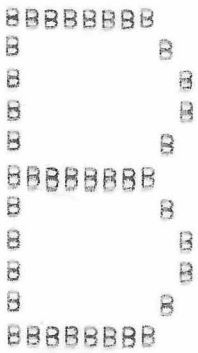
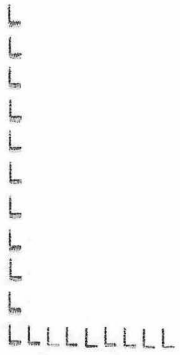
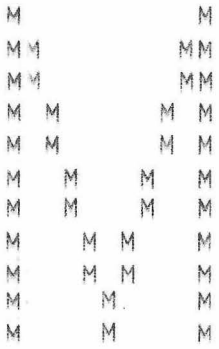
CAPEK







BALM USERS MANUAL
VERSION 3 - SEPT 69

MALCOLM C. HARRISON
JEFFREY RUBIN
DOUGLAS ALBERT

COURANT INSTITUTE
NEW YORK UNIVERSITY

ACKNOWLEDGEMENTS: THIS SYSTEM HAS BEEN DEVELOPING FOR A COUPLE OF YEARS, WITH WORK BEING CONTRIBUTED BY MANY STUDENTS INCLUDING JACK GROSS, EDE JACOBY, SHAWN SPENCER, MICHAEL GREENBERG, CHARLES WELDON, AND OTHERS.

CONTENTS

SUMMARY

THE SYSTEM DESCRIBED HERE IS AN ATTEMPT TO CREATE A SEDUCTIVE AND EXTENDABLE LANGUAGE WITH SOPHISTICATED LIST-PROCESSING AND SELF-REFERENCING FACILITIES. IT HAS FACILITIES FOR MANIPULATING NUMBERS, NAMES, STRINGS, ARRAYS, LISTS, EXPRESSIONS, AND PROCEDURES IN A HIGH-LEVEL ALGOL-LIKE LANGUAGE. IT CAN BE USED EITHER IN BATCH MODE OR CONVERSATIONAL MODE WITH ON-LINE EDITING FACILITIES. ALL DATA OBJECTS HAVE A CONVENIENT PRINTED FORM, SO INPUT/OUTPUT IS VERY SIMPLE.

PROVISION IS MADE FOR ADDING NEW PREFIX OR INFIX OPERATORS, TOGETHER WITH MACROS FOR SPECIFYING THEIR TRANSLATION INTO THE INTERMEDIATE LANGUAGE.

IT IS IMPLEMENTED IN A MIXTURE OF FORTRAN AND MLISP (A MACHINE-INDEPENDENT MACRO-LANGUAGE WHICH IS VERY SIMILAR TO LISP, TRANSLATED BY A STANDARD MACRO-ASSEMBLER), TOGETHER WITH A FEW CRITICAL MACHINE/CODED ROUTINES. USERS CAN ADD THEIR OWN ROUTINES IN FORTRAN OR MLISP, THOUGH IF THEY CHOOSE FORTRAN, A CERTAIN AMOUNT OF IMPLEMENTATION KNOWLEDGE IS NECESSARY.

1. OVERVIEW OF BALM FEATURES

A BALM PROGRAM CONSISTS OF A SEQUENCE OF COMMANDS SEPARATED BY SEMI-COLONS. EACH COMMAND WILL BE EXECUTED BEFORE THE NEXT ONE IS READ. THE USER CAN SUBMIT HIS PROGRAM EITHER AS A DECK OF CARDS, OR TYPE IT IN DIRECTLY FROM A TELETYPE. WHEN SUBMITTED AS A CARD DECK, ANY DATA REQUIRED BY THE COMMAND SHOULD FOLLOW THE COMMAND IMMEDIATELY, AND ON THE OUTPUT A LISTING OF THE CARDS WILL APPPEAR, INTERSPERSED WITH ANY PRINTED OUTPUT RESULTING FROM A COMMAND. WHEN A TELETYPE IS USED, JUST THE OUTPUT REQUESTED WILL APPEAR.

VARIABLES IN BALM DO NOT HAVE A TYPE ASSOCIATED WITH THEM, SO EACH VARIABLE CAN BE ASSIGNED ANY VALUE. THE COMMAND:

```
A = 1.2;
```

WOULD ASSIGN THE VALUE 1.2 TO A, WHILE:

```
PRINT(A);
```

WOULD PRINT OUT THE FOLLOWING:

```
1.2
```

ARITHMETIC OPERATIONS ARE EXPRESSED IN THE USUAL WAY:

```
X = 2*A+3; PRINT(X);
```

WOULD PRINT:

```
5.4
```

AUTOMATIC TYPE CONVERSION IS DONE WHERE NECESSARY.

A ≥ SYMBOL IS USED TO ALLOW THE INPUT OF LISTS:

```
A = ≥(A (B C) D);
PRINT(HD TL A);
```

WOULD PRINT:

```
(B C)
```

THE PROCEDURES HD AND TL ARE SYNONYMOUS WITH CAR AND CDR IN LISP, GIVING RESPECTIVELY THE FIRST ELEMENT OF A LIST AND THE LIST WITHOUT ITS FIRST ELEMENT. THE LISP CONS OPERATOR IS AVAILABLE EITHER AS A PROCEDURE, OR AS AN INFIX COLON ASSOCIATING TO THE RIGHT. THUS:

```
A = ≥A:≥(B C):≥D:NIL;
```

WOULD ALSO ASSIGN THE LIST ≥(A (B C) D) TO A. NOTE THAT THE EMPTY LIST IS WRITTEN AS NIL.

VECTORS CAN BE INPUT IN A NOTATION SIMILAR TO THAT FOR LISTS, BUT USING SQUARE BRACKETS INSTEAD OF PARENTHESES. ELEMENTS OF VECTORS ARE ACCESSED BY INDEXING:

```
V = ≥[A [B C] D]; PRINT(V[2]);
```

WOULD PRINT:

```
(B C)
```

LISTS CAN BE MEMBERS OF VECTORS, AND VICE VERSA:

```
PRINT(TL≥(A (B C) D));
```

WOULD PRINT:

```
((B C) D)
```

WHILE:

```
PRINT(≥[A (B C) D][2]);
```

WOULD PRINT:

```
(B C)
```

ARRAYS CAN BE INPUT AS VECTORS OF VECTORS, SO A NON-RECTANGULAR MATRIX CAN BE ASSIGNED BY:

```
     W=≥[[1][2 3][4 5 6]];
```
AND ELEMENTS CAN BE EXTRACTED BY INDEXING:
```
     PRINT(W[2]);
```
GIVING:
```
     [2 3]
```
WHICH MAY BE DONE REPEATEDLY:
```
     PRINT(W[2][1]);
```
GIVING:
```
     2
```
ASSIGNMENTS TO VECTOR ELEMENTS ARE STRAIGHTFORWARD:
```
     W[2][1]=≥(A B);
```

A WHOLE VECTOR OR LIST CAN BE ASSIGNED FROM ONE VARIABLE TO ANOTHER VARIABLE IN A SINGLE STATEMENT, OF COURSE, BUT THEN ANY OPERATION WHICH CHANGES A COMPONENT OF ONE WILL CHANGE A COMPONENT OF THE OTHER. IF THIS IS NOT DESIRED, THE VECTOR OR LIST SHOULD BE COPIED BEFORE THE ASSIGNMENT:
```
     Z = COPY(W);
```
SUBSEQUENT CHANGES TO Z WILL THEN NOT AFFECT W.

AN ARBITRARY STRUCTURE CAN BE BROKEN UP INTO ITS CONSTITUENT PARTS BY THE PROCEDURE BREAKUP. THIS TAKES TWO ARGUMENTS, A STRUCTURE WHOSE ELEMENTS ARE CONSTANTS OR VARIABLES, AND A STRUCTURE TO BE BROKEN UP. PARTS OF THE SECOND STRUCTURE CORRESPONDING TO VARIABLES ARE ASSIGNED AS THE VALUES OF THOSE VARIABLES, WHILE CONSTANTS MUST MATCH. IF THE STRUCTURES CANNOT BE MATCHED, THE BREAKUP PROCEDURE IS TERMINATED AND GIVES THE VALUE NIL. OTHERWISE IT HAS THE VALUE TRUE. FOR EXAMPLE:
```
     BREAKUP(≥(A B),≥((C C) (D D)));
```
WILL GIVE THE VALUE TRUE AND WILL ASSIGN ≥(C C) TO A AND ≥(D D) TO B. EITHER STRUCTURE CAN INVOLVE VECTORS, AND CONSTANTS IN THE FIRST STRUCTURE ARE SPECIFIED BY PRECEDING THEM WITH THE QUOTE MARK ≥. THUS:
```
     BREAKUP(≥[A ≥B C],≥[[X X] B [Y Y]]);
```
WILL HAVE THE VALUE TRUE AND WILL ASSIGN ≥[X X] TO A AND ≥[Y Y] TO B. THE CONVERSE OF BREAKUP IS CONSTRUCT, WHICH IS GIVEN A SINGLE STRUCTURE WHOSE ELEMENTS ARE VARIABLES, AND WHICH WILL CONSTRUCT THE SAME STRUCTURE BUT WITH VARIABLES REPLACED BY THEIR VALUES. THUS:
```
     X=≥(A B);Y=≥[C D];PRINT(CONSTRUCT(≥(X Y)));
```
WILL PRINT ((A B) [C D]). THESE TWO PROCEDURES ALLOW CONVENIENT FORMS SUCH AS
```
     IF BREAKUP(≥(A ≥+ B),X) THEN RETURN(CONSTRUCT(≥[A B PLUS]));
```
BREAKUP AND CONSTRUCT ARE QUITE EFFICIENT, AND SHOULD BE USED IN PREFERENCE TO THE MORE PRIMITIVE OPERATIONS WHENEVER POSSIBLE.

CHARACTER STRINGS OF ARBITRARY LENGTH CAN BE SPECIFIED:
```
     C = <EXAMPLE OF A STRING>;
                    +A + A   8 --
     E = SUBSTR(D,9,4); PRINT(E);
```
WOULD PRINT:
```
     <OF A>
```

THE BALM SYSTEM ALLOWS THE USER TO ASSIGN PROPERTIES TO VARIABLES (NAMES). A PROPERTY CONSISTS OF A NAME AND A VALUE, FOR EXAMPLE:
```
     ≥VAR PROP ≥ABCD = <STR> ;
```
ASSIGNS THE PROPERTY CALLED ABCD WITH AN ASSOCIATED VALUE OF <STR> TO THE VARIABLE VAR.

```
    X=≥VAR PROP ≥ABCD ;
```
WILL SET THE VALUE OF X TO THE VALUE OF THE PROPERTY ABCD OF VARIABLE VAR. A VARIABLE CAN HAVE ANY NUMBER OF PROPERTIES AND ANY NUMBER OF VARIABLES CAN HAVE THE SAME PROPERTY.

