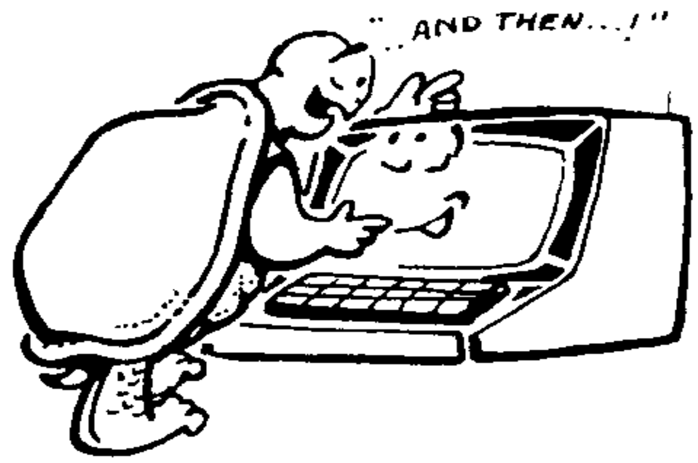
Chapter 11. Talk To Your Computer

One of the more interesting things about the Logo language is list processing. Early computer languages were known as "number crunchers." Everything was represented by numbers. Logo is different. It uses "symbolic computation" that allows you to process ideas.

Did you ever talk to your computer?



Here's a short and simple procedure. Type it and then run it. What happens? It's like a short conversation, isn't it? But, no, you're not really talking to the computer.

```
TO TALK
PRINT [HI! WHAT'S YOUR NAME?]
MAKE “NAME READWORD
PRINT SENTENCE [I DON'T THINK WE'VE MET,]
      :NAME
PRINT [HAVE YOU EVER TALKED TO A
      COMPUTER?]
TEST RC = “N
IFFALSE PRINT [WOW! DO YOU TALK TO
      TURTLES, TOO?]
IFTRUE PRINT [OH BOY!  A BEGINNER.]
END
```

You’ve read a little bit about Logo characters, words, numbers, and lists before. But there is more to learn, especially when you’re working with list processing. So let’s make sure we know the differences before we get down to business.

Characters, Words, and Lists

“What do you mean by LIST processing?”

“One of the really neat things about Logo is that it allows you to process information, or data, in many different forms. It can be numbers, words, lists, property lists, or arrays.”

“Arrays?”

“Arrays are like tables…information arranged in rows and columns. But don’t get confused by them. You see arrays everyday. A calendar is an example of an array. However, we’re not going to worry about arrays in this book. Maybe the next one.

“But you do need to pay attention to the type of information…the type of data…you’re working with. You have to identify it properly so that Logo knows what you’re talking about.

“You’ll see as we go along.”

Numbers

Numbers can consist of one or more integers. Integers are “whole” numbers such as 3, 25, 423, or 1,324,598. Fractions and decimals are not whole numbers. They are parts of numbers.

```
FORWARD 5          SETH 45
REPEAT 4           SETH 0
RIGHT 20           SETH -90
BACK 30            SETH -270
```

There are positive numbers…they have a plus (+) sign in front of them. However, to keep life simple, we leave that + sign off. We only label negative numbers…those that are less than 0.

Decimal numbers include a part of a whole number, such as 1.25, 3.24, 89.23. MSW Logo lets you use decimals…

FD 100.125 BK 21.75

Logo writes very big and very small numbers using what they call engineering notation, such as 1.0E-2. This is too confusing for this book. But if you see something like that, at least you'll know the computer has not gone crazy.

Characters

Logo also displays alphabetic and numeric characters, using the CHAR or ASCII (American Standard Code for Information Interchange) primitives.

SHOW CHAR 65
  displays the letter A.
SHOW CHAR 67
  displays the letter C.
SHOW CHAR 49
  displays the number 1.
SHOW CHAR 32
  displays a space. (Yes, there's a code for a space.)

