Lisp: Basically Speaking—Part I

by Randy Beer

Interested in a language that uses objects instead of numbers? Lisp is a symbol-manipulation language that uses lists of objects.

Lisp is a programming language, usually considered to fall somewhere between machine language and higher-level languages such as Basic, Pascal, APL, or Fortran. Its syntax and data structures differ from more traditional languages. Much of today's research in symbolic math systems, natural language interfaces, and artificial intelligence is being done in Lisp, or in a higher-level language based on Lisp.

However, writing a program in Lisp no more guarantees that it will be intelligent than having a truckload of materials guarantees that you will be able to build a house. The basic building blocks appear to be there, but more work is necessary to even begin to assemble them into programs that exhibit intelligent behavior.

Perhaps because of its association with such abstract things as artificial intelligence, a stigma of complexity has been associated with Lisp. People who have seen a Lisp program without understanding it remember only the seemingly confusing syntax and endless parentheses. These things that tend to confuse the uninitiated are what makes Lisp powerful in the hands of an experienced programmer.

Understanding Lisp

Lisp is a symbol-manipulation language. Where many languages work with numbers, Lisp works with objects such as "chair" and "block." Relations between objects are represented as lists; hence, it is a list processor (from which Lisp gets its name). An example of a relationship between a chair and a block would be shown as: (ON BLOCK CHAIR).

These words or objects are called atoms. Numbers are also atoms. Symbolic atoms, however, cannot begin with a number, but can contain one. Thus FACT, ARG1, ONE, 12 and −3.14159 are all atoms; FACT, ARG1, and ONE are symbolic atoms; and 12 and −3.14159 are numbers. Two special atoms come predefined in every Lisp system; they are the atoms T and NIL, and can usually be thought of as logical true and false, respectively.

Lists are built out of atoms and other lists, with a left parenthesis to mark the beginning of a list and a right parenthesis to mark the end. (A B C), (MUL 2 3), (A (B (C D) E) F G), and () are all examples of lists.

The atom NIL serves a dual purpose in that it is also used to represent the empty list. NIL and () are equivalent in all respects.

Lisp works with symbolic or s-expressions composed of atoms and lists. Thus, anything that's an atom or a list is also an s-expression. In the eyes of a Lisp interpreter, programs and data are nearly identical. This fact contributes greatly to the power of Lisp, because it allows one program to write

User input is next to the $ Basic Lisp prompt, with interpreter response on the next line, as in all examples in this article. Note that all messages from the interpreter are preceded by a semi-colon. The OB LIST is where new atoms are stored.

```
$ (MUL2)
; MUL2 INVALID FUNCTION NAME

$ (%)
; MUL2 DELETED FROM OB LIST

$ (MUL 2 2)
4
```

Fig. 1. Typing Error Correction

```
$ (ADD 1 2.51 −3)
.51

$ (SUB 4 2)
2

$ (MUL 4 3 2 1)
24

$ (DIV 22 7)
3.14286

$ (POWER 2 3)
8

$ (MUL (ADD 2 3) (POWER 2 .5))
7.07107
```

Fig. 2. Arithmetic in Basic Lisp

The Key Box

Model I, II/16, and III
16K RAM, Cassette Basic
32K RAM, Disk Basic
64K RAM, Model II/16

another program and then execute it.

Using Basic Lisp

It has been proven time and again that the best way to teach almost anything is to let the student get his hands dirty from the beginning. A Lisp interpreter written in Basic is included in Listing 1. Type in the program and try all of the examples given in this series and any other ideas you may have. Though it may seem contradictory to write an interpreter for such a symbolic, recursive language in Basic, it may help make Lisp available to as many people as possible.