THERE IS COMPLETE GARBAGE COLLECTION OF ALL INACESSIBLE OBJECTS IN THE SYSTEM, SO THE USER DOES NOT NEED TO KEEP TRACK OF PARTICULAR LISTS OR VECTORS. PROCEDURES ARE AVAILABLE FOR CREATING LISTS OR VECTORS WITH VALUES OF EXPRESSIONS AS THEIR ELEMENTS, WITH STORAGE BEING ALLOCATED DYNAMICALLY:

```
    LL = LIST(Z+Q, ABC, S → <XY>);
    VV = VECTOR(X+W, ABC, S →<XY>);
```

A PROCEDURE IN BALM IS SIMPLY ANOTHER KIND OF DATA-OBJECT WHICH CAN BE ASSIGNED AS THE VALUE OF A VARIABLE, THE VARIABLE CAN THEN BE USED TO INVOKE THE PROCEDURE IN THE USUAL WAY. THE STATEMENT:

```
    SUMSQ = PROC(X,Y),X↑2+Y↑2 END;
```

ASSIGNS A PROCEDURE WHICH RETURNS AS ITS VALUE THE SUM OF THE SQUARES OF ITS TWO ARGUMENTS. THE TRANSLATOR TRANSLATES THE PROC..END PART INTO THE APPROPRIATE INTERNAL FORM, WHICH IS ASSIGNED TO SUMSQ. IN FACT THIS IS SIMPLY A LIST, WHICH COULD EQUALLY WELL HAVE BEEN CALCULATED AS THE VALUE OF AN EXPRESSION. THE PROCEDURE CAN SUBSEQUENTLY BE APPLIED:

```
    PRINT(5 + SUMSQ(2,3) + 0.5);
```

WOULD PRINT:

```
    18.5
```

INSTEAD OF ASSIGNING A PROCEDURE AS THE VALUE OF A VARIABLE, WE CAN SIMPLY APPLY IT, SO THAT:

```
    X = 5 + PROC(X,Y),X↑2+Y↑2 END(2,3) + 0.5;
```

WOULD ASSIGN 5 + 13 + 0.5 = 18.5 AS THE VALUE OF X. NOTE THAT A PROCEDURE CAN ACCEPT ANY DATA-OBJECT AS AN ARGUMENT, AND CAN PRODUCE ANY DATA-OBJECT AS ITS RESULT, INCLUDING VECTORS, LISTS, STRINGS AND PROCEDURES. THUS IT IS POSSIBLE TO WRITE:

```
    M = MSUM(M1,MPROD(M2,M3));
```

WHERE M1, M2, M3, AND M ARE MATRICES. PROCEDURES CAN BE RECURSIVE, OF COURSE.

A PROCEDURE IS SIMPLY AN EXPRESSION WITH CERTAIN VARIABLES SPECIFIED AS ARGUMENTS, SO THE RANGE OF PROCEDURES IS DETERMINED BY THE RANGE OF EXPRESSIONS. THE MOST USEFUL EXPRESSION FOR PROCEDURE DEFINITIONS IS THE BLOCK, WHICH IS SIMILAR TO THAT USED IN ALGOL, BUT CAN HAVE A VALUE. THE STATEMENT:

```
    REVERSE  =  PROC(L),
                BEGIN(X),
                COMMENT <FIRST TEST FOR ATOMIC ARGUMENT>
                IF ATOM(L) THEN RETURN(L),
                COMMENT <OTHERWISE ENTER REVERSING LOOP>
                X = NIL,
                COMMENT <EACH TIME ROUND REMOVE ELEMENT
                  FROM L, REVERSE IT, AND PUT AT BEGINNING OF X>
        NXT,    IF NULL(L) THEN RETURN (X),
                X=REVERSE(HD L):X,
                L = TL L, GO NXT
                END END;
```

SHOWS THE USE OF A BLOCK DELIMITED BY BEGIN AND END IN DEFINING A

PROCEDURE REVERSE WHICH REVERSES A LIST AT ALL LEVELS. THE COMMENT OPERATOR CAN FOLLOW ANY INFIX OPERATOR, AND WILL CAUSE THE FOLLOWING DATA-ITEM TO BE IGNORED.

AS WELL AS AN IF...THEN...STATEMENT THERE IS AN IF...THEN...ELSE... AS WELL AS AN IF...THEN...ELSEIF...THEN... ETC. LOOPING STATEMENTS INCLUDE A FOR...REPEAT...AS WELL AS A WHILE...REPEAT... . A LABEL SHOULD BE REGARDED JUST AS A LOCAL VARIABLE WHOSE VALUE IS THE INTERNAL REPRESENTATION OF THE STATEMENTS FOLLOWING IT. ACCORDINGLY, ASSIGNMENTS TO LABELS, AND TRANSFERS TO VARIABLES OR EXPRESSIONS ARE LEGAL. AND CAN GIVE THE EFFECT OF A SWITCH. A COMPOUND STATEMENT WITHOUT LOCAL VARIABLES OR TRANSFERS CAN BE WRITTEN DO.....END. OF COURSE ANY OF THESE STATEMENTS CAN BE USED AS AN EXPRESSION, GIVING THE APPROPRIATE VALUE. NOTE THAT A COMMA IS USED TO SEPARATE STATEMENTS AND LABELS WITHIN A BLOCK AND A COMPOUND STATEMENT. THE SEMICOLON IS INTERPRETED AS AN END-OF-COMMAND BY THE SYSTEM (UNLESS IT OCCURS WITHIN A STRING), EVEN IF IT OCCURS WITHIN PARENTHESES OR BRACKETS. ANY UNPAIRED PARENTHESES OR BRACKETS WILL BE PAIRED AUTOMATICALLY, WITH A WARNING MESSAGE BEING ISSUED.

2 DATA-ITEMS

A DATA-ITEM IN BALM IS EITHER A PRIMITIVE OR AN AGGREGATE. PRIMITIVE DATA-ITEMS COMPRISE INTEGERS, REALS, OCTALS, SHORT-STRINGS, NAMES, OR ENTRY-POINTS. AGGREGATE DATA-ITEMS COMPRISE VECTORS, LISTS, AND STRINGS. EXPRESSIONS AND PROCEDURES CAN ALSO BE REGARDED AS AGGREGATE DATA-ITEMS, BUT IN FACT ARE REPRESENTED INTERNALLY AS LISTS.

TYPES CAN BE DETERMINED DYNAMICALLY AT RUN TIME. ACCORDINGLY, THERE ARE NO TYPE DECLARATIONS, AND WHEN NECESSARY CONVERSION FROM ONE TYPE TO ANOTHER IS DONE DYNAMICALLY. ANY DATA-ITEM CAN BE ASSIGNED AS THE VALUE OF A VARIABLE, PRODUCED AS THE VALUE OF AN EXPRESSION, USED AS AN ARGUMENT TO A PROCEDURE, OR RETURNED AS THE VALUE OF A PROCEDURE.

2.1 CONSTANTS AND EXPRESSIONS

IN GENERAL A CONSTANT IS DISTINGUISHED FROM AN EXPRESSION (THAT IS, SOMETHING WHICH IS TO BE EVALUATED) BY PRECEEDING IT BY THE QUOTE MARK '. EXPRESSIONS WILL BE TRANSLATED BY THE BALM TRANSLATE FUNCTION INTO A LIST FORM WHICH IS ESSENTIALLY IDENTICAL TO THAT USED IN LISP 1.5. CONSTANTS WILL ALSO BE TRANSLATED, BUT USING A DIFFERENT SYNTAX.

THE SYNTAX FOR CONSTANTS IS VERY SIMPLE. A CONSTANT IS EITHER:

1. AN INTEGER
2. A REAL NUMBER
3. AN OCTAL NUMBER
4. A SHORT-STRING
5. AN ENTRY-POINT
6. A NAME PRECEDED BY A QUOTE MARK
7. A STRING
8. A LIST CONSISTING OF A QUOTE MARK, A LEFT PARENTHESIS, AN ARBITRARY NUMBER OF DATA-ITEMS SEPARATED BY SPACES IF NECESSARY, AND A RIGHT PARENTHESIS.
9. A VECTOR CONSISTING OF A QUOTE MARK, A LEFT SQUARE BRACKET, AN ARBITRARY NUMBER OF DATA-ITEMS SEPARATED BY SPACES IF NECESSARY, AND A RIGHT SQUARE BRACKET.
10. TRUE, FALSE, NIL (NIL AND FALSE ARE INTERNALLY IDENTICAL)

IN THE ABOVE A DATA-ITEM CAN BE ANY CONSTANT, BUT IF IT IS A NAME, LIST, OR VECTOR, THE LEADING QUOTE MARK MUST BE LEFT OUT.

THE SYNTAX OF CONSTANTS IS DESCRIBED IN BNF-TYPE FORM BELOW. HE NOTATION <ABC>[I,J] IS USED TO INDICATE NOT LESS THAN I AND NOT MORE THAN J OCCURRENCES OF THE SYNTACTIC TYPE <ABC>. PARENTHESES ARE USED TO GROUP ANSEQUENCES AS A SINGLE UNIT, WRITTEN AS (AND) TO AVOID

```
<CONSTANT>::=      <INTEGER>  /  <REAL>  /  <OCTAL>  /
                <SHORT-STRING>  /  <ENTRY-POINT>  /  <STRING>  /
                '  <NAME>  /  '  <LIST>  /  '  <VECTOR>
<INTEGER>::=  <SIGN>[0,1]  <DIGIT>[1,14]
<SIGN>::=     +  /  -
```

```
<REAL>::=      <SIGN>[0,1]  <DIGIT>[1,]  .  <DIGIT>[1,]
                            <EXPONENT>[0,1]
<EXPONENT>::= E  <SIGN>  <DIGIT>  <DIGIT>
<OCTAL>::=     <OCTALDIGIT>[1,20]  B
<SHORT-STRING>::=   ≠  <CHARACTER>[0,10]    ≠
<STRING>::=    <  <CHARACTER>[0,]  >
<ENTRY-POINT>::=      <LETTER>  <LETTER-DIGIT>[0,6]  <EP-TYPE>  >
<EP-TYPE>::=   0   /   1   /   2
<NAME>::=      <LETTER>  <LETTER-DIGIT>[0,69]
               /   <SPECIAL-CHARACTER>
<VECTOR>::=    [  <ITEM>[0,]  ]      .
<LIST>::=      (  <ITEM>[0,]  )   /   (  <ITEM>[1,] . <ITEM>  )
<ITEM>::=      <INTEGER>  /  <REAL>  /  <OCTAL>  /
               <SHORT-STRING>  /  <ENTRY-POINT>  /  <STRING>  /
               <NAME>  /  <LIST>  /  <VECTOR>
```

A SPACE WILL TERMINATE AN INTEGER, REAL, OCTAL, ENTRY-POINT, OR NAME, BUT WILL OTHERWISE BE IGNORED. A SEMI-COLON WILL HAVE THE EFFECT OF TERMINATING ANY INCOMPLETE VECTORS OR LISTS, AND A WARNING MESSAGE WILL BE ISSUED.
THE CHARACTER = CAN BE USED INSTEAD OF ≥ AS A QUOTE MARK.

EXAMPLES OF <ITEM>S:

```
(AA 15 17.3)
((AA 15 17.3)(ABCO> <THIS IS A STRING>))
[EL1 + <THIRD ELEMENT>]
[[<ELEMENTS>][OF TRIANGULAR][ARRAY 2 /]]
(MIXED (LISTS AND )[VECTORS AND <STRINGS>])
[MIXED(LISTS)[VECTORS AND ENTRY0> POINTS1>]]
```

2.2 NUMBERS

NUMBERS CAN BE EXPRESSED AS INTEGER, REAL, OR OCTAL, AND ALL WILL BE REPRESENTED AS FULL WORDS (60-BIT PRECISION IN THE CDC 6600). THE USUAL ARITHMETIC OPERATIONS ARE PERMITTED ON SUCH QUANTITIES, WITH AUTOMATIC TYPE CONVERSION FROM OCTAL TO INTEGER OR REAL, OR FROM INTEGER TO REAL, IF NECESSARY.

2.3 SHORT-CHARACTER-STRINGS

A CHARACTER-STRING SHORT ENOUGH TO BE CONTAINED IN A MACHINE WORD (UP TO 10 CHARACTERS OR THE CDC 6600) IS WRITTEN IN THE FORM ≠ABC≠. THE CHARACTERS ≠ AND → WITHIN SUCH CONSTANTS SHOULD BE WRITTEN AS →≠ OR →→. IN THE INTERNAL REPRESENTATION THE → WILL BE REMOVED, BUT WILL BE RE-INSERTED IF PRINTED.

