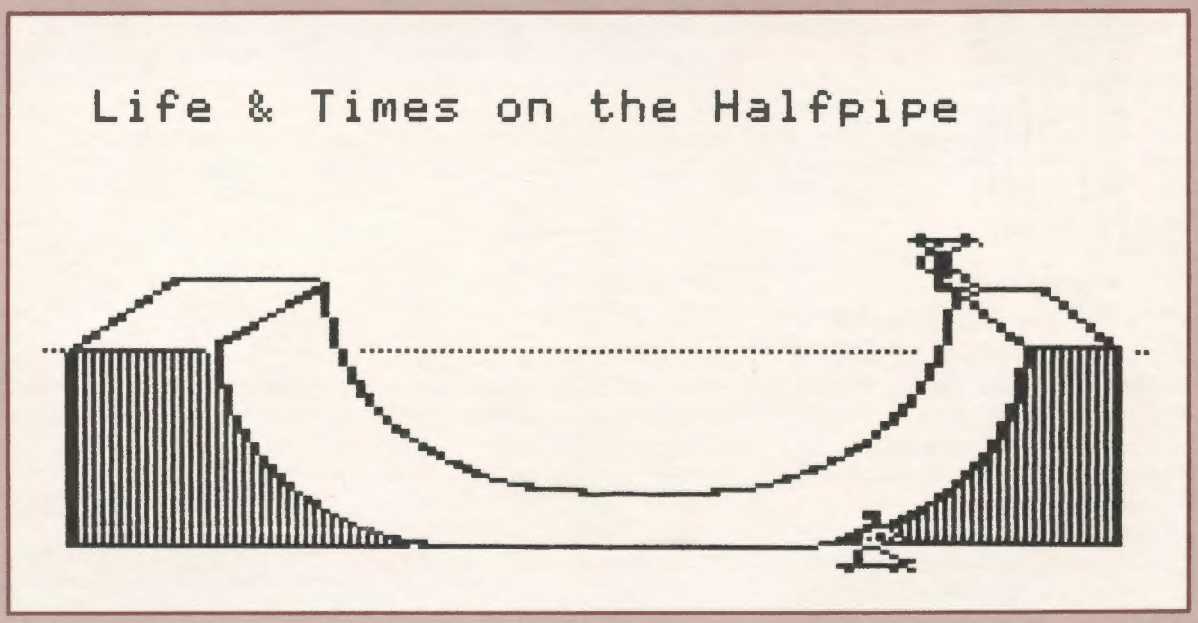
Journal of the ICCE Special Interest Group for Logo-Using Educators

LOGO EXCHANGE

APRIL 1988

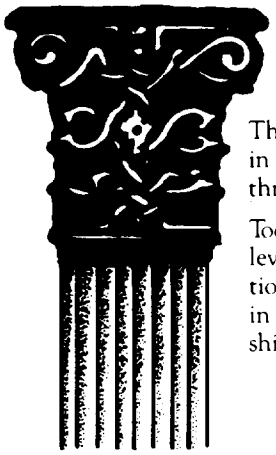
VOLUME 6 NUMBER 8



International Council for Computers in Education

ICCE Publications

ICCE



About ICCE

The International Council for Computers in Education was founded by Dr. David Moursund in 1979 as an organization that would foster appropriate instructional use of computers throughout the world.

Today ICCE is the largest professional organization for computer educators at the precollege level. It is nonprofit, supported by 14,000 individual members and more than 50 organizations of computer-using educators worldwide. These organizations are statewide or regionwide in scope, averaging 500 members each. Approximately 84% of ICCE's individual membership is in the United States, 12% is in Canada, and the remainder is scattered around the globe.

About *The Computing Teacher*

ICCE publishes *The Computing Teacher* journal. *The Computing Teacher* provides accurate, responsible, and innovative information for educators, administrators, computer coordinators, and teacher educators. The journal addresses both beginning and advanced computer educators through feature articles, columns, software reviews, and new product and conference listings. Contributors to *The Computing Teacher* are leaders in their fields, consistently providing the latest information in computer education and applications.