Program Listing

```
5 REM     * BASIC LISP VER 1.1 *
10 REM * BY RANDY BEER; AUG., 1981 *
15 CLS:CLEAR325:DEFINTA-E,G-V,X-Z:DEFSTRO:DIMLM(1100),PL(1100),O
B(90),PT(90),ST(350),FP(50),T1(15),X1(15):N=3000
22 PRINTTAB(23)"BASIC LISP VER 1.1":PRINT:PRINT"INITIALIZING . .
 . WAIT":PRINT
24 FORJ=0TO48:READOB(J),PT(J):NEXT:PE=48:FE=1:OB(46)=CHR$(13):FP
(1)=MEM
26 FORJ=1TO1099:PL(J)=J+1:NEXT:PL(1100)=N:AS=1
28 T=3001:LP=3043:RP=3044:CC=33:N1=58:N2=44:LB=3031:QU=3030:NB=3
032
30 A=0:QT=0:J=0:PRINT:PRINT"$ ";:ONERRORGOTO26000:GOSUB50:GOSUB2
65:GOSUB210:GOTO30
50 J1=0:PRINTCHR$(14);:GOSUB90
55 GOSUB100:IFX<>LPRETURN
60 J1=J1+1:X1(J1)=AS:T1(J1)=AS:LM(T1(J1))=0:AS=PL(AS):IFQRETURN
65 GOSUB55:IFX=RPGOTO80
70 IFLM(T1(J1))<>0THENPL(T1(J1))=AS:T1(J1)=AS:AS=PL(AS)
75 LM(T1(J1))=X:IFQRETURNELSE65
80 PL(T1(J1))=N:X=X1(J1):IFLM(X)=0ANDPL(X)=NTHENPL(X)=AS:AS=X:X=
N
85 J1=J1-1:RETURN
90 A$=INKEY$:IFA$=""THEN90ELSEPRINTA$;:KK=ASC(A$):RETURN
100 IFKK=40THENX=LP:GOTO200
105 IFKK=41THENX=RP:IFJ1=1ORJ1=2ANDQTRETURNELSE200
110 IFKK=39THENQ=-1:QT=QT+1:GOSUB60:LM(T1(J1))=QU:Q=0:GOSUB90:GO
SUB55:Q=-1:GOSUB70:Q=0:GOSUB80:QT=QT-1:RETURN
```

Listing continues

Basic Lisp is only a subset of a full-blown Lisp system, but it should prove useful in teaching basic aspects of Lisp programming. All the examples in this series will be geared toward Basic Lisp, but important differences between it and more standardized versions will be pointed out along the way.

Typing an expression to the interpreter is easy. After entering a statement like (ADD 1 1), there is no need to hit return. As soon as you close all the open parentheses, the expression is evaluated and answered. In this case, a 2 is returned. One important thing to remember is that atoms must be separated by a space or a carriage return, so that (ADD11) is not at all the same as (ADD 1 1).

Since Lisp is a more highly interactive language than Basic (it actually processes some of your input as you're typing it in) and since Basic Lisp is an interpreter written in another language, speed typists beware! Trying to type too fast will only get you into trouble. A moderate, steady pace is best. Note that this speed problem stems from the fact that Basic Lisp is written in Basic, and is not a problem inherent in Lisp itself.

When you make a typo, it is best to delete it immediately to avoid filling up the interpreter's internal memory with mistakes. A special function is provided in Basic Lisp to make these deletions. You should immediately close the remaining open parentheses. (Backspacing will not work.) When the prompt returns (usually after an error message warning you that a mistake has been made), type (%) and the mistake is deleted. Figure 1 shows an example of the complete routine. Again, a more sophisticated Lisp system supports far easier methods of correcting mistakes.

The actual operation of a Lisp interpreter is simple. It reads and evaluates an s-expression and prints the result (also an s-expression).

An s-expression is evaluated using a few simple rules. The value of T is T, the value of NIL is NIL, and the value of any number is itself. The value of any other atom is the s-expression it is bound to (bound and unbound atoms will be explained shortly). Type in some atoms and let the interpreter evaluate them for you.

When a list (ADD 1 1) is evaluated, the first atom is treated a a function and the rest of the elements of the list are treated as arguments to that function. This is known as prefix nota-