2.4 NAMES

A NAME IS WRITTEN EITHER AS ANY SEQUENCE OF UP TO 70 LETTERS OR DIGITS STARTING WITH A LETTER, OR AS ANY SINGLE SPECIAL CHARACTER EXCEPT ( ) [ ] < > . AND ≠. NAMES CAN HAVE PROPERTIES WHICH CAN BE SET OR REFERRED TO. ONE IMPORTANT PROPERTY OF A NAME IS ITS VALUE, AND THE MOST FREQUENT USE OF A NAME WILL BE AS A VARIABLE, WHOSE VALUE CAN BE CHANGED BY ASSIGNMENT OR USED IN AN EXPRESSION. IF THE NAME RATHER THAN ITS VALUE IS TO BE USED IN AN EXPRESSION IT SHOULD BE PRECEDED BY A QUOTE MARK . NOTE THAT A SHORT-CHARACTER-STRING IS QUITE DIFFERENT FROM A NAME, THOUGH IT IS POSSIBLE TO CALCULATE ONE FROM THE OTHER. THE VALUES OF THE NAMES T AND F ARE INITIALIZED TO TRUE AND FALSE RESPECTIVELY.

2.5 ENTRY-POINTS

A MACHINE-CODED PROCEDURE IS REPRESENTED BY A DATA-ITEM WHICH GIVES THE ADDRESS OF THE ENTRY-POINT. BUILT-IN PROCEDURES IN BALM SIMPLY HAVE SUCH AN ITEM AS THE VALUE OF THE NAME USED TO INVOKE THEM, SO THAT THE PROCEDURE WHICH PRINTS ITS ARGUMENT IN THE OUTPUT FILE, FOR INSTANCE, IS REPRESENTED BY A VARIABLE WHOSE NAME IS PRINT AND WHOSE VALUE IS AN ENTRY-POINT. THIS VALUE CAN BE CHANGED BY AN ASSIGNMENT STATEMENT, OR CAN BE ASSIGNED AS THE VALUE OF ANOTHER VARIABLE, WHICH CAN THEN BE REGARDED AS A SYNONYM FOR PRINT. THREE TYPES OF ENTRY-POINT ARE AVAILABLE. A SUB-TYPE SUBR WHICH EXPECTS EVALUATED ARGUMENTS, A SUB-TYPE FSUBR WHICH EXPECTS A SINGLE ARGUMENT REPRESENTING A LIST OF UNEVALUATED EXPRESSIONS, AND A SUB-TYPE NSUBR WHICH EXPECTS A SINGLE ARGUMENT REPRESENTING A LIST OF EVALUATED EXPRESSIONS. THE EXTERNAL (PRINTED) FORM OF A SUBR, FSUBR, OR NSUBR IS THE NAME OF THE ENTRY-POINT AS KNOWN TO THE LOADER FOLLOWED BY THE TWO CHARACTERS 0> OR 1> OR 2>.

2.6 VECTORS

A VECTOR IS AN AGGREGATION OF ARBITRARY DATA-ITEMS EACH OF WHICH CAN BE REFERRED TO BY INDEXING. A CONSTANT VECTOR IS REPRESENTED BY A QUOTE MARK FOLLOWED BY A LEFT SQUARE BRACKET, A SEQUENCE OF CONSTANT DATA-ITEMS WITH IMPLIED QUOTE MARKS, AND TERMINATED BY A RIGHT SQUARE BRACKET. THUS:

```
≥[1.1 XYZ <ABCDE>]
```

IS A VECTOR WHOSE ELEMENTS ARE THE REAL QUANTITY 1.1, THE NAME ≥XYZ, AND THE STRING <ABCDE>, WHILE ≥[[1 2] [3 4]] IS A VECTOR WHOSE FIRST ELEMENT IS ≥[1 2] AND WHOSE SECOND ELEMENT IS ≥[3 4]. A VECTOR WHOSE VALUES ARE THE VALUES OF EXPRESSIONS CAN BE CONSTRUCTED BY USING THE PROCEDURE VECTOR, SO THAT

```
VECTOR(X+Y,F(A,B))
```

IS A VECTOR OF LENGTH 2 WHOSE FIRST ELEMENT IS THE VALUE OF X+Y AND WHOE SECOND IS THE VALUE OF F(A,B). VECTORS CAN BE INDEXED BY FOLLOWING THE VECTOR (OR EXPRESSION WHOSE VALUE IS A VECTOR) BY AN EXPRESSION SURROUNDED BY SQUARE BRACKETS. THUS IF THE VALUE OF A IS A VECTOR, THE (I+J)TH ELEMENT IS REFERRED TO BY A[I+J]. IF B IS A VECTOR WHOSE ELEMENTS ARE VECTORS, (WHICH CAN BE THOUGHT OF AS A TWO-DIMENSIONAL ARRAY, THE J TH ELEMENT OF THE I TH ELEMENT OF B CAN BE WRITTEN B[I][J] SIMILARLY, IF THE VALUE OF F(A,B) IS A VECTOR, THE EXPRESSION

```
F(A,B)[I]
```

WILL GIVE ITS I TH ELEMENT. INSTEAD OF F(A,B) ANY EXPRESSION COULD HAVE BEEN WRITTEN, ENCLOSED IN PARENTHESES IF NECESSARY. THUS THE STATEMENT

```
P[I][J]=X+(IF A≡B THEN Y ELSE Z)[I+J]+W;
```

IS QUITE LEGAL.

2.7 LISTS

LISTS ARE THE MOST WIDELY-USED AGGREGATE DATA-ITEMS IN BALM, AND ARE SIMILAR TO THOSE OF LISP 1.5. A CONSTANT LIST IS WRITTEN AS A QUOTE MARK FOLLOWED BY A LEFT PARENTHESIS, A SEQUENCE OF CONSTANT DATA-ITEMS WITH IMPLIED QUOTE MARKS, AND A RIGHT PARENTHESIS. THUS ≥(A (B C) [D E]) IS A LIST WHOSE FIRST ELEMENT IS ≥A, WHOSE SECOND IS ≥(B C) AND WHOSE THIRD IS ≥[D E]. THE PRIMITIVE PREFIX OPERATORS HD AND TL WILL GIVE THE FIRST ELEMENT OF A LIST, AND THE LIST WITHOUT ITS FIRST ELEMENT RESPECTIVELY. THE INFIX OPERATOR : WILL CONSTRUCT A NEW LIST WHOSE HD AND TL ARE TO THE LEFT AND RIGHT OF THE : OPERATOR. THE EMPTY LIST IS WRITTEN AS NIL OR () OR ≥NIL OR ≥()OR ≥[] , SO THAT THE STATEMENT:

```
L=A:B:C:NIL;
```

ASSIGNS TO L A LIST OF THREE ELEMENTS WHICH ARE THE VALUES OF THE VARIABLES A, B, AND C. THE STATEMENTS:

```
L1=HD L;
L2=TL L;
L3=B:C:NIL;
```

WILL ASSIGN THE SAME VALUES TO L2 AND L3, AND WILL ASSIGN ≥A TO L1.

2.8 STRINGS

CONSTANT CHARACTER-STRINGS OF ARBITRARY LENGTH CAN BE WRITTEN BY ENCLOSING THEM WITHIN < AND >. THE CHARACTERS > AND → CAN BE WRITTEN AS →> AND →→ RESPECTIVELY, ALTHOUGH THE FIRST → WILL NOT BE REPRESENTED INTERNALLY. THE INTERNAL REPRESENTATION OF A STRING IS SIMILAR TO A LIST OF SHORT-CHARACTER-STRINGS, WITH APPROPRIATE DISTINGUISHING FLAGS. THE SHORT-CHARACTER-STRINGS COMPRISING A STRING CAN BE OF ANY LENGTH PERMITTED FOR A SHORT-CHARACTER-STRING, BUT IT IS CONVENIENT TO DISTINGUISH PACKED STRINGS (MAXIUM LENGTH SHORT-CHARACTER-STRINGS) FROM UNPACKED STRINGS (1 CHARACTER SHORT-CHARACTER-STRINGS). ZERO LENGTH SHORT-CHARACTER-STRINGS MAY APPEAR ONLY AS THE SOLE ELEMENT IN A STRING, IN WHICH CASE THE STRING IS A NULL STRING <> . ALL BUILT-IN STRING OPERATIONS ARE INDEPENDENT OF REPRESENTATION.

2.9 EXPRESSIONS AND PROCEDURES

EXPRESSIONS AND PROCEDURES ARE STORED INTERNALLY AS LISTS, AND CAN BE MANIPULATED BY THE ORDINARY LIST OPERATORS. THE USUAL WAY OF DEFINING A PROCEDURE IS SIMPLY TO ASSIGN THE LIST REPRESENTING IT TO THE VARIABLE WHICH WILL SUBSEQUENTLY BE USED TO INVOKE IT. FOR CONVENIENCE, HOWEVER, AN ALGOL-LIKE NOTATION IS PERMITTED. FOR EXAMPLE:

```
SUMSQ=PROC(X,Y),X↑2+Y↑2 END;
```

SUBSEQUENTLY SUMSQ CAN BE USED TO APPLY THIS PROCEDURE TO ITS ARGUMENTS IN EXPRESSIONS SUCH AS:

```
Y=Z+SUMSQ(I+4,K/2);
```

THIS COULD ALSO BE WRITTEN:

```
Y=Z+PROC(X,Y),X↑2+Y↑2 END(I+J,K/2);
```

SIMILARLY, AN EXPRESSION WRITTEN IN THE FORM:

```
E=EXPR X↑2+Y↑2 END;
```

WOULD HAVE THE SAME EFFECT AS:

```
E=TRANSLATE(≥(X↑2+Y↑2));
```

AND COULD SUBSEQUENTLY BE USED IN THE FORM:

```
V=EVAL(E);
```

OR EQUIVALENTLY AS V=$E; USING THE PREFIX EVALUATION OPERATOR $. THE RESULT WILL BE THE SAME AS IF:

```
V=X↑2+Y↑2;
```

HAD BEEN WRITTEN.

3. STATEMENTS AND EXPRESSIONS

THERE IS NO DISTINCTION IN BALM BETWEEN STATEMENTS AND EXPRESSIONS. WHAT IS USUALLY THOUGHT OF AS A STATEMENT IS SIMPLY AN EXPRESSION WHOSE VALUE IS IGNORED. ANYTHING WHICH CAN BE USED AS AN EXPRESSION CAN BE USED AS A STATEMENT, AND VICE VERSA. WE WILL REFER TO BOTH AS EXPRESSIONS.

3.1 BLOCKS

A BLOCK IS SIMILAR TO AN ALGOL BLOCK, BUT CAN BE USED AS AN EXPRESSION, AND SO HAS A VALUE. IT IS DEFINED AS:

<BLOCK>::= BEGIN (<VARLIST>) <LXLIST> END
<LXLIST>::= ↑ , <LABELOREXPR> →[0,]
<VARLIST>::= <EMPTY> / <NAME> ↑ , <NAME> →[0,]
<LABELOREXPR>::= <NAME> / <EXPR>

NOTE THAT WITHIN A BLOCK LABELS AND EXPRESSIONS ARE SEPARATED BY COMMAS. EXIT FROM A BLOCK OCCURS IF THE LAST EXPRESSION IN THE BLOCK WAS EVALUATED AND DID NOT CAUSE A TRANSFER, IN WHICH CASE THE VALUE OF THE BLOCK IS NIL. EXIT ALSO OCCURS WHEN THE RETURN PROCEDURE IS EXECUTED, THE ARGUMENT OF RETURN BEING THE VALUE OF THE BLOCK.

<VARLIST> IS A LIST OF VARIABLES WHICH ARE TO BE LOCAL TO THE BLOCK. OTHER VARIABLES REFERRED TO WITHIN THE BLOCK WILL BE ASSUMED TO BE GLOBAL, AND TO HAVE HAD VALUES ASSIGNED TO THEM PREVIOUS TO ENTERING THE BLOCK. SIMILARLY THE LOCAL VARIABLES OF THE BLOCK WILL BE ACCESSIBLE AS GLOBAL VARIABLES TO ANY BLOCKS AND EXPRESSIONS INVOKED FROM WITHIN THE BLOCK. NOTE THAT GLOBAL VARIABLES TAKE ON VALUES DETERMINED BY THE CALLING BLOCK OR PROCEDURE, RATHER THAN THE ENCLOSING BLOCK OR PROCEDURE AS IN ALGOL 60.