Publications, Special Interest Groups

In addition to *The Computing Teacher*, ICCE provides a number of publications to computer-using educators. ICCE's Special Interest Groups provide in-depth information for computer coordinators, teacher educators, computer science educators, and Logo-using educators. *C.A.L.L. Digest* is published nine times per year for ESL teachers. ICCE committees address a variety of ethical and practical issues important to the computer-educating community.

Independent Study Courses

ICCE offers graduate-level independent study courses, designed to provide staff development and leadership. These courses have been approved by the College of Education at the University of Oregon and carry graduate credit from the Oregon State System of Higher Education. Participants correspond with instructors by mail.

Write for information and a free catalog today!

ICCE • University of Oregon • 1787 Agate St. • Eugene, OR 97403 • Ph: 503/686-4414

LOGO EXCHANGE

VOLUME 6 NUMBER 8 Journal of the ICCE Special Interest Group for Logo-Using Educators APRIL 1988

Logo Exchange is the journal of the International Council for computers in Education Special Interest Goup for Logo-using Educators (SIGLogo), published monthly September through May by ICCE, University of Oregon, 1787 Agate Street, Eugene, OR 97403-9905, USA.
POSTMASTER: Send address changes to Logo Exchange, UofO, 1787 Agate St., Eugene, OR 97403.

CONTENTS

SIGLogo Membership *(includes The Logo Exchange)*

	U.S.	NON-U.S.
ICCE Member	24.95	29.95
Non-ICCE Member	29.95	34.95
ICCE Membership *(includes The Computing Teacher)*	28.50	31.50

Send membership dues to ICCE. Add $2.50 for processing if payment does not accompany your dues. VISA and Mastercard accepted.

The *Logo Exchange* is produced on a Macintosh SE and Laser Writer donated by Apple Computer, Inc.

(LOCAL "PAGE)

From the Editor

Who owns Logo?

by Sharon Burrowes Yoder

More correctly, who owns Logo ideas: those who developed Logo? those who sell Logo? Logo users? Papert? No one? Everyone? Such a seemingly innocent question doesn't seem to have a simple answer at all.

In the past few months, this simple-complex question has arisen numerous times as I have prepared issues of *LX* for publication. Let me give you some background.

Frequently authors and columnists "footnote" code contained in their articles, sometimes as their own and sometimes as belonging to others. It is not uncommon for me to recognize this code as the product of some of the very earliest work in Logo, often having its roots before Logo was even available for microcomputers. Many of the references given by these authors are not technically accurate because they don't give credit to the originators of the ideas.

Allow me to cite a few examples. The PICK procedure used to randomly select an item from a word or list is practically a Logo primitive for those working in traditional list processing. Often references in *LX* articles cite Glen's work...but I personally first saw the PICK procedure in Hal Abelson's books before *LX* even existed, and undoubtedly that procedure was part of even earlier work with Logo. Glen, who is well aware of Abelson's work, first used PICK in *LX* in a column in 1984. So, who "owns" PICK?

Another example is dynamic turtle microworlds. Dan Watt's books (*Learning With Logo*, McGraw Hill, 1984) are often cited here, and yet much of the work with dynamic turtles goes even further back to Andy diSessa's work in the early 1970's at MIT. Does someone "own" the dynamic turtle concept? I have seen references to language microworlds such as Bob Lawler's Beach World attributed to both Michael Tempel and Gary Stager, both of whom extended the ideas, but they did not originate them.

There seem to be three things going on here. First there is the "requirement" placed on those who publish articles to give appropriate references. From a scholarly point of view, accurate citation of sources is certainly important. Proliferation of misinformation is not something any of us wants to foster, and it is certainly not a good model to set for our students in any case.

Secondly there is the sharing that is an innate part of the Logo culture and enviroment. We encourage our students to work together, to help each other, to share ideas. We share our ideas among ourselves at professional meetings and over telecommunications networks. We seldom think about ownership in these situations.