CHAR <code number from 0 to 127> displays the letter, number, or punctuation mark for the specified ASCII code. No…you don't have to know ASCII code.

If you ever want to find out the ASCII code for a something, type…

SHOW ASCII <character>

SHOW ASCII “A
displays 65.
SHOW ASCII “a
displays 97.

Try a few ASCII and CHAR commands so that you get a good feel for what they do. Just remember that the traditional ASCII code consists of 128 characters numbered from 0 to 127.

Words

Notice that when you identify a letter, you must use a quotation mark in front of it. You don’t have to do that with numbers. Logo treats 3 as a number just as it treats 325491587 as a number.

It may seem strange but Logo treats a single letter as a word. It treats...

PRINT "T the same as

PRINT "Tyranosaurus

Of course, it isn’t really strange at all once you think about it.

“A” is a word, isn’t it? It’s one of the three articles in the English language. The other two are “an” and “the.”

What about the word “I?”

A Logo word is any combination of letters, numbers, and punctuation marks…with no spaces.

```
SHOW "ABC_456
  displays ABC_456.

SHOW "A93HK8
  displays A93HK8.
```

Lists

Lists are elements within square brackets…[ ]. A List can include words, numbers, or other lists. Of course, if a List contains a list, both have to be enclosed in brackets.

```
PRINT [This is a list.]
  displays
         This is a list.

PRINT [1 2 3 this is also a list_A B C?]
  displays
         1 2 3 this is also a list_A B C?
```

What about this?

```
PRINT [1 2 3 [this is also a list_A B C]]
```

Here's a list within a list that displays as…

```
1 2 3 [this is also a list_A B C]
```

Take a look at the TALK procedure. There's really only one thing new there. Do you see how SENTENCE works? SENTENCE, or SE, takes two inputs and prints them together. These can be two words, two lists of words, or a combination of words and lists.

```
PRINT SE [I DON'T THINK WE'VE MET,] :NAME
```

I DON'T THINK WE'VE MET, is a list. This is the first input. :NAME is the second. This is the word you typed when the procedure asked you for your name.

SPECIAL NOTE: When you want to add more than two inputs to a Sentence, use parentheses. For example…

```
PR (SE :NAME “, [HOW ARE YOU?])
```

Let’s say that :NAME is the variable with the value of Ernestine. This line would result in…

```
ERNESTINE , HOW ARE YOU?
```

While it may seem strange to you, Logo treats the comma as a separate word. Anything that follows a single quotation mark is a word. The word ends with a space.

You can include numbers, characters, words, and lists in a Logo Sentence.

```
PR (SE :NAME “is CHAR 49 CHAR 52 [years old!])
```

This results in…

```
ERNESTINE is 14 years old!
```

Let's change the TALK procedure around a bit so we can see more of what the SENTENCE command can do.

```
TO TALK
PRINT [HI! WHAT'S YOUR NAME?]
MAKE “NAME WORD READWORD “,
MAKE “REPLY SE :NAME [MY NAME IS
        ERNESTINE.]
PRINT SE [I DON'T THINK WE'VE MET,] :REPLY
PRINT [HAVE YOU EVER TALKED TO A
        COMPUTER?]
TEST READWORD = 'NO
IFTRUE PRINT [WOW! DO YOU TALK TO
        TURTLES, TOO?]
IFFALSE PRINT [OH BOY!  A BEGINNER.]
END
```

Hey…there's another command in there now…WORD. We talked about Words before. Do you see the difference between SENTENCE and WORD?

Look closely. SENTENCE combines words, numbers, characters, and lists. WORD only combines characters, numbers, and other words. SENTENCE produces a list. WORD produces another word. To see the difference, try something like this…

```
PRINT WORD “A “B
PRINT SENTENCE “A “B
```

or this…

```
PRINT WORD 1 2
PRINT SE 1 2
```

Hmmmm…this is getting interesting. What would happen if we did this…

```
MAKE “A WORD 1 2
MAKE “B WORD 3 4
PRINT :A + :B
```

Try this…

```
PRINT WORD :A :B
```

What happened? Bet you got 1234, right? So let's do some testing to see what we've got here.

```
PRINT :A
IF NUMBERP :A [PRINT “TRUE]
```

Remember NUMBERP! Some other versions of Logo, including PC Logo and MicroWorlds, use a ? rather than a P at the end of NUMBER, EMPTY, MEMBER, and the like. If you want to review, go back to the end of chapter 8.