A LABEL SHOULD BE REGARDED AS A LOCAL VARIABLE WHOSE VALUE IS A LIST WHOSE ELEMENTS ARE THE INTERNAL REPRESENTATIONS OF THE STATEMENTS FOLLOWING THE LABEL. THEY CAN BE MANIPULATED JUST LIKE ANY OTHER LOCAL VARIABLES. FOR INSTANCE, THE STATEMENT:

L←TL L,

WHERE L IS A LABEL WILL EFFECTIVELY MOVE THE LABEL DOWN ONE STATEMENT. THIS IS A TEMPORARY CHANGE, OF COURSE, ONLY EFFECTIVE FOR THE ONE INSTANCE OF THE BLOCK.

ON EXIT FROM A BLOCK, LOCAL VARIABLES AND LABELS WILL HAVE THEIR VALUES RESTORED TO THE ONES THEY HAD ON ENTRY TO THE BLOCK.

3.2 COMPOUND EXPRESSIONS

A COMPOUND EXPRESSION IS WRITTEN:

<COMPOUND>::= DO <EXPR> ↑ , <EXPR> →[0,] END

IT IS EVALUATED BY EVALUATING THE SPECIFIED EXPRESSIONS FROM LEFT TO RIGHT, AND HAS AS ITS VALUE THE VALUE OF THE LAST SUCH EXPRESSION. TRANSFER AND RETURN EXPRESSIONS WILL NOT HAVE AN IMMEDIATE EFFECT, BUT

WILL ONLY BE DETECTED AT THE END OF THE TOP LEVEL EXPRESSION OF AN ENCLOSING BLOCK.

3.3 ASSIGNMENTS

THE USUAL FORM OF ASSIGNMENT IS PERMITTED:

<ASSIGNMENT>::= <LEFTHANDSIDE> = <EXPR>
<LEFTHANDSIDE>::= <NAME> / <VECTORELEMENT> / <LISTELEMENT>
/ <STRINGELEMENT>
/ <INDIRECTREFERENCE> / <LHS-PROC-CALL>

THAT IS, EXPRESSIONS WHICH CAN APPEAR ON THE LEFT-HAND-SIDE OF AN ASSIGNMENT STATEMENT INCLUDE VARIABLES, INDEXED EXPRESSIONS, AND EXPRESSIONS WHOSE TOP-LEVEL OPERATOR IS HD, TL, EVAL, $, SUBSTR, PROP, PROPS, =, OR ≥. IF ≥ OR = IS THE TOP-LEVEL OPERATOR, THEN THE EXPRESSION IS TREATED AS A MULTIPLE ASSIGNMENT (I.E. A CALL TO BREAKUP). FOR EXAMPLE:

```
≥(A B C)=≥(W (X Y) Z);
```

WILL PERFORM THE SAME ASSIGNMENTS AS A=≥W; B=≥(X Y); C=≥Z;
ADDITIONAL FORMS OF EXPRESSION CAN BE USED ON THE LEFT-HAND-SIDE IF THE APPROPRIATE MACROS ARE DEFINED BY THE USER (SEE SECTION 5).

3.4 CONDITIONAL EXPRESSIONS

THE CONDITIONAL EXPRESSION IS OF THE FORM:

<CONDEXPR>::= IF <LOGEXPR> THEN <EXPR>
ϕ ELSEIF <LOGEXPR> THEN <EXPR> ϕ[0,]
ϕ ELSE <EXPR> ϕ[0,1]

WHERE <LOGEXPR> IS AN EXPRESSION WHICH IS TAKEN TO BE FALSE IF ITS VALUE IS NIL, AND TRUE OTHERWISE. AN EXPRESSION IS NOT EVALUATED UNLESS ITS PRECEDING <LOGEXPR> IS NOT NIL. THE FIRST SUCH PAIR TERMINATES THE EVALUATION OF THE <CONDEXPR>, WHOSE VALUE IS THAT OF THE CORRESPONDING <EXPR>. IF THERE IS NO NON-NIL <LOGEXPR>, THE VALUE OF THE ELSE CLAUSE IS TAKEN IF THERE IS ONE, AND NIL OTHERWISE. THE NAMES TRUE AND NIL ARE RESERVED WORDS WHICH SHOULD NOT BE USED AS NAMES OF VARIABLES, AND ARE ACTUALLY REPRESENTED INTERNALLY AS CONSTANTS. THE VALUE OF THE VARIABLE T IS INITIALISED TO TRUE, AND THE VALUE OF THE VARIABLES F AND FALSE ARE INITIALISED TO NIL, SO IF NOT USED AS VARIABLES THESE CAN BE USED INSTEAD OF TRUE AND NIL IN EXPRESSIONS.

3.5 TRANSFERS

<TRANSFER>::= GO <EXPR>

WHERE THE <EXPR> EVALUATES TO AN INTERNAL REPRESENTATION OF A LIST OF EXPRESSIONS. THE NORMAL FORM OF <EXPR> IS SIMPLY A NAME USED AS A LABEL. NOTE THAT:

```
L1=L2,GO L1
```

IS JUST THE SAME AS GO L2. AN ARBITRARY EXPRESSION CAN BE USED AFTER GO, INCLUDING A VECTOR ELEMENT OR PROCEDURE CALL, SO THE EXPRESSION:

```
GO IF A≡B THEN L1[I+J] ELSE F (X,Y)
```

IS THE SAME AS:

```
IF A≡B THEN GO L1[I+J] ELSE GO F(X,Y)
```

3.6 PROCEDURE CALL

A PROCEDURE CALL IS WRITTEN IN THE FORM:

<PROC-CALL>::= <PROC-EXPR> (<ARGLIST>)

<ARGLIST>::= <EMPTY> / <EXPR> ϕ , <EXPR> ϕ[0,]

WHERE <PROC-EXPR> IS AN EXPRESSION WHICH EVALUATES TO A PROCEDURE. THIS IS USUALLY A VARIABLE TO WHICH HAS BEEN ASSIGNED A SUBR, FSUBR, OR PROGRAMMER-DEFINED PROCEDURE WRITTEN AS PROC...END. HOWEVER, IT COULD ALSO BE AN ARBITRARY EXPRESSION, OR THE PROC...END EXPRESSION ITSELF.

3.7 FOR LOOPS

FOR LOOPS SIMILAR TO ALGOL CAN BE WRITTEN IN THE FORM:

<FORLOOP>::= FOR <NAME> = (<EXPR> ϕ , <EXPR> ϕ[1,2])
REPEAT <EXPR>

THE VALUES OF THE TWO OR THREE <EXPR>S ARE EVALUATED ONCE BEFORE ENTERING THE LOOP, AND WILL BE TAKEN TO BE THE INITIAL, TERMINAL, AND STEP VALUES FOR THE CONTROL VARIABLE. A STEP OF 1 WILL BE ASSUMED IF LEFT OUT. THE EXPRESSION MAY BE EVALUATED ZERO TIMES. THE STEP MAY BE NEGATIVE, IN WHICH CASE REPETION STOPS WHEN THE CONTROL VARIABLE IS LESS THEN THE TERMINAL VALUE. THE VALUE OF THE EXPRESSION IS THE VALUE OF THE LAST <EXPR>.

3.8 WHILE LOOPS

<WHILE-LOOP>::= WHILE <LOGEXPR> REPEAT <EXPR>

THE <EXPR> IS EVALUATED CONTINUOUSLY UNTIL THE VALUE OF THE <LOGEXPR> IS NIL, POSSIBLY ZERO TIMES. THE VALUE OF THE <WHILE-LOOP> IS THE VALUE OF THE LAST <EXPR> EVALUATED.

3.9 COMMENTS

<COMMENT>::= COMMENT <DATA-ITEM>

A COMMENT CAN OCCUR ANYWHERE A UNARY OPERATOR COULD OCCUR, SUCH AS AFTER A COMMA. IT WILL BE COMPLETELY IGNORED. THE USUAL FORM OF <DATA-ITEM> IS A <STRING>.

3.10 PROCEDURE DEFINITIONS

<PROC-DEF> ::= PROC (<FARGLIST>) , <EXPR> END /
FPROC (<NAME>) , <EXPR> END /
NPROC (<NAME>) , <EXPR> END

<FARGLIST> ::= <EMPTY> / <NAME> ϕ,<NAME>ϕ[0,]

PROC TYPE PROCEDURES TAKE A FIXED NUMBER OF ARGUMENTS WHICH ARE EVALUATED UPON ENTRY TO THE PROCEDURE. WHEN THE PROC IS CALLED, THE VALUES OF THE FORMAL PARAMETERS OF THE PROC (THE NAMES IN <FARGLIST>) ARE SET TO THE VALUES OF THE CORRESPONDING ACTUAL PARAMETERS. THE FORMAL PARAMETERS ARE CONSIDERED LOCAL TO THE PROC. THE VALUE OF <EXPR> IS THEN RETURNED AS THE VALUE OF THE PROC. EXAMPLE:

```
     FN=PROC(W,X,Y,Z) , X:Y:Z:NIL END;
     A=1; B=2; C=3; D=4;
     PRINT(FN(A,B,C,D));
```
WOULD PRINT OUT THE FOLLOWING:
```
     (2 3 4)
```

FPROC TYPE PROCEDURES TAKE A VARIABLE NUMBER OF ARGUMENTS. WHEN THE FPROC IS CALLED, THE VALUE OF ITS SINGLE FORMAL PARAMETER (CONSIDERED LOCAL TO THE FPROC) IS SET TO A LIST OF THE UNEVALUATED ACTUAL PARAMETERS OF THE FPROC. THE VALUE OF THE <EXPR> IS THEN RETURNED. EXAMPLE:

```
     FN=FPROC(X), TL X END;
     A=1; B=2; C=3; D=4;
     PRINT(FN(A,B,C,D));
```
WOULD PRINT OUT THE FOLLOWING:
```
     (B C D)
```

NPROC TYPE PROCEDURES TAKE A VARIABLE NUMBER OF ARGUMENTS. WHEN THE NPROC IS CALLED, THE VALUE OF ITS SINGLE FORMAL PARAMETER (CONSIDERED LOCAL TO THE NPROC) IS SET TO A LIST OF THE VALUES OF THE ACTUAL PARAMETERS TO THE NPROC. THE VALUE OF <EXPR> IS THEN RETURNED. EXAMPLE:

```
     FN=NPROC(X), TL X END;
     A=1; B=2; C=3; D=4;
     PRINT(FN(A,B,C,D));
```
WOULD PRINT OUT THE FOLLOWING:
```
     (2 3 4)
```

4. INPUT AND OUTPUT

INPUT AND OUTPUT IN BALM ARE FILE ORIENTED. ALL FILES ARE READ AND WRITTEN IN CODED MODE, AND ARE TREATED AS IF THEY WERE TAPES (IE, SEQUENTIAL ACCESS ONLY). FILE BUFFERS CAN BE CREATED DYNAMICALLY, BUT CANNOT BE RELEASED ONCE THEY HAVE BEEN CREATED. STANDARD I/O PROCEDURES WHICH OBEY BALM I/O CONVENTIONS HAVE BEEN PROVIDED FOR FILE MANIPULATION. SPECIAL I/O PROCEDURES HAVE BEEN INCLUDED FOR USERS WHO WISH TO DO THEIR OWN INPUT OR OUTPUT CONVERSIONS.

4.1 CREATION OF FILES

FILES ARE REPRESENTED WITHIN THE BALM SYSTEM AS SPECIALLY FORMATTED VECTORS WHICH CONTAIN I/O BUFFERS, LINE BUFFERS, FLAGS, AND SYSTEM INFORMATION. FILES ARE CREATED BY CALLING THE PROCEDURE FILE , WHICH RETURNS A FILE. FOR EXAMPLE,

```
TAPE1=FILE(≠PUNCH≠,80);
```

CREATES A FILE WHOSE SCOPE NAME IS PUNCH AND WHOSE LINE LENGTH IS 80 CHARACTERS. THE NAME TAPE1 CAN BE USED TO REFER TO THIS FILE. FOR EXAMPLE,

```
WRITE(=(A MESSAGE),TAPE1);
```

WILL WRITE THE LIST (A MESSAGE) ON THE SCOPE FILE PUNCH. FILES CAN BE ASSIGNED AS THE VALUES OF VARIABLES, RETURNED BY PROCEDURES, MADE ELEMENTS OF LISTS OR VECTORS, ETC. FILE BUFFERS RESIDE IN BLOCKSPACE, AND ONCE THEY ARE CREATED THEIR STORAGE CANNOT BE RELEASED.
THE VALUES OF THE VARIABLES INPUT AND OUTPUT ARE INITIALIZED TO THE STANDARD INPUT AND OUTPUT FILES BY THE BALM SYSTEM.