Listing continued

```
115 IFKK<CCGOSUB90:GOTO100ELSE125
120 IFKK<CCORKK=40ORKK=41ORKK=39THEN130
125 I$=I$+A$:GOSUB90:GOTO120
130 IFASC(I$)<N1ANDASC(I$)>N2THEN150
135 FORJ=0TOPE:IFOB(J)=I$THENX=J+N:I$="":J=0:RETURNELSENEXT
145 J=0:PE=PE+1:OB(PE)=I$:X=PE+N:I$="":RETURN
150 WW=VAL(I$):GOSUB10000:I$="":RETURN
200 GOSUB90:RETURN
210 IFA$<>CHR$(13)PRINT
215 J1=1:X1(J1)=X:GOSUB225:PRINT:RETURN
225 IFX>5000PRINT"; UNPRINTABLE MACHINE CODE";:RETURNELSEIFX>400
0PRINTFP(X-4000);CHR$(24);:RETURN
230 IFX>=NPRINTOB(X-N);:RETURN
235 IFX=0RETURN
237 IFLM(X)=QUPRINT"'";:X=LM(PL(X)):GOSUB225:RETURN
240 J1=J1+1:X1(J1)=X:PRINT"(";
245 X=X1(J1):X=LM(X):GOSUB225
250 X=X1(J1):J1=J1-1:X=PL(X):IFX=NPRINT")";:RETURNELSEIFX>NPRINT
" . ";:GOSUB225:PRINT")";:RETURNELSEIFX=0THENX=1/0
255 J1=J1+1:X1(J1)=X:PRINT" ";:GOTO245
265 FP(1)=MEM:IFX>4000ANDX<5001ORX=NORX=TRETURN
270 IFX>NTHENV=X:X=PT(X-N):IFX=0ANDA=0THENER=6:GOTO25000ELSERETU
RN
275 ST(A+1)=TT:ST(A+2)=AL:ST(A+3)=C:ST(A+4)=E:A=A+4
280 AL=PL(X):E=X:X=LM(X):GOSUB265
285 IFX>=NANDX<4001THENER=1:GOTO25000
290 IFX>6000THEN320ELSEIFX>5000THEN315ELSEIFLM(X)=LBTHEN335ELSEI
FLM(X)=NBTHEN337ELSEER=1:GOTO25000
315 TT=X:GOSUB500:ONTT-5000GOSUB4000,4010,4025,4035,4060,4070,42
95,4290,4085,4095,4130,4170,4200,4220,4230,4245,4255,4300,4315,4
310,4450:GOTO330
320 R=X:X=AL:ONR-6000GOSUB4050,50,4120,4150,4190,4285,4265,4275,
4399,4500,4600,4650,4700,4750
330 E=ST(A):C=ST(A-1):AL=ST(A-2):TT=ST(A-3):A=A-4:RETURN
335 TT=AL:E=PL(X):AL=LM(E):GOSUB500:AL=TT:GOSUB500:C=LM(E):A=A-S
T(A):GOTO340
337 TT=AL:E=PL(X):AL=LM(E):GOSUB500
338 ST(A+1)=TT:ST(A+2)=1:C=LM(E):A=A+1
340 IFC<>NTHENPT(LM(C)-N)=ST(A):A=A+1:C=PL(C):GOTO340
345 A=A-ST(A)-1:TT=PL(E)
350 IFTT<>NTHENX=LM(TT):GOSUB265:TT=PL(TT):GOTO350
355 C=LM(E):A=A-ST(A)
360 IFC<>NTHENPT(LM(C)-N)=ST(A):A=A+1:C=PL(C):GOTO360
365 A=A-ST(A)-1:GOTO330
500 C=0:IFAL=NTHENIFC=0THENA=A+1:ST(A)=0:GOTO510ELSE510
505 X=LM(AL):GOSUB265:C=C+1:A=A+1:ST(A)=X:IFPL(AL)<>NTHENAL=PL(A
L):GOTO505
510 A=A+1:ST(A)=C:RETURN
4000 IFST(A)<>1THENER=2:GOTO25000
4005 A=A-1:IFST(A)=NTHENX=N:A=A-1:RETURN
4006 IFST(A)<2001ANDST(A)>0THENX=LM(ST(A)):A=A-1:RETURN
4007 ER=4:GOTO25000
4010 IFST(A)<>1THENER=2:GOTO25000
4015 A=A-1:IFST(A)=NTHENX=N:A=A-1:RETURN
4017 IFST(A)<2001ANDST(A)>0THENX=PL(ST(A)):A=A-1:RETURN
4020 ER=4:GOTO25000
4025 IFST(A)<>2THRNER=2:GOTO25000
4030 A=A-1:T2=AS:AS=PL(AS):LM(T2)=ST(A-1):PL(T2)=ST(A):A=A-2:X=T
2:RETURN
4035 IFST(A)<>2THENER=2:GOTO25000
4040 A=A-1:IFST(A-1)<NORST(A-1)>4000THENER=3:GOTO25000
4045 PT(ST(A-1)-N)=ST(A):A=A-2:RETURN
4050 X=LM(AL):RETURN
4060 WW=0:FORJ=1TOST(A):A=A-1:IFST(A)>4000ANDST(A)<5001THENWW=WW
+FP(ST(A)-4000):NEXTELSEER=5:GOTO25000
4065 A=A-1:GOSUB10000:RETURN
4070 IFST(A)<>2THENER=2:GOTO25000
4075 A=A-1:IFST(A)<4001ORST(A)>5000ORST(A-1)<4001ORST(A-1)>5000T
HENER=5:GOTO25000
4080 WW=FP(ST(A-1)-4000)-FP(ST(A)-4000):A=A-2:GOSUB10000:RETURN
4085 WW=1:FORJ=1TOST(A):A=A-1:IFST(A)>4000ANDST(A)<5001THENWW=WW
*FP(ST(A)-4000):NEXTELSEER=5:GOTO25000
4090 A=A-1:GOSUB10000:RETURN
4095 IFST(A)<>2THENER=2:GOTO25000
4100 A=A-1:IFST(A)<4001ORST(A)>5000THENER=5:GOTO25000
4105 A=A-1:IFST(A)<4001ORST(A)>5000THENER=5:GOTO25000
4110 IFFP(ST(A+1)-4000)=0THENER=7:GOTO25000
4115 WW=FP(ST(A)-4000)/FP(ST(A+1)-4000):A=A-1:GOSUB10000:RETURN
4120 IFLM(AL)>=NANDLM(AL)<4000THENX=LM(PL(AL)):GOSUB265:PT(LM(AL
)-N)=XELSEER=3:GOTO25000
4125 AL=PL(AL):IFAL=NTHENER=2:GOTO25000ELSEAL=PL(AL):IFAL=NRETUR
NELSE4120
4130 IFST(A)<>1THENER=2:GOTO25000
4135 A=A-1:IFST(A)>=NANDST(A)<5000THENX=T:A=A-1:RETURNELSEX=N:A=
A-1:RETURN
4150 C=LM(AL):X=LM(C):GOSUB265:IFX=NTHENAL=PL(AL):IFAL=NRETURNEL
SE4150
4155 AL=PL(C)
```