But let’s get back to NUMBERP.

Hmmmm? This tells Logo that IF :A is a number, print TRUE. Since nothing happened, I guess you can say that :A is a word and not a number. Another way to say this, using another new primitive…

```
PRINT :A
IF NOT NUMBERP :A [PRINT “FALSE]
```

This time, Logo prints FALSE. IF :A is NOT a number, print FALSE. In this situation, Logo sees the numbers as words and not as numbers. Seems weird, doesn't it.

Before we leave this confusion, let's mess it up some more.

What would this line print?

```
PRINT (WORD 1 CHAR 32 2 CHAR 32 3)
```

The result would look something like a Sentence but would actually be a Word.

```
1 2 3
```

This actually reads as 1 space 2 space 3, because the spaces are deliberately inserted as characters. In Logo, that's a Word.

Just to check the difference, try it this way…

```
PRINT (WORD 1 2 3)
```

Where'd the spaces go?

Labeling Statements

The command, LABEL, means different things in different versions of Logo. In MSW Logo, Label's input, which may be a word or a list, is printed on the screen.

To define the font in which the word or list is displayed, use the SETTEXTFONT command. A font determines what text look like on the screen when using the command Label.

```
SETTEXTFONT [font]
```

The input to SETTEXTFONT is a list that completely describes a font. The list contains…

[[*Font name*] *Height Weight Italic Underline StrikeOut*]

[***Font name***] **Specifies the typeface name of the font. You can use any font available on your computer. Note, if you misspell the font name, MSW Logo will list what fonts are available.**

Height **An integer that specifies the desired height for the font. If this value is greater than zero, it specifies the cell height of the font. If it is less than zero, it specifies the character height of the font.**

Weight **An integer that specifies the font weight. This member ranges from 0 to 900 in steps of 100. A value of 0 means use default weight.**

Italic **An integer that specifies an italic font. Use zero (0) to select regular type.**

Underline **An integer that specifies an underlined font. Use zero (0) to select regular type.**

StrikeOut **An integer that specifies a strikeout font if nonzero. Use zero (0) to select regular type.**

Here's an example…

```
SETTEXTFONT [[HELV] 40 900 1 0 0]
REPEAT 6 [LABEL "HELLO]
```

Now try a bunch of your own Labels.

Logo Postcards

Has your family ever been on a vacation and sent lots of postcards to friends back home? Those cards all seem the same, don't they…

Dear ______,

Here we are in wonderful ______. We're having a great time ________ and _______. The weather is very _______. We'll be home _________. See you then.

Love,

____

Wouldn't it be nice to have a procedure to print your cards for you.

```
TO SETUP
PRINT [WHO IS THIS CARD FOR?]
MAKE “FRIEND READWORD
PRINT [WHERE ARE YOU WRITING FROM?]
MAKE “VAC READWORD
MAKE “VAC WORD :VAC “.
PRINT [WHAT ARE YOU DOING?]
MAKE “ACT1 READWORD
PRINT [WHAT ELSE?]
MAKE “ACT2 READWORD
MAKE “ACT2 WORD :ACT2 “.
PRINT [HOW'S THE WEATHER?]
MAKE “WEA READWORD
MAKE “WEA WORD :WEA “.
PRINT [WHEN WILL YOU BE HOME?]
MAKE “ARR READWORD
MAKE “ARR WORD :ARR “.
```

```
PRINT [HOW WILL YOU SIGN THE CARD?
MAKE “SIG READWORD
END
```