4.2 STANDARD I/O PROCEDURES

ALL STANDARD I/O PROCEDURES TAKE A FILE AS ONE OF THEIR ARGUMENTS. MANY OF THE PROCEDURES PERMIT THE USER TO OMIT THE ARGUMENT FILE, IN WHICH CASE THE STANDARD I/O FILES (AS DECLARED ON THE BALM CONTROL CARD) ARE ASSUMED. THE FOLLOWING PROCEDURES HAVE BEEN PROVIDED FOR FILE MANIPULATION:

```
READ(TAPE)
WRITE(ITEM,TAPE)
REWIND(TAPE)
ENDFILE(TAPE)
BACKSPACE(TAPE)
LINES(NUMBER,TAPE)
EJECT(TAPE)
```

THE VALUE OF TAPE IS ASSUMED TO BE A FILE IN THE ABOVE EXAMPLES.
THE BALM I/O ROUTINES WILL OPTIONALLY PERFORM THE FOLLOWING OPERATIONS DURING INPUT OR OUTPUT:

1. ECHOING OF INPUTTED LINES ON THE STANDARD OUTPUT FILE
INITIALLY THIS OPTION IS ENABLED (SWITCHED ON) FOR THE STANDARD INPUT FILE, UNLESS THE TTY PARAMETER IS PRESENT ON THE BALM CONTROL CARD. ECHOING IS INITIALLY DISABLED FOR FILES CREATED BY THE FILE PROCEDURE. THE PROCEDURE ECHO CAN BE USED TO ENABLE OR DISABLE THIS OPTION.

2. AUTOMATIC INSERTION OF CHARIAGE CONTROL CHARACTERS DURING OUTPUT
INITIALLY THIS OPTION IS ENABLED FOR THE STANDARD OUTPUT FILE AND ALL CREATED FILES. THE OPTION CAN BE SWITCHED ON OR OFF BY THE CARRIAGE PROCEDURE. CARRIAGE ALSO PERMITS THE USER TO SPECIFY WHICH CHARACTER IS TO BE INSERTED FOR CARRIAGE CONTROL.

3. INDENTED OUTPUT OF DATA STRUCTURES
THIS OPTION, WHICH IS INITIALLY ENABLED FOR ALL FILES, CAN BE SWITCHED ON OR OFF BY CALLING THE PRETTYPRINT PROCEDURE.

4.3 SPECIAL I/O PROCEDURES

SPECIAL I/O PROCEDURES HAVE BEEN PROVIDED FOR USERS WHO WISH TO INPUT OR OUTPUT DATA-STRUCTURES IN NON-STANDARD FORMATS. THE PROCEDURES BLANKS AND WRITENF INTRODUCE CHARACTERS INTO THE OUTPUT LINE BUFFERS OF FILES. WRITE OR FLUSH CAN THEN BE USED TO COMPLETE THE LINE AND TRANSFER IT TO THE OUTPUT BUFFER. IF EVEN MORE CONTROL OVER I/O FORMAT IS DESIRED, READLINE AND WRITELINE CAN BE USED. THESE PROCEDURES PERFORM INPUT OF A LINE DIRECTLY INTO A STRING AND OUTPUT DIRECTLY FROM A STRING. THE CHARACTERS < > → ARE TREATED AS ORDINARY CHARACTERS BY READLINE AND WRITELINE. THE FOLLOWING ADDITIONAL PROCEDURES WILL SIMPLIFY (HOPEFULLY) THE WORK OF USER-FORMATTED I/O:

```
OPCHARP(CHAR)
NUMCHARP(CHAR)
ALPHCHARP(CHAR)
MKVAR(STRNG)
MKNUM(CHAR)
VARNAME(VAR)
```

5. EXAMPLES

WE GIVE BELOW A GENERALIZED GAME-PLAYING ROUTINE BEST , USING AN ALPHA-BETA MINIMAX SEARCH. WITH THE APPROPRIATE ROUTINES POSSMOVES, NEWPOSN, AND SCORE ADDED, IT WILL PLAY TIC-TAC-TOE, CHECKERS, CHESS, OR GO, WITH VARYING DEGREES OF SUCCESS.

ARGUMENTS TO BEST ARE: POSN, THE POSITION FROM WHICH THE MOVE MUST BE MADE; DEPTH, THE NUMBER OF MOVES THE ROUTINE SHOULD LOOK AHEAD; HISB AND MYB, THE RANGE BETWEEN WHICH SCORES ARE ACCEPTABLE, WITH HISB BEING THE MINIMUM AND MYB THE MAXIMUM, LARGE SCORES BEING GOOD. FOR TIC-TAC-TOE, FOR INSTANCE, POSN IS OF THE FORM:

```
[[X--][-0-][---]X]
```

THE EXTRA X INDICATING IT IS X TO MOVE.

THE FULL DECK TO PLAY A COMPLETE GAME OF TIC-TAC-TOE AGAINST ITSELF IS CONSTRUCTED AS FOLLOWS:

```
A123456,CM60000.    HARRISON
BALM.
EOR.
COMMENT <TEST PROGRAM - TIC-TAC-TOE PLAYER>

 BEST=PROC(POSN,DEPTH,HISB,MYB),
       BEGIN(ML,DM1,BESTSC,BESTM,TRY),
       IF DEPTH=0 THEN RETURN(-SCORE(POSN):NIL:NIL),
       ML=POSSMOVES(POSN),
       DM1=DEPTH-1,BESTSC=MYB,BESTM=NIL,
 NXT,  IF ¬ML THEN RETURN(-BESTSC:BESTM:NIL),
       TRY=BEST(NEWPOSN(POSN,HD ML),DM1,-BESTSC,-HISB),
       IF HD TRY GT BESTSC THEN
          DO BESTSC=HD TRY,BESTM=HD ML END,
       IF ¬ BESTSC LT HISB THEN RETURN(-BESTSC:BESTM:NIL),
       ML=TL ML, GO NXT;

 SCORE=PROC(P),
       BEGIN(ROWSC,M,H,I,J),
 ROWSC=PROC(INIT,STEP),
       BEGIN(NM,NH,IJ,S),
       $INIT,NM=0,NH=0,
       FOR IJ=(1,3) REPEAT
          DO IF (S=P[I][J])=M THEN NM=NM+1
                ELSEIF S=H THEN NH=NH+1,
          $STEP END,
       RETURN(IF (NM*NH)=0 THEN NM*NM-NH*NH ELSE 0)
       END END,
       M=P[4],H=IF M=≥X THEN ≥0 ELSE ≥X,
       RETURN
        (ROWSC(EXPR I=J=1 END,EXPR J=J+1 END) +
         ROWSC(EXPR I=1+(J=1) END,EXPR J=J+1 END) +
         ROWSC(EXPR I=2+(J=1) END,EXPR J=J+1 END) +
         ROWSC(EXPR I=J=1 END,EXPR I=I+1 END) +
         ROWSC(EXPR J=1+(I=1) END,EXPR I=I+1 END) +
         ROWSC(EXPR J=2+(I=1) END,EXPR I=I+1 END) +
```

```
        ROWSC(EXPR I=J=1 END,EXPR I=J=J+1 END) +
        ROWSC(EXPR I=2+(J=1) END,EXPR DO I=I-1,J=J+1 END END) );

POSSMOVES=PROC(P),
      BEGIN(I,J,M),
      FOR I=(1,3) REPEAT
         FOR J=(1,3) REPEAT
            IF P[I][J]≡≥- THEN M=(I:J:NIL):M,
      RETURN(M);

NEWPOSN=PROC(P,M),
      BEGIN(N),
      N=COPY(P),
      N[(HD  M)][(HD  TL  M)]=P[4],
      N[4]=IF P[4]≡≥X THEN ≥0 ELSE ≥X,
      RETURN(N);


PLAY=PROC(),
      BEGIN(POSN,BM),
      POSN=≥[[- - -][- - -][- - -] X],
 NEXT,PRINT(POSN),
      BM=HD TL BEST(POSN,1,9999,-9999),
      IF BM=NIL THEN RETURN(POSN),
      POSN=NEWPOSN(POSN,BM),
      GO NEXT;

PLAY();
EXIT();
EOF.
```

THIS USES A LOOK-AHEAD OF ONLY ONE LEVEL, PRINTS OUT EACH POSITION, AND TERMINATES WHEN ALL SQUARES ARE OCCUPIED.

6. USER-DEFINED LANGUAGE EXTENSIONS

THE TRANSLATE PROCEDURE USED BY BALM TO TRANSLATE STATEMENTS INTO THE APPROPRIATE INTERNAL FORM IS PARTICULARLY SIMPLE, CONSISTING OF A PRECEDENCE ANALYSIS PASS FOLLWED BY A MACRO-EXPANSION PASS. BUILT-IN SYNTAX IS PROVIDED ONLY FOR PARENTHESIZED SUBEXPRESSIONS, COMMENTS, THE QUOTE OPERATOR , THE UNARY OPERATOR NOOP, PROCEDURE CALLS, AND INDEXING. ALL OTHER SYNTAX INFORMATION IS PROVIDED IN THE FORM OF THREE LISTS WHICH ARE THE VALUES OF THE VARIABLES UNARYLIST, INFIXLIST, AND MACROLIST. THE USER CAN MANIPULATE THESE LISTS AS HE WISHES, BY ADDING, DELETING, OR CHANGING OPERATORS OR MACROS.

OPERATORS ARE CATEGORIZED AS UNARY, BRACKET, OR INFIX, AND HAVE PRECEDENCE VALUES AND A PROCEDURE (OR MACRO) ASSOCIATED WITH THEM. EXAMPLES OF UNARY OPERATORS ARE -(MINUS), CAR, AND IF, WHILE INFIX OPERATORS INCLUDE +, THEN, AND ELSE. BRACKET OPERATORS ARE SIMILAR TO UNARY OPERATORS BUT REQUIRE A TERMINATING INFIX OPERATOR WHICH IS IGNORED. EXAMPLES OF BRACKETS OPERATORS ARE BEGIN AND PROC, WHICH BOTH CAN BE TERMINATED BY THE INFIX OPERATOR END.

NEW OPERATORS CAN BE DEFINED BY THE PROCEDURES UNARY, BRACKET, OR INFIX. THESE ADD APPROPRIATE ENTRIES ONTO UNARYLIST OR INFIXLIST. FOR EXAMPLE THE STATEMENT:

```
UNARY(≥PR,150,≥PRINT);
```

WOULD ESTABLISH THE UNARY OPERATOR PR WITH PRIORITY 150 AS BEING THE SAME AS THE PROCEDURE PRINT. THUS WE COULD SUBSEQUENTLY WRITE PR A INSTEAD OF PRINT(A). SIMILARLY WE COULD DEFINE AN INFIX OPERATOR BY

```
INFIX(≥→,49,50,≥APPEND);
```

TO ALLOW AN INFIX APPEND OPERATION. THE NUMBERS 49 AND 50 ARE THE PRECEDENCES OF THE OPERATOR WHEN IT IS CONSIDERED AS A LEFT-HAND AND RIGHT-HAND OPERATOR RESPECTIVELY, SO THAT AN EXPRESSION SUCH AS A→B→C WILL BE ANALYZED AS THOUGH IT WERE A→(B→C)

THE OUTPUT OF THE PRECEDENCE ANALYSIS IS A TREE EXPRESSED AS A LIST IN WHICH THE FIRST ELEMENT OF EACH LIST OR SUBLIST IS AN OPERATOR OR MACRO. FOR EXAMPLE, THE STATEMENT:

```
SQ=PROC(X),X*X END;
```

WOULD BE INPUT AS THE LIST:

```
(SQ = PROC (X) , X * X END)
```

AND WOULD BE ANALYZED INTO:

```
(SETQ SQ (PROC (COMMA X (TIMES X X))))
```

THIS WOULD THEN BE EXPANDED BY THE MACRO-EXPANDER, GIVING:

```
(SETQ SQ (QUOTE (LAMBDA (X) (TIMES X X))))
```

THE APPROPRIATE INTERNAL FORM. THIS WOULD THEN BE EVALUATED, HAVING THE SAME EFFECT AS THE STATEMENT:

```
SQ = (LAMBDA(X) (TIMES X X));
```

WHICH WOULD IN FACT BE TRANSLATED INTO THE SAME THING.