Listing continues

tion and, though awkward at first, it becomes easy to read with some practice.

Figure 2 contains some examples of arithmetic in Basic Lisp. Type them in, along with a few of your own, to better understand this notation.

Note that ADD and MUL work with any number of arguments. Note also that, in the last example, the arguments to a function can be another function call. In that case, the inner function call is evaluated first, and the results are returned as arguments to the first function. In the example, the results of (ADD 2 3) and (POWER 2 .5) are then multiplied together to obtain the final answer. This ability to nest expressions in Lisp is very important and can be carried to any reasonable depth (up to 15 in Basic Lisp).

Manipulating S-Expressions with Lisp

Before delving further into Lisp, you must know the process of quoting. An apostrophe is used to quote an s-expression in Basic Lisp. By quoting something, you are telling the interpreter not to evalute any further. In ef-

```
$ (SET 'BROTHERS '(RALPH JOHN))
(RALPH JOHN)

$ (SETQ SISTERS '(SHERRY BETTY))
(SHERRY BETTY)

$ BROTHERS
(RALPH JOHN)

$ SISTERS
(SHERRY BETTY)

$ (SETQ GIRLS SISTERS)
(SHERRY BETTY)

$ GIRLS
(SHERRY BETTY)

$ (SETQ GIRLS 'SISTERS)
SISTERS

$ GIRLS
SISTERS

$ SISTERS
(SHERRY BETTY)

$ SHERRY
; SHERRY UNBOUND ATOM

$ (SETQ ONE 1 TWO 2 THREE 3)
3

$ ONE
1
```

Fig. 3. SET and SETQ

✓See List of Advertisers on Page 403

Listing continued

```
4160 X=LM(AL):GOSUB265:IFPL(AL)=NRETURNELSEAL=PL(AL):GOTO4160
4165 AL=PL(C)
4170 IFST(A)<>2THENER=2:GOTO25000
4175 A=A-1:IFST(A)=ST(A-1)THENX=TELSEX=N
4180 A=A-2:RETURN
4190 PL(E)=AS:AS=E:X=LM(AL):PT(X-N)=AL:IFLM(PL(AL))=NTHENLM(AL)=
LB:RETURNELSEIFLM(LM(PL(AL)))=LBORLM(LM(PL(AL)))=NBTHENPT(X-N)=L
M(PL(AL)):RETURNELSELM(AL)=LB:RETURN
4200 IFST(A)=0THENX=N:A=A-1:RETURNELSEX=AS:F=ST(A):A=A-F:FORJ=1T
OF:IFST(A)=0THENER=4:GOTO25000ELSEG=AS:AS=PL(AS):LM(G)=ST(A):A=A
+1:NEXT:PL(G)=N:A=A-ST(A)-1:RETURN
4220 A=A-1:IFST(A)=NTHENX=TELSEX=N
4225 A=A-1:RETURN
4230 IFST(A)<>1THENER=2:GOTO25000ELSEA=A-1
4235 IFST(A)>4000ANDST(A)<5000THENX=TELSEX=N
4240 A=A-1:RETURN
4245 IFST(A-1)>4000ANDST(A-1)<5000THENFORJ=1TOST(A)-1:A=A-1:IFST
(A-1)>4000ANDST(A-1)<5000THENIFFP(ST(A)-4000)<FP(ST(A-1)-4000)TH
ENX=T:NEXT:A=A-2:RETURNELSE4252ELSE4250
4250 ER=5:GOTO25000
4252 X=N:A=A-2:RETURN
4255 IFST(A-1)>4000ANDST(A-1)<5000THENFORJ=1TOST(A)-1:A=A-1:IFST
(A-1)>4000ANDST(A-1)<5000THENIFFP(ST(A)-4000)>FP(ST(A-1)-4000)TH
ENX=T:NEXT:A=A-2:RETURNELSE4261ELSE4260
4260 ER=5:GOTO25000
4261 X=N:A=A-2:RETURN
4265 IFAL<>NTHENX=LM(AL):GOSUB265:IFX<>NTHENAL=PL(AL):GOTO4265
4270 RETURN
4275 IFAL<>NTHENX=LM(AL):GOSUB265:IFX=NTHENAL=PL(AL):GOTO4275
4280 RETURN
4285 X=E:RETURN
4290 IFST(A)<>1THENER=2:GOTO25000ELSEA=A-1:X=ST(A):GOSUB210:X=0:
A=A-1:RETURN
4295 IFST(A)<>1THENER=2:GOTO25000ELSEA=A-1:X=ST(A):GOSUB265:A=A-
1:RETURN
4300 IFST(A)<>1THENER=2:GOTO25000
4305 A=A-1:X=ST(A):IFX>=NANDX<5000GOSUB225:X=0:A=A-1:RETURNELSEE
R=3:GOTO25000
4310 IFST(A)=0ORST(A-1)=NTHENX=N:A=A-ST(A)-1:RETURNELSEX=AS:FORJ
=A-ST(A)TOA-1:Y=ST(J):IFY=0ORY>2000ANDY<>NTHENER=4:ST(A)=Y:GOTO2
```