Third, there is the natural tendency of human beings to express pride in their accomplishments. Thus, when an author or columnist develops or extends an idea, it is only natural for him or her to take credit for the work.

What is the place of each of these viewpoints in the Logo literature? As I see it, those who write about Logo fall into three categories. First is the scholar writing research articles or Ph.D. dissertations. Here accurate citation of original sources is clearly appropriate.

Second are those individuals with deep roots into the history and philosophy of Logo. These people have either been working with Logo for a great deal of time or have read much of the early Logo literature, and thus are aware of the historical roots of many Logo ideas. Those with such deep roots should share in knowledge with the rest of us, especially when such knowledge is anecdotal and unlikely to be cited elsewhere.

Finally, there is the newcomer to Logo who is extremely enthusiastic about his or her Logo experiences, but lacks the depth of the researcher or the historian. Such individuals should feel free to openly share their personal experiences. Their point of view is a valuable as that of the scholar or historian.

Which of these viewpoints belong in *LX*? To that my answer would be all three. We need the enthusiastic newcomer to tell us that Logo is still as exciting as it once was to those of us with experience. We need to hear from those with deep roots in the Logo culture to give us perspective. And, certainly, we need to hear from the scholars in order to know what is going on in the research community.

So, perhaps, my concern has really been unfounded. In keeping with the Logo philosophy, we should all be more accepting of a variety of points of view from all quarters. Only through sharing can we all grow and learn.

[Sharon Burrowes Yoder, ICCE/SIGLogo, University of Oregon, 1787 Agate St., Eugene, OR 97403-9905].

Cover: Mark Benda was in the 7th grade when he made this skateboard program. He is in the Walker Jr. High School Computer Club in La Palma, California. Mark thinks Logo is neat to make shapes, draw scenes and animate pictures.

Monthly Musings

Hyper Hype III

by Tom Lough

In the February and March musings, I introduced HyperCard and explored some specific comparisons between features of HyperCard and those of Logo. I'd like to wrap up this series of musings with a look at one of the most exciting capabilities of HyperCard. This capability is shared with Logo and other powerful computing languages, and brings HyperCard reasonably close to what some might term an artificial intelligence environment. In a word, I'm talking about self-modification.

The properties of various HyperCard objects make self-modification possible. For those of you familiar with the Logo property lists, the concept of properties is not totally foreign. However, the idea is applied a little differently in HyperCard.

HyperCard provides nearly 40 properties which can be applied to a variety of objects. The properties are gathered into global, window, stack, background, card, field, button, and painting property groups. Properties include such things as location, name, text font, and the like. The values of the properties are manipulable by HyperCard instructions within scripts.

The particular property which makes self-modification possible is the *script* property. It roughly corresponds to the TEXT operation of Logo, in that it provides access to the text of HyperCard scripts. This means that the text of a script (of a card, button, etc.) can be modified from within the same or another script. Thus, it is possible to devise stacks with scripts which can change themselves. Scripts might be able to "learn" from experience and change their performance accordingly. This powerful capability suggests that there may be some artificial intelligence related applications in HyperCard's future.

Experienced Logo users have been experimenting with self-modifying procedures for several years. There are many well known examples of changing values of variables as a means of modifying the behavior of the procedures as they run. For example, see the ANIMAL procedures in Abelson's BYTE/McGraw-Hill series of Logo books (e.g., *Apple Logo*, p. 162 ff.).