THE MACRO-EXPANDER IS A FUNCTION EXPAND WHICH IS GIVEN THE SYNTAX TREE AS ITS ARGUMENT. IT IS ACTUALLY DEFINED AS:

```
EXPAND = PROC(TR),
      BEGIN(Y),
```

```
        IF ATOM(TR) THEN RETURN(TR),
        Y=LOOKUP(HD TR,MACROLIST),
        IF ¬Y THEN RETURN (MAPX(EXPAND,TR)),
        RETURN(Y(TR));
```

THAT IS, IF THE TOP-LEVEL OPERATOR IS A MACRO, IT IS APPLIED TO THE WHOLE TREE. OTHERWISE EXPAND IS APPLIED TO EACH OF THE SUBTREES RECURSIVELY. MOST OPERATORS WILL NOT REQUIRE MACROS BECAUSE THE OUTPUT OF THE PRECEDENCE ANALYSIS IS IN THE CORRECT FORM. HOWEVER, OPERATORS SUCH AS IF, THEN, FOR, PROC ... ETC. REQUIRE THEIR ARGUMENTS TO BE PUT IN THE CORRECT FORM FOR THE INTERPRETER. FOR INSTANCE, THE IF MACRO CAN BE DEFINED:

```
    MIF=PROC(TR),
        BEGIN(X),
        X=HD TL TR,
        IF HD X=≥THEN THEN RETURN
          (≥COND:LIST(EXPAND(HD TL   X),EXPAND(HD  TL  TL  X)):NIL),
        RETURN(≥COND:EXPAND(X));
```

WHERE RECURSIVE CALLS TO EXPAND ARE USED TO TRANSFORM SUBTREES IN THE APPROPRIATE WAY. THE STATEMENT:

```
    MACRO(≥IF,MIF);
```

WOULD ASSOCIATE THE MACRO MIF WITH THE OPERATOR IF.

ONE PARTICULARLY USEFUL OUTCOME OF THIS EXPANSION PROCEDURE IS THE ABILITY TO WRITE PROCEDURES ON THE LEFT-HAND SIDE OF ASSIGNMENT STATEMENTS. THESE CAN BE HANDLED BY A MACRO ASSOCIATED WITH THE ASSIGNMENT OPERATOR, WHICH TESTS FOR PARTICULAR EXPRESSIONS ON THE LEFT-HAND SIDE AND MAKES APPROPRIATE MODIFICATIONS. FOR INSTANCE, THE HD OPERATOR USED ON THE LEFT ALLOWS:

```
    HD X=Y
```

TO BE WRITTEN INSTEAD OF:

```
    RPLACA(X,Y);
```

WHICH THE SETQ MACRO WILL IN EFFECT PRODUCE. SIMILARLY, WE CAN WRITE:

```
    MACRO(IF)=PROC(TR) ... ;
```

AS A MORE CONCISE WAY OF DEFINING THE IF MACRO, AS LONG AS THE SETQ MACRO WERE PREPARED FOR THIS LEFT-HAND SIDE FORM.

TO PROVIDE THIS FLEXIBILITY, THE MACRO FOR THE ASSIGNMENT OPERATOR IS DEFINED TO BE:

```
    MSETQ=PROC(X),
        BEGIN(E1,E2,LM)
        E1=HD TL X, E2=HD TL TL X,
        IF ATOM(E1) THEN GO NOTLM,
        LM=LOOKUP(HD E1,≥LMACROLIST),
        IF NULL(LM) THEN GO NOTLM,
        RETURN (LM(X)),
    NOTLM,RETURN(≥SETQ:EXPAND(E1):EXPAND(E2):NIL);
```

THIS LOOKS UP TOP-LEVEL OPERATORS ON THE LEFT-HAND-SIDE ON THE LIST LMACROLIST, WHICH CAN BE EXTENDED BY THE USER. THE STATEMENT:

```
    LMACRO(NAME,LMAC);
```

ADDS MACROS TO LMACROLIST IN A WAY ANALOGOUS TO MACRO.

7. THE BALM COMPILER

A BALM COMPILER WHICH WILL CHANGE A USER DEFINED PROCEDURE IN BALM INTERNAL FORM (A LIST STRUCTURE) INTO MACHINE CODE (A VECTOR OF MACHINE LANGUAGE INSTRUCTIONS) IS AVAILABLE. THE COMPILER IS ITSELF A PROCEDURE WRITTEN IN BALM AND CALLED COMPILE. IT IS NOT A PERMANENT PART OF THE BALM SYSTEM, AND CAN BE INCLUDED OR ERASED AT THE USER S REQUEST. MACHINE CODED (COMPILED) PROCS, FPROCS, AND NPROCS ARE KNOWN AS SUBRS, FSUBRS, AND NSUBRS RESPECTIVELY. COMPILED PROCEDURES EXECUTE APPROXIMATLY 5 TIMES FASTER THAN UNCOMPILED PROCEDURES IN BALM INTERNAL FORM.

6.1 DEFINING THE COMPILER

THE BALM COMPILER IS SIMPLY A DECK OF PUNCHED CARDS CONTAINING THE BALM SOURCE STATEMENTS TO DEFINE THE PROCEDURE COMPILE AND ITS ASSOCIATED PROCEDURES. IN ORDER TO INCLUDE THE COMPILER IN THE BALM SYSTEM, THE USER PLACES THE COMPILER DECK BEFORE HIS OWN BALM STATEMENTS. THIS WILL CAUSE THE STATEMENTS IN THE COMPILER DECK TO BE TRANSLATED AND EXECUTED FIRST, THUS ESTABLISHING THE PROPER DEFINITIONS. THE USER CAN MODIFY THE COMPILER SOURCE CARDS IN ORDER TO PROVIDE HIS OWN SPECIAL PURPOSE COMPILER.

AN ALTERNATE WAY TO OBTAIN THE COMPILER IS TO READ ITS SOURCE STATEMENTS FROM A SCOPE COMMON FILE. THE COMMON FILE BALMC WILL USUALLY BE AVAILABLE AND WILL CONTAIN THE COMPILER DEFINITIONS. THE SCOPE CONTROL CARDS NECESSARY TO OBTAIN THIS COMMON FILE AND TO EXECUTE BALM ARE:

```
ID-CARD
COMMON(BALMC)
REWIND(BALMC)
COPYBF(BALMC,TAPE3)
RETURN(BALMC)
REWIND(TAPE3)
BALM.
E-O-R

BALM STATEMENTS

E-O-F
```

THE BALM STATEMENT

```
  COMPILER=FILE(≠TAPE3≠);
  EXECUTE(COMPILER,OUTPUT);
```

WILL THEN READ, TRANSLATE, AND EXECUTE THE COMPILER DEFINITIONS. IN ORDER TO ERASE THE COMPILER, CALL THE FUNCTION EXCISE(). TO RESTORE THE COMPILER AFTER A CALL TO EXCISE (IF THE COMPILER WAS OBTAINED FROM A COMMON FILE) -

```
  REWIND(COMPILER);
  EXECUTE(COMPILER,OUTPUT);
```

6.2 CALLING THE COMPILER

THE COMPILE PROCEDURE HAS TWO ARGUMENTS: THE NAME OF THE PROCEDURE TO BE COMPILED AND A SHORT STRING OF LESS THAN 8 CHARACTERS REPRESENTING THE ENTRY POINT NAME OF THE PROCEDURE. THE ENTRY POINT NAME IS USED INTERNALLY BY THE SYSTEM, AND RE-APPEARS EXTERNALLY ONLY IF THE USER PRINTS THE ENTRY POINT OF THE COMPILED PROCEDURE (EG. PRINT($ABS); WILL PRINT KABS0> , WHERE KABS IS THE ENTRY POINT NAME OF THE MACHINE CODED PROCEDURE ABS). ENTRY POINT NAMES OF COMPILED PROCEDURES NEED NOT BE UNIQUE.

IF THE PROCEDURE TO BE COMPILED CALLS ANY UNCOMPILED PROCEDURES, THEN THESE PROCEDURES WILL BE AUTOMATICALLY COMPILED. THE COMPILER WILL GIVE THEM ALL THE ENTRY POINT NAME COMPED.

EXAMPLE:

```
-COMMENT<READ IN COMPILER>
-EXECUTE(FILE(≠TAPE3≠),OUTPUT);
-COMMENT<DEFINE PROCEDURES TO BE COMPILED>
-REV=PROC(X), IF ATOM(X) THEN X ELSE REV1(X,NIL) END;
-REV1=PROC(X,Y), IF -X THEN Y ELSE REV1(TL X,REV(HD X);Y) END;
-COMMENT<NOW COMPILE BOTH PROCEDURES>
-COMPILE(≥REV,≠SREV≠);
-PRINT($REV);
SREV0>
-PRINT($REV1);
COMPED0>
```

6.3 SOURCE CODE RESTRICTIONS

PROCEDURES WHICH ARE SUBRS, FSUBRS, OR NSUBRS AT COMPILATION TIME MUST NOT BE REDEFINED AFTER THE COMPILATION.

IF AN EXPRESSION IS USED AS A FUNCTION, THEN IT WILL BE EVALUATED AT COMPILATION TIME. IF THE VALUE IS NIL THEN THE FUNCTION IS ASSUMED TO BE UNDEFINED. THIS WILL NOT CAUSE AN ERROR MESSAGE, THOUGH, AS LONG AS THE FUNCTION BECOMES DEFINED BEFORE THE COMPILED CALL TO IT IS EXECUTED. IF THE VALUE OF AN EXPRESSION USED AS A FUNCTION IS A SUBR, FSUBR, OR NSUBR THEN A DIRECT LINKAGE TO IT WILL BE COMPILED. IF THE VALUE IS A PROC, FPROC, OR NPROC THEN IT WILL BE CONVERTED INTO A SUBR, FSUBR, OR NSUBR (IE. THE REFERENCED PROCEDURE WILL BE COMPILED). THE VALUES OF ALL LOCAL VARIABLES AND FORMAL PARAMETERS OF PROCEDURES BEING COMPILED ARE SET TO NIL DURING COMPILATION AND ARE RESTORED AFTERWARDS.

THE INTERNAL STRUCTURE OF A LABEL IN COMPILED CODE DIFFERS FROM THE INTERNAL STRUCTURE OF AN UNCOMPILED LABEL. STATEMENTS SUCH AS GO TL LABEL ARE ILLEGAL IN SOURCE CODE TO BE COMPILED.

SUBRS MAY HAVE A MAXIMUM OF 5 ARGUMENTS. ALL PROCS OF MORE THAN 5 ARGUMENTS SHOULD BE RE-WRITTEN AS NPROCS BEFORE THEY ARE COMPILED.