Now we'll take this information and print it on the postcards.

```
TO POSTCARD
PRINT SE [DEAR] :FRIEND
PRINT “
TYPE SE [HERE WE ARE IN WONDERFUL] :VAC
    PRINT [WE’RE HAVING]
TYPE SE [A GREAT TIME] :ACT1 TYPE SE “AND
    :ACT2 PRINT “THE
TYPE SE [WEATHER IS] :WEA PRINT SE [WE'LL BE
    HOME] :ARR
PRINT [SEE YOU THEN.]
PRINT “
PRINT [LOVE,]
PRINT “
PRINT :SIG
END
```

After running this procedure, I bet you already know the difference between PRINT and TYPE. If not, go back and look at it again.

Making Headlines

Look at any book on Logo and usually you'll find a procedure that picks parts of speech and combines these into sentences; something like this…

```
TO HEADLINES
MAKELISTS
PRINTHEADLINES
END

TO MAKELISTS
MAKE “ADJ [HAPPY GLAD ANXIOUS GOOD
        SINCERE]
MAKE “NOUN [SANTA PEOPLE FRIENDS
        PARENTS]
MAKE “VERB [GAVE RECEIVED SPREAD SHARED
        WISHED]
MAKE “OBJ [GIFTS LOVE CHEER GLADNESS JOY
        HAPPINESS]
END

TO PRINTHEADLINES
PR (SE PICK :ADJ PICK :NOUN PICK :VERB PICK
        :OBJ)
WAIT 10
PRINTHEADLINES
END

TO PICK :WORDS
OUTPUT ITEM (1 + RANDOM (COUNT :WORDS))
        :WORDS
END
```

When you enter HEADLINES, the first thing that Logo does is make up lists of words: ADJ, NOUN, VERB, and OBJ. Then the next procedure is called.

PRINTHEADLINES tells the computer to PRint a SEntence…but what sentence?

What do you think the PICK procedure does?

Remember…to understand any Logo line, you start at the left and move to the right, one command at a time. OUTPUT takes one input. That input is ITEM.

Well, ITEM takes two inputs: a number and something else…like the 4th list, the 2nd word. The next thing you see after ITEM is in parenthesis. That means that…

```
(1 + RANDOM (COUNT :WORDS))
```

is all one element. But what about all those parentheses?

The easiest way to read those is to start with the parentheses on the inside…

```
COUNT :WORDS
```

So…if I start with PICK :ADJ, then this line becomes…

```
COUNT :ADJ
```

How many words are inside the brackets in the MAKELISTS procedure? There are five adjectives, right? So…if we COUNT the ADJectives, we have 5. Now we move to the next set of parenthesis. Now we have…

```
(1 + RANDOM 5)
```

RANDOM 5 by itself would give us the five numbers 0, 1, 2, 3, and 4. Since we don't want Logo to use the '0, we tell Logo to add 1 to RANDOM 5. In this way, we're working with five whole numbers, 1, 2, 3, 4, and 5.

So now we have…

```
OUTPUT ITEM (1 + RANDOM (COUNT :WORDS))
:WORDS
```

or

OUTPUT the 1st word in the ADJ list…or the 2nd, 3rd, 4th or 5th.

Now this line begins to make a bit more sense…

```
PR (SE PICK :ADJ PICK :NOUN PICK :VERB
PICK :OBJ)
```

Logo is going to print a random ADJ, a random NOUN, a random VERB, and a random :OBJ. You'll get sentences like this…

```
HAPPY SANTA GAVE CHEER
GLAD PEOPLE RECEIVED GIFTS
GOOD PEOPLE SHARED JOY
ANXIOUS PARENTS WISHED JOY
HAPPY PARENTS SPREAD LOVE
```

How can MSW Logo handle this?

```
TO HEADLINES
TYPE SE ADJ NOUN PRINT SE VERB OBJ
WAIT 100 HEADLINES
END
```

This procedures tells Logo to TYPE a SEntence using the output of ADJ and NOUN. Then, on the same line, PRINT a SEntence using VERB and OBJ.