It is also possible to change the behavior by modifying the Logo procedures themselves. For LX readers who have not yet given this activity a try, here is a brief example to introduce you to the idea. Once, when I was using Logo as the basis of an algebra exploration, I wanted a procedure which looked like this:

```
TO  X
 OUTPUT 42
END
```

The actual number to be output by the X procedure would change from time to time. In fact, I wanted this number to be assigned by a random number generator at specific points. Since the class members had not had much computer experience, I wanted to stay away from the quotes-vs-dots syntax problems. I designed an additional procedure to redefine the X procedure with a random number as the output value.

```
TO REDO
 DEFINE "X LIST [ ] LIST "OUTPUT (1 +
   RANDOM 50)
END
```

Here, I used the fact that the text of a Logo procedure is just a list of two lists. The first list contains the names of any input variables. In the X procedure, there are none, as indicated by the empty brackets [ ] in the DEFINE line above. The second list consists of procedure instructions. Here, I wanted OUTPUT and a random number from 1 to 50.

To see the text of a Logo procedure, use the TEXT operation.

```
SHOW TEXT "X
[ [ ]  [ OUTPUT 42 ] ]
```

An IS procedure helped create a sort of "natural language" feel to the experience. [Note: The IS procedure will not work for Apple Logo I, Terrapin Logo, and IBM Logo. If you are using one of these versions, do not define the IS procedure. Use PRINT instead, as in PRINT X > 5.]

```
TO IS  :WHAT
 PRINT  :WHAT
 PRINT  [ ]
END
```

Now the students could do things such as the following.

```
REDO
IS  X  >  5
TRUE

IS  X  <  10
TRUE
```

Monthly Musings — CONTINUED

```
IS  X  <  7
FALSE

IS  X  <  9
TRUE

IS  X  =  8
TRUE

PRINT  X
8

REDO
X  =  8
FALSE
```

Although this example is nowhere near an artificial intelligence application, I hope that it illustrates how self modifying languages, such as Logo, can change the way procedures perform.

The same idea applies to the self-modification concept with HyperCard. In his widely-read book, *The Complete HyperCard Handbook*, Danny Goodman suggests that some HyperCard stacks might be designed so that they will "learn" the habits of their users and make appropriate adjustments in their performance. This has some significant implications where personalization of instruction or learning environments is an issue.

For example, in an article in the August 1987 issue of *Computer Currents*, George Por talks about the potentials of HyperCard-like programs for such educational applications. "Some of those potentials are a fully personalized educational system in which learners are more in charge of the pace, style, and direction of their learning; and the development of collaborative learning/teaching methods that provide both team rewards and individual accountability."

This certainly reminds one of some similar properties of Logo learning environments, doesn't it?

To be sure, HyperCard could be used for computer assisted instruction, drill and practice, online quizzes, and the like. This may happen, since HyperCard does not come with a suggested philosophy of application such as Papert provided Logo users in *Mindstorms*. But the potential is there for individualized learning environments with the learners in charge. The question is, "How to exploit it?"

In his article, Por holds up a challenge to educators. "The new [hyper] media can provide a supportive technological environment for educational innovation, but it won't fulfill its promises without teachers and learners dedicated to exploring its potentials and pushing its limits."

But, with all this, let's remember that we can also apply in our Logo environments what we learn from HyperCard. For example, if you were at the Great Lakes/East Coast Logo Conference in Cleveland May 5 - 7, 1988, you may have seen Andy David's Saturday session entitled, "Hypertext: A LogoWriter-Based Environment for Non-Linear Reading and Writing." That's right, folks. Andy has devised a way to create a hypertext environment from the LogoWriter's screen pages. His project is being field-tested in Chicago classrooms.

In the past three columns, I have introduced HyperCard and pointed out several similarities it shares with Logo. I am not suggesting that HyperCard is a replacement or substitute for Logo. But I believe that HyperCard's potential is worth exploring and learning from. We are in a position to influence constructively the direction of development of worthwhile HyperCard applications in education, based on Logo-related teaching philosophies, and, in turn, be influenced by HyperCard itself. Shall we give it a go?

FD 100!