Listing continues

fect, you're declaring a constant. Thus, (MUL 2 3) is a function call resulting in a 6 and '(MUL 2 3) is just a list of three atoms: MUL, 2, and 3. The apostrophe is actually a shorthand for the QUOTE function and '(MUL 2 3) is represented internally as (QUOTE (MUL 2 3)). The two notations are identical in all respects, and either can be used. The single quote mark is more common because of the increased clarity.

Much like Basic variables, Lisp atoms can have values. Atoms that have been assigned a value are called bound atoms. Atoms that haven't yet received a value are called unbound atoms. Unlike regular variables, the value of an atom can be any Lisp object: a list, a number, or another atom.

There are no "string" atoms and "integer" atoms. A single atom can hold either value at different times. As mentioned earlier, the value of T, NIL, or any number is simply itself. The values of predefined function names like MUL and ADD are unprintable machine code and are actually pointers to the Basic subroutines that perform the functions. One atom that comes predefined in Basic Lisp is the atom FREE. Its value at any time is the amount of

TRS-80 memory remaining and it acts just like the MEM function in Level II.

Initially, all atoms that aren't predefined are unbound. They can be given values using the Lisp functions SET and SETQ. For example, suppose you wanted the atom BROTHERS to represent the list (RALPH JOHN). Typing in (SET 'BROTHERS '(RALPH JOHN)) would work, as would (SETQ BROTHERS '(RALPH JOHN)). Evaluating BROTHERS would return the list (RALPH JOHN).

Note the difference between SET and SETQ. SET first evaluates the atom to be bound, so that the atom must be quoted if that evaluation is to be stopped. SETQ performs no such evaluation.

Study the examples in Fig. 3, and try a few of your own, to practice using these two very important Lisp functions. Notice that SETQ can work with several assignments at once. Note also that SET and SETQ evaluate their second arguments (and fourth, sixth, and so on for SETQ) so that this argument must be quoted if that evaluation isn't desired. Remember that '2 and 2 result in the same thing, since the value of any number is itself.

Both SET and SETQ actually do two things. First, they assign the value of their second argument to their first. Second, they return the value of their last arguments. An assignment is known as a side-effect, because something has been permanently changed. Almost all Basic Lisp functions return a value, but only a few have side-effects. An example of a function without a side-effect is (ADD 1 1). This function call returns a 2, but changes nothing. Sometimes a function is used for its returned value, or for its side-effects (if any), and sometimes for both. The (SETQ B 'C) in (SETQ A (SETQ B 'C)) assigns C to B and returns C,

```
$ (CAR '((A B) (C D)))
(A B)

$ (CDR '((A B) (C D)))
((C D))

$ (CDR '(A))
NIL

$ (CAR NIL)
NIL

$ (CDR '())
NIL

$ (CAR (CDR '(RALPH SHERRY JOHN BETTY)))
SHERRY

$ (CAR '(CDR (A B C)))
CDR
```