In the parts of speech procedures, Logo does things a bit differently. We're going to use numbers here to keep things simple.

```
TO ADJ
LOCAL “ANS
MAKE “ANS RANDOM 10
IF :ANS = 0 [MAKE “ADJ “HAPPY]
IF :ANS = 1 [MAKE “ADJ “GLAD]
IF :ANS = 2 [MAKE “ADJ “ANXIOUS]
IF :ANS = 3 [MAKE “ADJ “GOOD]
IF :ANS = 4 [MAKE “ADJ “GLAD]
IF :ANS = 5 [MAKE “ADJ “BEAUTIFUL]
IF :ANS = 6 [MAKE “ADJ “GREAT]
IF :ANS = 7 [MAKE “ADJ “SAD]
IF :ANS = 8 [MAKE “ADJ “TENDER]
IF :ANS = 9 [MAKE “ADJ “FUNNY]
OUTPUT :ADJ
END
```

In the ADJ procedure, Logo changes the local variable, ANS, into an adjective. Here, she uses quotes to mark her ADJ words. But she does things differently in the NOUN procedure. Why?

```
TO NOUN
LOCAL “ANS
MAKE “ANS RANDOM 6
IF :ANS = 0 [MAKE “NOUN [SANTA ]]
IF :ANS = 1 [MAKE “NOUN [PARENTS ]]
IF :ANS = 2 [MAKE “NOUN [PEOPLE ]]
IF :ANS = 3 [MAKE “NOUN [KIDS ]]
IF :ANS = 4 [MAKE “NOUN [FRIENDS ]]
IF :ANS = 5 [MAKE “NOUN [NEIGHBORS ]]
OUTPUT :NOUN
END
```

Logo wants to print headlines correctly. The SEntence command will leave a space between the ADJective and the NOUN, but it won't leave a space after the noun unless we tell it how to do that.

Here we use a list. You might think that's just one word inside the brackets, but Logo sees that as a word and a space. That's a little different.

```
TO VERB
LOCAL “ANS
MAKE “ANS INTEGER RANDOM 5
IF :ANS = 0 [MAKE “VERB “GAVE]
IF :ANS = 1 [MAKE “VERB “RECEIVED]
IF :ANS = 2 [MAKE “VERB “SPREAD]
IF :ANS = 3 [MAKE “VERB “SHARED]
IF :ANS = 4 [MAKE “VERB “WISHED]
OUTPUT :VERB
END
```

In the OBJect procedure, we've asked Logo to do something else different. Sentences should have periods, right?

OK, maybe newspaper headlines don't always have periods in them. These headlines do so you can see another way to use the WORD command. Here Logo combines the word with the quotes with the period.

```
TO OBJ
LOCAL “ANS
MAKE “ANS RANDOM 6
IF :ANS = 0 [MAKE “OBJ WORD “GIFTS “.]
IF :ANS = 1 [MAKE “OBJ WORD “LOVE “.]
IF :ANS = 2 [MAKE “OBJ WORD “CHEER “.]
IF :ANS = 3 [MAKE “OBJ WORD “JOY “.]
IF :ANS = 4 [MAKE “OBJ WORD “PEACE “.]
IF :ANS = 5 [MAKE “OBJ WORD “GLADNESS “.]
IF :ANS = 6 [MAKE “OBJ WORD “TIDINGS “.]
IF :ANS = 7 [MAKE “OBJ WORD “HAPPINESS “.]
OUTPUT :OBJ
END
```

Now Logo can print sentences just like the other types of word processing software. Only Logo's headlines have periods.

```
HAPPY SANTA GAVE CHEER.
```

Before we finish with HEADLINES, let's take another look. Four words like these really aren't very exciting. Can we dress up our HEADLINES a bit more?

```
TO HEADLINES
TYPE SE ADJ NOUN PRINT SE VERB OBJ
WAIT 100 HEADLINES
END
```

What if we changed that second line…

```
TYPE SE ADJ NOUN TYPE VERB PRINT SE ADJ
OBJ
```

That's OK…except we have the verb and the second adjective running together. Oh, well…what do you think? Is there a way to fix that?