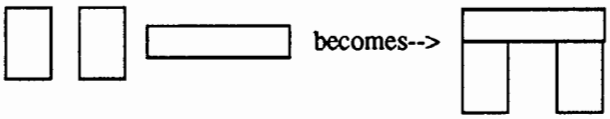
Tom Lough
PO Box 5341
Charlottesville, VA 22905

Bottom Up Programming

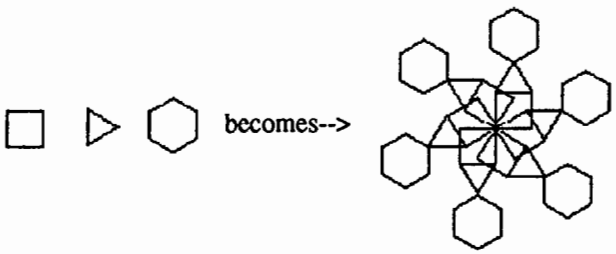
by David Moursund
and
Sharon Burrowes Yoder

Much of the writing aimed at Logo teachers talks about discovery learning. These discussions are often based on ideas put forth by Papert in *Mindstorms*. Teachers are encouraged to let students explore on their own; not to impose particular assignments to be completed. At the heart of this idea is the concept of *Bottom Up Programming*. Programming in the bottom up style involves working from the parts to the whole. The classic "draw a house" example is often approached in a bottom up manner. A student first learns to draw a square and then later learns to draw a triangle (as well as other geometric shapes).

Throughout the learning process, the student is encouraged to mess around with shape-creating procedures, using one or more of of these shapes to produce interesting combinations. One student may report, "I made an arch using two squares and a rectangle as the top."



Another may report "I made an interesting design using a triangle, square and hexagon."



Note that the important concept here is that the *student* who drew the house sees that a square and a triangle go together to make a house rather than being told to draw a house. But not all students will spontaneously make such discoveries.

Teaching Bottom Up Programming

How then does the teacher encourage students to explore using bottom up techniques? Initially, the student may need models such as those shown above. Bulletin boards of samples of other student's work could serve as a guide for those just beginning such open ended work. If a computer system appropriate for demonstration is available, a teacher could demonstrate (with student input) a variety of explorations

Brainstorming is a technique used in many other contexts, and it is especially valuable here.. The teacher can present students with components, such as the shapes used in the examples above, and ask students what they can think of that could be made with these shapes. Here, working with a group of students is important. Ideas generated by one student will trigger ideas from others. It is important to value every student's contribution to the discussion. There should be no "right" answers, and little or no discussion of any student's ideas. With a variety of ideas in hand, students can then move to the computer to explore on their own.

Assignments to Mess About

The assignment to mess around and see what you can create can be given at any place in a student's Logo development. And, it can always be accompanied by a suggestion that students brainstorm by themselves before going to the computer. (This is particularly appropriate when computer access is quite limited.) For example, given mastery of the fundamental turtle graphics commands, the HOUSE procedure given below, and the REPEAT command, a student can easily produce a variety of products.

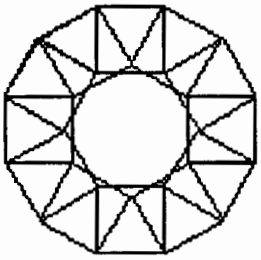
```
TO HOUSE
 SQUARE
 MOVE.TO.ROOF
 TRIANGLE
END

TO SQUARE
 REPEAT 4 [FORWARD 50 RIGHT 90]
END

TO MOVE.TO.ROOF
 FORWARD 50
 RIGHT 30
END
```

Bottom Up Programming — CONTINUED

```
TO TRIANGLE
 REPEAT 3 [FORWARD 50 RIGHT 120]
END
```

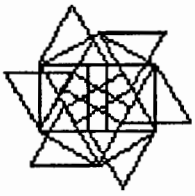
Below is the kind of exploration that you might observe in a Logo classroom where students have learned to use these procedures and have been encouraged to use them creatively.

```
REPEAT 12 [HOUSE]
```



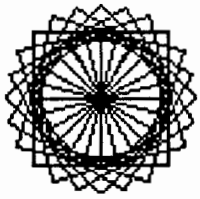
Hmmmm...this looks a bit like a wreath. Perhaps I could draw a holiday scene.

```
REPEAT 6 [HOUSE RIGHT 30]
```