8. CATALOGUE OF OPERATORS AND PROCEDURES

THE OPERATORS IN THE CURRENT SYSTEM, IN ORDER OF DECREASING PRECEDENCE, ARE AS FOLLOWS:

OPERATOR	TYPE	PROCEDURE	LEFT PREC.	RIGHT PREC.	COMMENT
≥	UNARY	QUOTE	-	-	HANDLED BY TRANSLATOR
=	UNARY	QUOTE	-	-	HANDLED BY TRANSLATOR
COMMENT	UNARY	-	-	-	HANDLED BY TRANSLATOR
NOOP	UNARY	-	-	-	HANDLED BY TRANSLATOR
PROP	INFIX	PROP	2100	2101	RIGHT-ASSOCIATIVE
PROPS	INFIX	PROP	2100	2101	RIGHT-ASSOCIATIVE
HD	UNARY	CAR	2000	-	
TL	UNARY	CDR	2000	-	
$	UNARY	EVAL	1900	-	
↑	INFIX	EXPT	1800	1801	RIGHT-ASSOCIATIVE
-	UNARY	MINUS	1700	-	
/	INFIX	QUOTIENT	1601	1600	
*	INFIX	TIMES	1601	1600	
+	INFIX	PLUS	1501	1500	
-	INFIX	DIFFERENCE	1501	1500	
EQ	INFIX	EQUAL	1400	1400	
≡	INFIX	EQUAL	1400	1400	
NE	INFIX	NEQUAL	1400	1400	
GT	INFIX	GREATERP	1400	1400	
LT	INFIX	LESSP	1400	1400	
GE	INFIX	NLESSP	1400	1400	
LE	INFIX	NGREATERP	1400	1400	
ZR	UNARY	ZEROP	1300	-	
NZ	UNARY	NZEROP	1300	-	
PL	UNARY	NMINUSP	1300	-	
NG	UNARY	MINUSP	1300	-	
VAR	UNARY	ATOM	1300	-	
NOT	UNARY	NOT	1200	-	
NULL	UNARY	NULL	1200	-	
¬	UNARY	NULL	1200	-	
AND	INFIX	AND	1100	1101	RIGHT-ASSOCIATIVE
^	INFIX	AND	1100	1101	RIGHT-ASSOCIATIVE
OR	INFIX	OR	1000	1001	RIGHT-ASSOCIATIVE
v	INFIX	OR	1000	1001	RIGHT-ASSOCIATIVE
→	INFIX	STRING	900	901	RIGHT-ASSOCIATIVE
:	INFIX	CONS	800	801	RIGHT-ASSOCIATIVE
←	INFIX	SETQ	700	701	RIGHT-ASSOCIATIVE
REPEAT	INFIX	-	600	600	DETECTED BY MACROS
GO	UNARY	GO	500	-	
GOTO	UNARY	GO	500	-	
FOR	UNARY	FORLOOP	500	-	
WHILE	UNARY	WHILE	500	-	
RETURN	UNARY	RETURN	500	-	
THEN	INFIX	-	400	400	DETECTED BY MACROS
ELSE	INFIX	-	300	300	DETECTED BY MACROS
ELSEIF	INFIX	-	300	300	DETECTED BY MACROS

IF	UNARY	COND	200	-	
,	INFIX	-	100	100	DETECTED BY MACROS
PROC	BRCKT	LAMBDA	100	-	
FPROC	BRCKT	FLAMBDA	100	-	
EXPR	BRCKT	-	100	-	DETECTED BY MACROS
BEGIN	BRCKT	PROG	100	-	
DO	BRCKT	PROGN	100	-	
END	INFIX	-	0	0	BRACKET TERMINATOR
TERMINATOR	INFIX	-	0	0	HANDLED BY TRANSLATOR

THE PRE-DEFINED PROCEDURES IN THE CURRENT SYSTEM, IN ALPHABETICAL ORDER, ARE AS FOLLOWS:

ABS(ARG)
RETURNS THE ABSOLUTE VALUE OF ARG.

ALPHCHARP(CHAR)
RETURNS TRUE IF THE SHORT-STRING CHAR IS AN ALPHABETIC CHARACTER (A-Z) , OTHERWISE NIL.

ANALYZE(LST)
PERFORMS A PRECEDENCE ANALYSIS ON LIST LST AND RETURNS THE RESULT. (SEE SEC. 5)

APPEND(LST1,LST2)
RETURNS THE LIST FORMED BY CONCATENATING LISTS LST1 AND LST2.
EXAMPLE: APPEND(≥(A B),≥(C D)) RETURNS (A B C D)

ATOM(ARG)
RETURNS TRUE IF ARG IS A NAME, NUMBER, SHORT STRING, ENTRY POINT, TRUE, OR NIL. NIL IS RETURNED FOR ALL OTHER ARGUMENTS.

BACKSPACE(TAPE)
BACKSPACES THE FILE TAPE ONE LOGICAL RECORD.

BLANKS(N,TAPE)
INTRODUCES N BLANK CHARACTERS INTO THE CURRENT LINE OF OUTPUT FILE TAPE. IF TAPE IS OMITTED, THEN THE STANDARD OUTPUT FILE IS ASSUMED.

BLOCKC(ARG)
RETURNS ARG CONVERTED TO A VECTOR. IF ARG IS A LIST, THEN THE RESULTING VECTOR WILL CONSIST OF THE TOP LEVEL ELEMENTS OF ARG. IF ARG IS A STRING, THEN THE RESULT WILL BE A VECTOR OF SHORT STRINGS, EACH OF WHICH IS A SINGLE CHARACTER OF ARG. IF ARG IS A VECTOR THEN IT IS RETURNED.

BLOCKP(ARG)
RETURNS TRUE IF ARG IS A VECTOR, OTHERWISE NIL.

BRACKET(OP,PREC,PROCNAM)
DEFINES OP AS A BRACKET OPERATOR WITH PRECEDENCE PREC AND ASSOCIATED PROCEDURE NAME PROCNAM. (SEE SEC. 5)

BREAKUP(PATRN,ARG)
BREAKUP PERFORMS A MULTIPLE ASSIGNMENT OF THE NAMES IN PATRN TO STRUCTURES IN ARG. FOR EXAMPLE
BREAKUP(≥(A [B C (D)]),≥(<STR> [57 (X) (Y)])
WILL CAUSE THE SAME ASSIGNMENTS AS A=<STR>;B=57;C=≥(X);D=≥Y;
IN GENERAL, THE NAMES IN PATRN ARE ASSIGNED VALUES WHICH ARE THE ELEMENTS APPEARING IN CORRESPONDING POSITIONS IN ARG. IF SUCH AN ASSIGNMENT CANNOT BE PERFORMED DUE TO A DIFFERENCE IN STRUCTURE BETWEEN PATRN AND ARG THEN NIL IS RETURNED, OTHERWISE TRUE. IN THE CASE OF AN ASSIGNMENT FAILURE, ONLY THOSE NAMES ENCOUNTERED BEFORE THE STRUCTURAL DISCREPANCY WILL BE ASSIGNED VALUES. FOR

EXAMPLE,
BREAKUP(≥(A (B C)),≥(X (3 4)))
WILL RETURN TRUE AND RESULT IN THE FOLLOWING ASSIGNMENTS:
A=≥X;B=3;C=4;
BREAKUP(≥(A(B C) D),≥(X (3 4 5))) WILL RETURN NIL AND CAUSE
THE SAME ASSIGNMENTS AS THE PREVIOUS EXAMPLE. D WILL REMAIN
UNCHANGED. CONSTANTS APPEARING IN PATRN MUST EXACTLY MATCH THEIR
CORRESPONDING ELEMENTS IN ARG FOR A SUCCESSFUL MATCH.
BREAKUP(≥(A B C <STR> 7),≥(1 2 3 <STR> 7)) WILL RETURN TRUE WHILE
BREAKUP(≥(A B C <STR> 7),≥(4 5 6 <STR> 8)) WILL RETURN NIL.
A CONSTANT NAME OR STRUCTURE CAN BE REPRESENTED IN PATRN BY
PRECEDING IT WITH A ≥ .
EXAMPLE:
BREAKUP(≥(A ≥B B ≥[C D]),≥(1 B 5 [C D])) WILL RETURN TRUE AND WILL
PERFORM THE ASSIGNMENTS A=1;B=5;
BREAKUP(≥(A ≥B C),≥(1 (B) <XYZ>)) WILL RETURN NIL AND WILL
PERFORM THE ASSIGNMENT A=1;

CAR(ARG)
SAME AS HD(SEE TEXT).

CARRIAGE(FLAG,TAPE,CHAR)
ENABLES CARRIAGE CONTROL ON FILE TAPE USING CHAR AS THE
CARRIAGE CONTROL CHARACTER. CHAR IS A SHORT-STRING CONTAINING A
SINGLE CHARACTER. EITHER CHAR OR CHAR AND TAPE MAY BE OMITTED.
DEFAULT VALUES FOR CHAR AND TAPE ARE ≠ ≠ AND THE STANDARD OUTPUT
FILE.

CDR(ARG)
SAME AS TL(SEE TEXT).

CHARACTER(NUMB)
RETURNS THE NUMBER NUMB CONVERTED TO A SHORT STRING.

CHARACTERP(ARG)
RETURNS TRUE IF ARG IS A SHORT STRING, OTHERWISE NIL.

CONC(ARG1,ARG2,...,ARGM,ARGN)
CONCATENATES ARG1,...,ARGN. ARG1,...,ARGM ARE MODIFIED.
ARG1,...,ARGN MUST BE EITHER LISTS OR STRINGS.

CONS(ARG1,ARG2)
SAME AS ARG1:ARG2(SEE TEXT).

CONSTRUCT(ARG)
RETURNS THE STRUCTURE ARG WITH ALL NAMES REPLACED BY THEIR VALUES
EXAMPLE:
BREAKUP(≥([A B] C 53),≥([(X Y) Z] R 53));
PRINT(CONSTRUCT(≥[(A B) C 65]));
WILL PRINT [((X Y) Z) R 65]

COPY(ARG)
RETURNS A COPY OF ITS ARGUMENT.

DELPROP(ARG1,PROP)
DELETES THE FIRST OCCURENCE OF PROPERTY PROP FROM NAME ARG1.

DELPROP RETURNS NIL IF THE PROPERTY IS NOT FOUND.

DIVIDE(NUMB1,NUMB2)
RETURNS A LIST WHOSE FIRST ELEMENT IS NUMB1/NUMB2 AND WHOSE SECOND ELEMENT IS THE REMAINDER.

ECHO(FLAG,TAPE)
ECHO WILL ENABLE OR DISABLE AUTOMATIC PRINTING OF LINES INPUT FROM THE FILE TAPE DEPENDING UPON WHETHER FLAG IS TRUE OR NIL. IF TAPE IS OMITTED, THEN THE STANDARD INPUT FILE IS ASSUMED.

EJECT(TAPE)
CAUSES A PAGE EJECT ON THE FILE TAPE. CARRIAGE CONTROL MUST BE ENABLED FOR THE FILE TAPE BEFORE EJECT IS CALLED. IF TAPE IS OMITTED, THEN THE STANDARD OUTPUT FILE IS ASSUMED.

ENDFILE(TAPE)
WRITES AN END OF FILE MARK ON THE FILE TAPE.

EQ(ARG1,ARG2)
RETURNS TRUE IF ARG1 AND ARG2 ARE THE SAME NAMES, OTHERWISE NIL.

EQUAL(ARG1,ARG2)
RETURNS TRUE IF ARG1 IS THE SAME AS (OR A COPY OF) ARG2.

ERROR(ARG)
PRINTS ARG, TERMINATES EXECUTION OF CURRENT STATEMENT, AND PROCEEDS TO THE NEXT TOP-LEVEL STATEMENT.

EVAL(ARG)
SAME AS $ (SEE TEXT)

EVLIS(LST)
RETURNS A LIST OF THE VALUES OBTAINED BY APPLYING EVAL TO EACH MEMBER OF LIST LST.

EXECUTE(INFILE,OUTFILE)
EXECUTE TRANSLATES AND EXECUTES BALM STATEMENTS FROM THE FILE INFILE UNTIL A STOP STATEMENT IS REACHED. EXECUTE WILL THEN RETURN NIL. TRANSLATOR ERROR MESSAGES ARE WRITTEN ON FILE OUTFILE.

EXIT()
TERMINATES BALM EXECUTION.