What would you do? A hint…what about a space?

Word Games

Logo postcards and headlines may seem a bit silly. But what about a word game…like the game of Nim.

```
TO GAME
(TYPE [There are ] CHAR 32 :TOTAL CHAR 32) PR
[ stones in a pile.]
(TYPE [Take from 1 to ] CHAR 32 :PICK CHAR 32)
PR [ stones.]
MAKE “KEY READCHAR
IGNORE RC
IF OR :KEY > :PICK :KEY < 1 [CORRECT]
(TYPE [I take] CHAR 32) PR :PICKS - :KEY
MAKE “TOTAL :TOTAL - :PICKS
IF :TOTAL > 1 [GAME]
IF :TOTAL = 1 [STOP]
END

TO NIM
CT
REPEAT 5 [PR “]
PR [Welcome to the Game of NIM!]
PR “
PR [This is a game where you and the]
PR [computer take turns picking stones]
PR [from a pile.  The challenge is not]
PR [to get stuck with the last stone.]
PR “
PR [You decide. How many stones will]
PR [each player pick on each turn?]
MAKE “PICK READWORD
MAKE “PICKS :PICK + 1
PR [How many turns will we have to pick?]
MAKE “TURNS READWORD
MAKE “TOTAL :PICKS * :TURNS + 1
```

```
GAME
OVER
END

TO OVER
REPEAT 3 [PR "]
PR [There is one stone left. I win again.]
END

TO CORRECT
PR [Try again!]
MAKE "KEY RC
END
```

Why does Logo always win?

Can you figure out how the it does that?

If you go through the game step-by-step you'll find it isn't very hard. Then you can zap your friends when you play the game off the computer!

Now…here's a real challenge!

Can you change NIM so that the computer does not always win? It's tough…but it can be done without too many changes.

The Amazing Oracle

Here's a fun game to play on a group of friends or on a class. The Oracle thinks up a story. The object of the game is to figure out the story by asking the Oracle direct questions that can be answered by Yes or No.

Well…the Oracle is sneaky. Take a look at the procedures and see if you can figure out how it works.

```
TO ASKS
PR “
PR [What’s your question?]
MAKE “QUESTION READLIST
IF NOT (LAST LAST :QUESTION) = “? [PRINT
        [Questions must end with a question mark.] ASKS]
IF MEMBERP (LAST BL LAST :QUESTION) [a e i o u]
        [PR “Yes ASKS]
IF (LAST BL LAST :QUESTION) = “y [PR “Maybe
        ASKS]
PR “No
ASKS
END

TO ORACLE
CLEARTEXT TEXTSCREEN
PR [I'm thinking of a story.  Ask me direct questions,]
PR [those I can answer with Yes or No.]
PR [Then I’ll tell you what it is.]
ASKS
END
```

Oracle begins by asking you for a question.

```
MAKE “QUESTION READLIST
```

When you type a question, you are actually typing a list of words. This list becomes the variable :QUESTION. Logo then checks to see if you added a question mark.

```
IF NOT (LAST LAST :QUESTION) = “? [PRINT
        [Questions must end with a question mark.] ASKS]
```

What this says is that if the LAST character of the LAST word in the list is not a question mark, print the statement and return to the top of the ASKS procedure.

Time-out for a few experiments…

Try this one…

```
SHOW LAST “TEST
```

What do you think is going to be shown?

```
SHOW FIRST “TEST
```

What’s this going to show?

How about these?

```
SHOW BUTFIRST “TEST
SHOW BF [THIS IS A TEST.]
SHOW LAST BF [THIS IS A TEST.]
SHOW LAST FIRST BF [THIS IS A TEST.]
```

Try a few of your own. Using the commands of FIRST, LAST, BUTFIRST, and BUTLAST, you can pick any letter of any word in a list.