Fig. 4. CAR and CDR

✓See List of Advertisers on Page 403

Listing continued

```
5ØØØ
4312 IFY<>NTHENZ=AS:AS=PL(AS):LM(Z)=LM(Y):Y=PL(Y):GOTO4312
4313 NEXT
4314 A=A-ST(A)-1:PL(Z)=N:RETURN
4315 IFST(A)<>2THENER=2:GOTO25ØØØ
432Ø A=A-1:IFST(A)<4ØØ1ORST(A)>5ØØØTHENER=5:GOTO25ØØØ
4325 A=A-1:IFST(A)<4ØØ1ORST(A)>5ØØØTHENER=5:GOTO25ØØØ
433Ø WW=FP(ST(A)-4ØØØ)[FP(ST(A+1)-4ØØØ):GOSUB1ØØØØ:A=A-1:RETURN
4399 IFLM(AL)<3ØØØORLM(AL)>4ØØØTHENER=1:GOTO4447ELSET2=PT(LM(AL)
-N):IFT2>2ØØØORT2=ØTHENER=1:GOTO4447ELSEIFLM(T2)<>LBANDLM(T2)<>N
BTHENER=1:GOTO4447
44ØØ PRINT:PRINT:PRINT"(DEFUN ";:X=LM(AL):A$=CHR$(13):GOSUB23Ø:P
RINT" (";:X=LM(T2):GOSUB23Ø:PRINT" ";:T2=PL(T2):X=LM(T2):J1=1:X1
(J1)=X:GOSUB225:J=Ø:J2=Ø
44Ø5 T2=PL(T2):IFT2<>NPRINT:PRINTTAB(3);:X1(J2)=-2:X=LM(T2):GOSU
B441Ø:GOTO44Ø5ELSEPRINT"))";:X=Ø:RETURN
441Ø IFX>4ØØØPRINTFP(X-4ØØØ);CHR$(24);:RETURN
4415 IFX>=NPRINTOB(X-N);:RETURN
442Ø IFLM(X)=QUPRINT"'";:X=LM(PL(X)):GOSUB225:RETURN
4425 J=J+1:T1(J)=X:D=LM(X):B=D-N:IFB=4ØORB=41ORB=31THEN4445ELSEI
FB<>6ANDB<>9ANDB<>1ØANDB<>14ANDB<>2ØANDB<>21PRINT"(";ELSE4435
443Ø X=T1(J):X=LM(X):GOSUB441Ø:X=T1(J):J=J-1:X=PL(X):IFX=NPRINT"
)";:RETURNELSEJ=J+1:T1(J)=X:PRINT" ";:GOTO443Ø
4435 T1(J)=PL(T1(J)):PRINTTAB(X1(J2)+2)"(";:J2=J2+1:X1(J2)=POS(Ø
):X=D:GOSUB4415:PRINT
444Ø X=LM(T1(J)):PRINTTAB(X1(J2)+2);:GOSUB441Ø:X=T1(J):J=J-1:X=P
L(X):IFX=NTHENJ2=J2-1:PRINT")";:RETURNELSEPRINT:J=J+1:T1(J)=X:GO
TO444Ø
4445 T1(J)=PL(T1(J)):PRINTTAB(X1(J2)+2)"(";:J2=J2+1:X1(J2)=POS(Ø
):X=D:GOSUB4415:PRINT" ";:X=LM(T1(J)):GOSUB441Ø:PRINT:T1(J)=PL(T
1(J)):GOTO444Ø
4447 E=Ø:LM(E)=LM(AL):GOTO25ØØØ
445Ø IFST(A)<>2THENER=2:GOTO25ØØØELSEA=A-1:IFST(A)>2ØØØTHENER=4:
GOTO25ØØØELSEA=A-1:IFST(A)<NORST(A)>4ØØØTHENER=3:GOTO25ØØØELSEJ=
ST(A+1):D=ST(A):X=AS:Z=N
4455 IFJ<>NTHENIFLM(J)=DGOTO446ØELSEZ=AS:AS=PL(AS):LM(Z)=LM(J)EL
SEIFZ=NTHENX=N:RETURNELSEPL(Z)=N:RETURN
446Ø J=PL(J):GOTO4455
45ØØ PRINT:PRINT"; HIT ENTER TO BEGIN";:GOSUB9Ø:PRINT#-1,FE,PE,A
S:FORJ=2TOFE:PRINT#-1,FP(J):NEXT:FORJ=49TOPE:PRINT#-1,OB(J),PT(J
```

Listing continues

which is then assigned to A by the first SETQ. The result of the entire function call is to set both A and B to C.

The function EVAL provides an extra round of evaluation beyond the one already performed. In other words, the result of evaluting the argument is then evaluated again. Figure 5 shows an example of how the function EVAL is used.

You will find it useful to be able to take lists apart. Lisp provides two functions for doing this, CAR and CDR. These functions would probably be more understandable if they had been called FIRST and REST respectively, but you are left with historical convention. CAR returns the first element of a list: (CAR '(A B C)) would return A. CDR returns a list of all of the elements of a list except the first: (CDR '(A B C)) would return (B C). Some examples of the use of these two functions are in Fig. 5.

DELETE is a function that removes parts of a list. In Basic Lisp, DELETE takes an atom and a list as arguments and returns a copy of the list with all top-level occurrences of the atom deleted. Full-blown Lisp systems can delete any s-expression from a list, but Basic Lisp can delete only atoms.