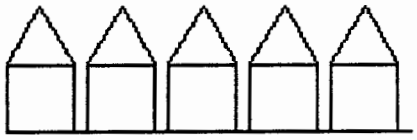
Now I have a pinwheel. Perhaps I could make a group of pinwheels of different colors.

```
REPEAT 24 [HOUSE LEFT 15 BACK 25]
```



What an attractive design! I could use it for the blossom of a flower in a flower garden.

```
REPEAT 5 [HOUSE LEFT 30 BACK 25 RIGHT 90
          FORWARD 30 LEFT 90]
```



Ah! A street full of houses...now I need trees and cars and maybe even some clouds in the sky.

Note that the kind of results described above can result from two categories of explorations:

1. The student is merely experimenting with combinations of commands, much as a composer might experiment with notes and phrases to produce a pleasing whole. The student has no specific goal in mind. Whenever a pleasing pattern is produced, the student takes note of the result and perhaps builds further on the result.

2. The student is trying to reach a particular goal ("I wanted to make a flower.") and initial progress or a bug produces something interesting that leads to a project different from the original plan. ("My attempts to make a flower produced something that reminded me of a Christmas wreath so I decided to make a Christmas wreath instead. See how nice it looks!")

Bottom up programming and Creativity

Bottom up programming often results in a lot of messing about — much as a child in art class might mix paints in an experimental manner to see what new colors emerge or try various pieces in a collage with no particular goal in mind at the outset. Students sometimes will experiment for a long time with a new idea. Just as the HOUSE procedure above was turned into a variety of designs leading to new ideas, so working with even the simplest building-block patterns may result in a myriad of interesting projects.

This kind of exploration can be viewed as encouraging creativity. Making new connections is encouraged and valued. Fluency (the number of different ideas produced) and flexibility (the variety of ideas produced) are both part of the bottom up process. In fact, bottom up programming can be *used* as a way to encourage and teach creativity.

What can be said about bottom up programming and problem solving? In the case of open ended exploration, the student

Bottom Up Programming — CONTINUED

needs to explicitly understand that the goal is to create an interesting whole made up of parts. In the case of an abandoned goal, the student should be helped to see that s/he explicitly stopped working towards the original goal in favor of a new goal. Both of these approaches encourage use of the creative process to put together pieces in unexpected ways.

A caution is in order here, however. Some students may always choose to abandon their stated goal when it is not easily reached. While there is a place for such decisions, especially when the student is particularly bothered by failure, there is also a place for establishing guidelines to be reached and achieving them.

Deciding when and how to honor the bottom up approach to Logo programming can be somewhat complex. Some students' learning style lends itself quite well to the bottom up approach. Others have structured styles of thinking that make the bottom up approach difficult for them to use. Still others may use the bottom up approach to avoid failure. Clearly dealing with these issues requires the attention of a caring and thoughful teacher. Making students explicitly aware of the advantages of building from the parts to the whole will help the students *themselves* make better decisions regarding the problem solving style that they should use for a particular project.

Transfer Examples

As indicated in the previous article in this series, learning to use a particular problem solving skill only in Logo is not enough. Explicit effort has to be made to help student to see how each new problem solving technique applies in other ares of their lives. Below is a list of examples that illustrate possible uses of a bottom up style problem solving in a variety of other areas.

1. You're hungry...you go to the refrigerator to see what is there. Your meal is based on what you find (rather than some pre-planned menu.)

2. In art class, a wide variety of materials are made available for a project. Some students will take a bottom up approach and let the materials they see dictate the content of their project.

3. A research project is assigned and a student picks a topic. Along the way, an interesting article leads towards a different topic entirely.

4. You want to listen to some music at a friend's house. Your choice is dictated by the tapes that the friend has in his/her library.

5. You are presented with a sentence in a foreign language that you know moderately well. You may begin the translation by examining the words you know and guessing at the rest from context.