Back to the Oracle…

```
IF MEMBERP (LAST BL LAST :QUESTION) [a e i o u]
    [PR “Yes ASKS]
```

That makes sense because what this statement says is if the next to the last…last but last…character of the last word of the sentence is a member of the list [a e i o u], print Yes and then call the ASKS procedure again.

If the next to last character is not a member of the list, Logo looks at the next line.

```
IF (LAST BL LAST :QUESTION) = “y [PR “Maybe
    ASKS]
```

This time, Logo asks if the next to last character is ’y. If so, Logo prints Maybe and calls the ASKS procedure again.

If the next to last character is neither a vowel or ’y, Logo prints ’No and calls ASKS again.

```
PR “No
```

Now that you’ve had a healthy taste of list processing, take a look at the Data Structure Commands in the online Help file. This will give you an overview of the other list processing commands. You’ll be using more of them in the next exercise.

Word Sums

“What’s two plus two?”

“How easy can you get? It’s four, of course!”

“How’d you get the answer?”

“I added two plus two and got four. What do you think?”

“Did you add 2 + 2? Or two plus two? If you’re talking about Logo, there is a difference you know.”

“Well, I never really thought about that.”

“OK, let’s give it a whirl.”

Adding words together is a great exercise in list processing. Here’s a brief description of how it works.

```
WORDSUM “THREE “SIX
```

WORDSUM counts through a list to see where the words THREE and SIX are located. They are in the third and the sixth position. So, Logo then adds…

3 + 6 = 9

Finally, Logo counts to the ninth position in the list and prints that word as the answer.

Logo does one more thing with this procedure; that’s to determine if there if the number is a ’teen. In the problem above, it isn’t so Logo simply printed the answer. But let’s take a look at another example…

```
WORDSUM “SIX “EIGHT
```

The answer is 14. This time it went to the TEENS procedure and printed…

```
The answer is fourteen.
```

Here’s the full procedure. Let’s take it apart.

```
TO ADDNUMS
MAKE “NUMS1 SE :NUMS1 :NUMS
MAKE “NUMS3 (FIRST :NUMS)
REPEAT (COUNT :NUMS2) - 1~
      [MAKE “NUMS1 BF :NUMS1~
      IF (FIRST :NUMS1) = (FIRST :NUMS)~
      [MAKE “NUMS3 FIRST BF :NUMS]]
```

```
IFELSE :NUMS3 = “ZERO [TYPE [The answer is:\ ]
        PR (FIRST :NUMS1)][TEENS]
PR [Press any key to continue.]
END

TO INFO
CLEARTEXT TEXTSCREEN
PR [Have some fun adding numbers.]
PR “
PR [Only here we add words.]
PR [To run the procedure, type the word]
PR [< < WORDSUM > > followed by the]
PR [names of two digits...like this...]
PR “
PR [\ \ \ \ \ WORDSUM “FOUR “THREE]
END

TO INFO2
CLEARTEXT
PR [You can change the list “NUMS” by changing]
PR [languages...Spanish, for example.  Or change to]
PR [Binary or Hex number systems.]
PR [Most importantly...Have Fun Exploring!]
END

TO SET1 :NUM1 :NUMS
IF (FIRST :NUMS) = :NUM1 [OP :NUMS]
OP SET1 :NUM1 BF :NUMS
END

TO SET2 :NUM2 :NUMS
IF (LAST :NUMS) = :NUM2 [OP :NUMS]
OP SET2 :NUM2 BL :NUMS
END
```

```
TO SETUP
CT MAKE “NUMS [ZERO ONE TWO THREE FOUR
        FIVE SIX SEVEN EIGHT NINE]
END

TO TEENS
IF (FIRST :NUMS1) = “ZERO [TYPE [The answer is:\ ]
        PR “ten.]
IF (FIRST :NUMS1) = “ONE [TYPE [The answer is:\ ]
        PR “eleven.]
IF (FIRST :NUMS1) = “TWO [TYPE [The answer is:\ ]
        PR “twelve]
IF (FIRST :NUMS1) = “THREE [TYPE [The answer
        is:\ ] PR “thirteen.]
IF (FIRST :NUMS1) = “FOUR [TYPE [The answer is:\ ]
        PR “fourteen.]
IF (FIRST :NUMS1) = “FIVE [TYPE [The answer is:\ ]
        PR “fifteen.]
IF (FIRST :NUMS1) = “SIX [TYPE [The answer is:\ ]
        PR “sixteen.]
IF (FIRST :NUMS1) = “SEVEN [TYPE [The answer
        is:\ ] PR “seventeen.]
IF (FIRST :NUMS1) = “EIGHT [TYPE [The answer
        is:\ ] PR “eighteen.]
END

TO WORDSUM :NUM1 :NUM2
SETUP
MAKE “NUMS1 SET1 :NUM1 :NUMS
MAKE “NUMS2 SET2 :NUM2 :NUMS
ADDNUMS
IGNORE RC
INFO2
END
```