Listing continued

```
):NEXT:FORJ=1TOAS:PRINT#-1,LM(J),PL(J):NEXT:X=0:RETURN
4600 PRINT:PRINT"; HIT ENTER TO BEGIN";:GOSUB90:INPUT#-1,FE,PE,A
S:FORJ=2TOFE:INPUT#-1,FP(J):NEXT:FORJ=49TOPE:INPUT#-1,OB(J),PT(J
):NEXT:FORJ=1TOAS:INPUT#-1,LM(J),PL(J):NEXT:X=0:RETURN
4650 X=0:A=A-1:IFPE>48THENPRINT:PRINT"; ";OB(PE);" DELETED FROM
OB LIST";:PT(PE)=0:OB(PE)="":PE=PE-1
4655 RETURN
4700 TT=LM(AL):E=PL(AL):AL=E
4705 X=TT:GOSUB265:IFX<>NTHENAL=E:GOSUB4800:GOTO4705ELSERETURN
4750 TT=LM(AL):E=PL(AL):AL=E
4755 X=TT:GOSUB265:IFX=NTHENAL=E:GOSUB4800:GOTO4755ELSERETURN
4800 IFAL<>NTHENX=LM(AL):GOSUB265:AL=PL(AL):GOTO4800
4805 RETURN
10000 FORJ=1TOFE:IFFP(J)=WWTHEN10010
10005 NEXT:FE=FE+1:FP(FE)=WW:X=FE+4000:RETURN
10010 X=J+4000:RETURN
25000 X=ST(A):J1=1:X1(J)=X:IFA$<>CHR$(13)THENPRINT
25001 A$=CHR$(13):ONERGOTO25002,25003,25004,25005,25006,25007,25
008
25002 PRINT"; ";:X=LM(E):GOSUB230:PRINT" INVALID FUNCTION NAME";
:GOTO25050
25003 PRINT"; IMPROPER NUMBER OF ARGUEMENTS TO SUBR OR NSUBR";:G
OTO25050
25004 PRINT"; ";:GOSUB225:PRINT" INVALID ATOM";:GOTO25050
25005 PRINT"; ";:GOSUB225:PRINT" INVALID LIST";:GOTO25050
25006 PRINT"; ";:GOSUB230:PRINT" INVALID NUMBER";:GOTO25050
25007 PRINT"; ";:X=V:GOSUB230:PRINT" UNBOUND ATOM";:GOTO25050
25008 PRINT"; DIVISION BY ZERO";:GOTO25050
25050 X=0:ONERRORGOTO25051:P=1/0
25051 PRINT:RESUME30
26000 IFA$<>CHR$(13)PRINT
26001 IFPE>90PRINT"; OB LIST FULL":PE=90:I$="":GOTO27100
26005 IFFE>50PRINT"; FP FULL":FE=50:I$="":GOTO27100
26010 IFAS=NPRINT"; LIST MEMORY FULL":GOTO27100
26013 IFERR/2+1=9THENIFA>350ORJ1>15ORJ2>15ORJ>15PRINT"; STACK OV
ERFLOW":GOTO27000
26015 PRINT"; ERROR"
27000 RESUME30
27100 PRINT"; HIT ENTER TO REINTIALIZE, ANY OTHER KEY TO CONTINU
E ":GOSUB90:IFA$=CHR$(13)PRINTCHR$(15):RUNELSE27000
50000 DATANIL,3000,T,3001,SETQ,6003,EQ,5012,CAR,5001,CDR,5002,CO
ND,6004,DEFUN,6005,ATOM,5011,LIST,5013,APPEND,5020,ADD,5005,SUB,
5006,MUL,5009,CONS,5003,NUMBERP,5015,GREATERP,5016,LESSP,5017,EV
AL,5007
50001 DATAPRINTF,6009,AND,6007,OR,6008,DELETE,5021,SET,5004,DIV,
5010,NOT,5014,POWER,5019,PRINT,5008,PATOM,5018,READ,6002,QUOTE,6
001,LAMBDA,6006,NLAMBDA,6006,SAVE,6010,LOAD,6011,RPAREN,3044,LPA
REN,3043,QT,3045,CR,3046
50002 DATASP,3047,DOWHILE,6013,DOUNTIL,6014,%,6012,(,0,),0,',0,C
R,0," ",0,FREE,4001
```