6. You are in charge of planning the choreography for a dance. Your steps are limited by what the members of the group that will perform your dance can do.

7. You have a game plan in mind for winning a game. In the midst of your assault on your opponent, your opponent's key player is injured and can no longer play. You immediately devise a new strategy to take advantage of weaknesses of the substitute player.

8. You're planning to create a last-minute gift. Your final product is limited by the scraps of fabric or wood that are available.

9. You are shopping for a birthday present for a friend. You had in mind buying a sweater. As you walk down the isle toward the sweater section of the store you see that shirts are on a half price sale. You decide to buy a shirt.

10. There's a jigsaw puzzle on the table to put together, but the box is missing.

Note that many of the above examples could indeed be viewed in light of the creative process. Non-linear or unusual connections need to be made to solve many of these problems. Any place that you as a teacher are working towards the development of the creative process may indeed be a place where a connection with bottom up programming can be made.

[Dave Moursund and Sharon Burrowes Yoder, ICCE, University of Oregon, 1787 Agate St., Eugene, OR 97403-9905].

Logo LinX

to the Left
Sunny Side Up
Judi Harris

"April Comes like an idiot, babbling and strewing flowers."
——Edna St. Vincent Millay, 1921

In honor of All Fools' Day (albeit a few months late), I would like to suggest some Logo foolery. WHAT? No fools are allowed in your classroom? No foolishness is prescribed in your curriculum guide? Your lesson plans are foolproof? It may be a foolhardy move, but may I humbly suggest:

"To never see a fool you lock yourself in your room and smash the looking-glass."
——Carl Sandburg, 1936

Slanted View

What is the difference between these two electronic mail messages?

> Subject: Project Grading
> From: Glen
> Date: April 1, 1988
> To: Judi
>
> Judi,
>
> About the assignments to grade: Let's toss 'em down the stairs & the ones that get nearest the **bottom** get the highest grades. (You did say grading was subjective.)
>
> Glen

> Subject: Project Grading
> From: Glen
> Date: April 1, 1988
> To: Judi
>
> Judi,
>
> About the assignments to grade: Let's toss 'em down the stairs & the ones that get nearest the **bottom** get the highest grades. (You did say grading was subjective.) ;-)
>
> Glen

After reading the first message, you may have thought that this professor was somewhat unfair in his evaluation practices. But after reading the second version, if your view was properly slanted, you would probably conclude that he was temporarily under the influence of Poe's Imp of the Perverse. The discerning factor, of course, was the strange symbol at the end of his suggestion. Still don't see it? Just tilt your head to the left until you see a familiar face.

Face It

Funny faces, made of text characters, and seen when looking sideways, can punctuate sentences with humorous emotion. These symbol collections are also called "emoticons," or

"....[figures] created with the symbols on a keyboard that [are] read with the head tilted to the left. Used to convey the spirit in which a line of text was typed."
——CompuServe data library

Emoticons are typically used in informal typed correspondence, such as electronic mail. They can also add graphic dimensions to word-processed stories and poems. And, as you may have guessed, they provide a creative Logo list processing challenge.

We can define procedures that output the characters of emoticons as lists.

```
TO EEK!
 OUTPUT [<:-O]
END

TO BATMAN
 OUTPUT [B^)]
END
```

These procedures can then be used in Logo stories.

```
TO CLIMAX
 PRINT SENTENCE [IT WAS THE MONSTER!] EEK!
 PRINT (SENTENCE [BATMAN] BATMAN [TO THE
   RESCUE!]
END
```

CLIMAX
```
IT WAS THE MONSTER!  <:-O
BATMAN  B^)  TO THE RESCUE!
```

A Fool's Paradise

Emoticon variations can reflect many different humorous themes. Here are just a few that I've collected from electronic correspondence.

:-)	humorous	:/)	not humorous
:-D	smile!	:-P	bleah!
:-*	oops! (covering mouth with hand)		