The first question, as you start to look at WORDSUM, is what do :NUMS1 and :NUMS2 equal.

Turn on TRACE and then run the procedure again. This allows you to follow the recursive calls of SET1 and SET2. Then you go to ADDNUMS.

```
TO ADDNUMS
MAKE “NUMS1 SE :NUMS1 :NUMS
MAKE “NUMS3 (FIRST :NUMS)
REPEAT (COUNT :NUMS2) - 1~
        [MAKE “NUMS1 BF :NUMS1~
        IF (FIRST :NUMS1) = (FIRST :NUMS)~
        [MAKE “NUMS3 FIRST BF :NUMS]]
IFELSE :NUMS3 = “ZERO [TYPE [The answer is:\ ]
        PR (FIRST :NUMS1)][TEENS]
PR [Press any key to continue.]
END
```

The first two lines are simple enough. (:NUMS3 is a number used to determine if the answer is ten or above.) The third is where things get a little complicated. But you know all those commands. So just start from the right and move through the line, one command at a time.

```
REPEAT (COUNT :NUMS2) - 1
```

Let’s say you typed WORDSUM “EIGHT “SIX. What would :NUMS2 be?

```
[ZERO ONE TWO THREE FOUR FIVE SIX]
```

So…COUNTS :NUMS2 - 1 is what? I get six, what do you get? So the line starts with REPEAT 6. Now what?

The next command is…

```
MAKE “NUMS1 BF :NUMS1
```

First of all, what’s :NUMS1?

```
EIGHT NINE ZERO ONE TWO THREE FOUR FIVE
    SIX SEVEN EIGHT NINE
```

What is BF :NUMS1? Remember…BF is BUTFIRST. So BF :NUMS1 is the list above without the first element, or…

```
NINE ZERO ONE TWO THREE FOUR FIVE
    SIX SEVEN EIGHT NINE
```

Next, you come to…

IF (FIRST :NUMS1)…that’s now NINE, right?

```
IF (FIRST :NUMS1) = (FIRST :NUMS)~
    [MAKE “NUMS3 FIRST BF :NUMS]
```

:NUMS is defined in the SETUP procedure. What’s FIRST :NUMS? Does it equal FIRST :NUMS1? No…so the procedure repeats again.

On the sixth repeat, what is FIRST :NUMS1? Four…right?

What has :NUMS3 become? It became ONE on the third repeat cycle. Since :NUMS3 does not equal ZERO, the TEENS procedure is called.

Awfully simple? Or simply awful?

"Wow! What can't you do in Logo?"

"Well, Morf, the most important lesson is "Never say never." You and I aren't experts, but have you ever found anything you can't do if you put your mind to it?

Another thing you've got to realize is that we have just scratched the surface of what you can do with MSW Logo. There is so much, much more you can do."

"OK, we're wasting time! What's next?"

MY JOURNAL
OR HOW I DID IT!