Model II/16 Conversion

BY JESSE W. BAKER

EDIT THE FOLLOWING LINES:

```
50 J1=0:PRINTCHR$(01);:GOSUB90
55 GOSUB100:IFX<>LPTHENRETURN
60 J1=J1+1:X1(J1)=AS:T1(J1)=AS:LM(T1(J1))=0:AS=PL(AS):IFQ<>0THENRETURN
65 GOSUB55:IFX=RPTHENGOTO80
75 LM(T1(J1))=X:IFQ<>0THENRETURNELSE65
105 IFKK=41THENX=RP:IFJ1=1ORJ1=2ANDQT<>0THENRETURNELSE200
115 IFKK<CCTHENGOSUB90:GOTO100ELSE125
210 IFA$<>CUR$(13)THENPRINT
225 IFX>5000THENPRINT"; UNPRINTABLE MACHINE CODE";:RETURNELSEIFX>4000THENPRINTFP
(X-4000);CHR$(28);:RETURN
230 IFX>=NTHENPRINTOB(X-N);:RETURN
235 IFX=0THENRETURN
237 IFLM(X)=QUTHENPRINT"'";:X=LM(PL(X)):GOSUB225:RETURN
250 X=X1(J1):J1=J1-1:X=PL(X):IFX=NTHENPRINT")";:RETURNELSEIFX>NTHENPRINT" . ";:G
OSUB225:PRINT")";:RETURNELSEIFX=0THENX=1/0
265 FP(1)=MEM:IFX>4000ANDX<5001ORX=NORX=TTHENRETURN
4025 IFST(A)<>2THENER=2:GOTO25000
4125 AL=PL(AL):IFAL=NTHENER=2:GOTO25000ELSEAL=PL(AL):IFAL=NTHENRETURNELSE4120
4150 C=LM(AL):X=LM(C):GOSUB265:IFX=NTHENAL=PL(AL):IFAL=NTHENRETURNELSE4150
4160 X=LM(AL):GOSUB265:IFPL(AL)=NTHENRETURNELSEAL=PL(AL):GOTO4160
4305 A=A-1:X=ST(A):IFX>=NANDX<5000THENGOSUB225:X=0:A=A-1:RETURNELSEER=3:GOTO2500
0
4330 WW=FP(ST(A)-4000)^FP(ST(A+1)-4000):GOSUB10000:A=A-1:RETURN
4405 T2=PL(T2):IFT2<>NTHENPRINT:PRINTTAB(3);:X1(J2)=-2:X=LM(T2):GOSUB4410:GOTO44
05ELSEPRINT"))";:X=0:RETURN
4410 IFX>4000THENPRINTFP(X-4000);CHR$(28);:RETURN
4415 IFX>=NTHENPRINTOB(X-N);:RETURN
4420 IFLM(X)=QUTHENPRINT"'";:X=LM(PL(X)):GOSUB225:RETURN
26000 IFA$<>CHR$(13)THENPRINT
26001 IFPE>90THENPRINT"; OB LIST FULL":PE=90:I$="":GOTO27100
26005 IFFE>50THENPRINT"; FP FULL":FE=50:I$="":GOTO27100
26010 IFAS=NTHENPRINT"; LIST MEMORY FULL":GOTO27100
26013 IFERR/2+1=9THENIFA>350ORJ1>15ORJ2>15ORJ>15THENPRINT"; STACK OVERFLOW":GOTO
27000
27100 PRINT"; HIT ENTER TO REINTIALIZE, ANY OTHER KEY TO CONTINUE ":GOSUB90:IFA$
=CHR$(13)THENPRINTCHR$(02):RUNELSE27000
```

If lists can be taken apart, there should also be ways to put them together. CONS, LIST, and APPEND are three Lisp functions that do just that. CONS takes a list and a new first element for the list and returns a list with the new first element added. LIST makes a list out of all of its arguments. APPEND strings the top-level contents of each list given as an argument into a single list. Figure 5 shows examples of the functions DELETE, CONS, LIST, and APPEND.

You now have a good foundation of basic skills in Lisp programming and have been introduced to most of the functions of Basic Lisp. In Part II, you will put some of these pieces together as you learn how to define your own functions. ■

Randy Beer, a student of computer engineering, can be reached at 911 Lex-Ontario Road, Mansfield, OH 44903.

```
$ (SETQ GIRLS 'SISTERS)
SISTERS

$ (SETQ SISTERS '(SHERRY BETTY))
(SHERRY BETTY)

$ GIRLS
SISTERS

$ (EVAL GIRLS)
(SHERRY BETTY)

$ (DELETE 'A '(A B C))
(B C)

$ (SETQ A-LIST '(A (A B) C A))
(A (A B) C A)

$ (DELETE 'A A-LIST)
((A B) C)

$ A-LIST
(A (A B) C A)

$ (CONS 'A '(B C))
(A B C)

$ (LIST 'A '(B C) 'D)
(A (B C) D)

$ (APPEND '(A) '(B C) '((D)))
(A B C (D))

$ (LIST 'MUL 2 3)
(MUL 2 3)

$ (EVAL (LIST 'MUL 2 3))
6
```

Fig. 5. EVAL, DELETE, CONS, LIST and APPEND