EXPAND(LST)
RETURNS THE RESULT OF A MACRO EXPANSION ON LIST LST. LST IS ASSUMED TO BE IN THE SAME FORM AS THE OUTPUT FROM THE PRECEDENCE ANALYZER. (SEE SEC. 5)

FILE(NAM,LINE,LBUFFER)
FILE CREATES AND RETURNS A FILE. THE SCOPE NAME OF THE FILE IS SPECIFIED BY NAM, WHICH IS A SHORT-STRING OF LESS THAN 8 CHARACTERS, THE FIRST BEING ALPHABETIC. LINE IS A NUMBER SPECIFYING THE NUMBER OF CHARACTERS PER LINE. LBUFFER IS A NUMBER SPECIFYING THE LENGTH OF THE I/O BUFFER FOR THE FILE.

ECHOING AND CARRIAGE CONTROL ARE INITIALLY DISABLED. INDENTED PRINTING IS INITIALLY ENABLED. FILE(NAM,LINE) AND FILE(NAM) ARE ALTERNATE FORMS OF A CALL TO FILE. THE DEFAULT VALUES OF LINE AND LBUFFER ARE 72 AND 200 (OCTAL) RESPECTIVELY.

FIXP(NUMB)
RETURNS NIL IF NUMB IS A REAL NUMBER, OTHERWISE TRUE.

FLOATP(NUMB)
RETURNS TRUE IF NUMB IS A REAL NUMBER, OTHERWISE NIL.

FLUSH(TAPE)
THE CURRENT LINE OF FILE TAPE IS OUTPUTTED. REPEATED CALLS TO FLUSH WILL PRINT BLANK LINES. IF TAPE IS OMITTED THEN THE STANDARD OUTPUT FILE IS ASSUMED.

GENSYM()
RETURNS A UNIQUE NAME EACH TIME CALLED.

GO(ARG)
IN A PROCEDURE, TRANSFERS TO THE STATEMENT WHOSE LABEL IS ARG.

INFIX(OP,LPREC,RPREC,PROCNAM)
DEFINES INFIX OPERATOR OP WITH LEFT PRECEDENCE LPREC AND RIGHT PRECEDENCE RPREC. PROCNAM IS THE NAME OF THE CORRESPONDING PROCEDURE. (SEE SEC. 5)

INTEGER(NUMB)
CONVERTS NUMBER NUMB TO AN INTEGER.

LAST(LIS)
RETURNS THE LAST TOP-LEVEL ARGUMENT OF LIST LIS.

LEFTSHIFT(NUMB,CNT)
RETURNS NUMB SHIFTED LEFT CNT BITS. IF CNT IS NEGATIVE, THEN A RIGHT SHIFT IS PERFORMED.

LENGTH(ARG)
RETURNS THE LENGTH OF A STRING IN CHARACTERS OR THE NUMBER OF TOP LEVEL ELEMENTS IN A LIST OR VECTOR.

LINES(N,TAPE)
SKIPS N LINES ON OUTPUT FILE TAPE. CARRIAGE CONTROL MUST BE ENABLED FOR THE FILE TAPE BEFORE CALLING LINES. IF TAPE IS OMITTED THE STANDARD OUTPUT FILE IS ASSUMED.

LIST(ARG1,ARG2,...,ARGN)
RETURNS A LIST OF ARG1,ARG2,...,ARGN.
(IE. ARG1:ARG2:...:ARGN:NIL)

LISTC(ARG)
RETURNS ARG CONVERTED TO A LIST. IF ARG IS A VECTOR,THEN THE RESULTING LIST WILL CONSIST OF THE ELEMENTS OF ARG. IF ARG IS A STRING, THEN THE RESULT WILL BE A LIST OF SHORT STRINGS, EACH OF WHICH IS A SINGLE CHARACTER OF ARG. IF ARG IS A LIST, THEN IT IS RETURNED.

LMACRO(NAM,PROCED)
DEFINES NAM AS A MACRO TO BE USED ON THE LEFT-HAND SIDE OF AN ASSIGNMENT STATEMENT. PROCED IS THE ASSOCIATED PROCEDURE. (SEE SEC. 5)

LOC(ARG)
RETURNS A POINTER TO A WORD WHICH CONTAINS A POINTER TO ARG.

LOGAND(NUMB1,NUMB2,...,NUMBN)
RETURNS THE LOGICAL PRODUCT OF NUMB1 THROUGH NUMBN.

LOGOR(NUMB1,NUMB2,...,NUMBN)
RETURNS THE LOGICAL SUM OF NUMB1 THROUGH NUMBN.

LOGXOR(NUMB1,NUMB2,...,NUMBN)
RETURNS THE LOGICAL DIFFERENCE OF NUMB1 THROUGH NUMBN.

MACRO(NAM,PROCED)
DEFINES NAM AS A MACRO WITH ASSOCIATED PROCEDURE PROCED. (SEE SEC. 5)

MAKBLOCK(NUMB)
RETURNS A VECTOR OF LENGTH NUMB.

MAP(LST,PROCED)
MAP APPLIES THE PROCEDURE PROCED TO THE LIST LST, THEN CDR(LST), THEN CDR(CDR(LST)), AND SO ON, UNTIL LST IS EXHAUSTED. NIL IS RETURNED.

MAPCON(LST,PROCED)
MAPCON APPLIES THE PROCEDURE PROCED TO THE LIST LST, THEN CDR (LST), THEN CDR(CDR(LST)), AND SO ON, UNTIL LST IS EXHAUSTED. THE RESULTS OF THESE APPLICATIONS OF PROCED ARE CONCATENATED (APPENDED) AND RETURNED. NOTE THAT PROCED MUST RETURN EITHER A LIST OR A STRING, SINCE THESE ARE THE ONLY STRUCTURES FOR WHICH CONCATENATION IS DEFINED.

MAPLIST(LST,PROCED)
MAPLIST APPLIES THE PROCEDURE PROCED TO THE LIST LST, THEN CDR (LST), THEN CDR(CDR(LST)), AND SO ON, UNTIL LST IS EXHAUSTED. A LIST CONSISTING OF THE RESULTS OF THESE APPLICATIONS OF PROCED IS RETURNED.
EXAMPLE:
Z=MAPLIST(≥(A B C D),PROC(X),LIST(CAR X,≥X) END); PRINT(Z);
WILL PRINT ((A X)(B X)(C X)(D X))

MAPX(ARG,PROCED)
MAPX APPLIES THE PROCEDURE PROCED TO EACH OF THE TOP LEVEL ELEMENTS OF ARG, RETURNING A LIST OR VECTOR OF THE RESULTS. IF ARG IS A LIST, THEN A LIST IS RETURNED. IF ARG IS A VECTOR, THEN A VECTOR IS RETURNED. ARG MAY NOT BE A STRING.

MAX(NUMB1,NUMB2,...,NUMBN)
RETURNS THE LARGEST OF NUMBERS NUMB1 THROUGH NUMBN.

MEMBER(ARG1,ARG2)
IF ARG1 IS A TOP LEVEL ELEMENT OF ARG2 (A LIST OR VECTOR) THEN
TRUE IS RETURNED, OTHERWISE NIL.

MIN(NUMB1,NUMB2,...,NUMBN)
RETURNS THE SMALLEST OF NUMBERS NUMB1 THROUGH NUMBN.

MINUSP(NUMB)
RETURNS TRUE IF NUMB IS NEGATIVE, OTHERWISE NIL.

MKNUM(CHAR)
MKNUM RETURNS THE INTEGER WHOSE VALUE IS SPECIFIED BY THE SHORT
STRING CHAR. CHAR MUST CONTAIN A SINGLE NUMERIC CHARACTER.

MKVAR(STRNG)
MKVAR RETURNS THE VARIABLE (SYMBOL) WHOSE NAME IS SPECIFIED BY
THE CHARACTERS OF STRNG.

NULL(ARG)
RETURNS TRUE IF ARG IS NIL, OTHERWISE NIL.

NUMBERP(ARG)
RETURNS TRUE IF ARG IS A NUMBER (REAL, INTEGER, OR OCTAL)
OTHERWISE NIL.

NUMCHARP(CHAR)
RETURNS TRUE IF THE SHORT-STRING CHAR IS A SINGLE NUMERICAL
CHARACTER (0-9) , OTHERWISE NIL.

OCTAL(NUMB)
RETURNS THE NUMBER NUMB, CONVERTED TO OCTAL.

OCTALP(ARG)
RETURNS TRUE IF ARG IS A NUMBER OF TYPE OCTAL.

OPCHAR(CHAR)
RETURNS TRUE IF THE SHORT-STRING CHAR IS AN OPERATOR CHARACTER.
OTHERWISE NIL.
THE OPERATOR CHARACTERS ARE
; $ / ( ) , . + - * = → [ ] : ≠ → ∨ ∧ ↑ ↓ < > ≤ ≥ ¬

PACKSTRING(ARG)
RETURNS THE PACKED STRING EQUIVALENT TO ARG.A PACKED STRING TAKES
UP THE MINIMUM STORAGE SPACE. PACKSTRING WILL ALSO CONVERT A
LIST OF SHORT STRINGS TO A STRING.

PRETTYPRINT(FLAG,TAPE)
ENABLES OR DISABLES INDENTED OUTPUT OF LINES ON FILE TAPE
DEPENDING UPON WHETHER FLAG IS TRUE OR NIL. IF TAPE IS OMITTED
THEN THE STANDARD OUTPUT FILE IS USED.

PRINT(ARG)
ARG IS PRINTED AFTER ANYTHING IN THE PRINT BUFFER, THE PRINT
BUFFER IS FLUSHED, AND ARG IS RETURNED.

PRINTNF(ARG)

THE PRINTED REPRESENTATION OF ARG IS ENTERED INTO THE PRINT BUFFER. THE PRINTER IS NOT SENT TO THE NEXT LINE. ARG IS RETURNED.

PRINTPROP(NAME)
PRINTPROP PRINTS THE PROPERTIES OF NAME, FOLLOWED BY A STRING CONTAINING THE NAME.

READ(TAPE)
READS AND RETURNS THE NEXT LINE OF THE FILE TAPE. IF TAPE IS OMITTED THEN THE STANDARD INPUT FILE IS ASSUMED.

READLINE(TAPE)
READS A LINE FROM THE FILE TAPE AND FORMS AN UNPACKED STRING FROM IT. THE STRING (WHICH IS RETURNED) IS OF LENGTH EQUAL TO THE LINE LENGTH OF FILE TAPE. IF TAPE IS OMITTED, THE STANDARD INPUT FILE IS ASSUMED.

REAL(NUMB)
RETURNS NUMBER NUMB CONVERTED TO A NUMBER OF TYPE REAL.

RECLAIM()
FORCES A GARBAGE COLLECTION AND RETURNS NIL.

REMAINDER(NUMB1,NUMB2)
RETURNS THE REMAINDER OF A DIVISION OF NUMB1 BY NUMB2.

REMOB(NAM)
FREES THE SPACE OCCUPIED BY THE VARIABLE (NAME) NAM.

RETURN(ARG)
RETURNS ARG AS THE VALUE OF A BEGIN-END BLOCK.

REVERSE(LST)
RETURNS THE LIST LST REVERSED ON ITS TOP LEVEL.

REWIND(TAPE)
REWINDS THE FILE TAPE.

RPLACA(LST1,ARG2)
REPLACES THE HEAD OF LIST LST1 WITH ARG2. RPLACA RETURNS THE NEW LST1 (I.E. RETURNS ARG2:TL LST1). LST1 IS MODIFYED.
EXAMPLE:

```
X=≥(A B C D);
PRINT(RPLACA(X,≥(D E)));
PRINT(X);
```

WOULD PRINT≥

```
((D E) B C D)
((D E) B C D)
```

RPLACD(LST1,ARG2)
REPLACES THE TAIL OF LIST LST1 WITH ARG2. RPLACD RETURNS THE NEW LST1 (I.E. RETURNS (HD LST1):ARG2). LST1 IS MODIFYED.
EXAMPLE:

```
X=≥(A B C D);
PRINT(RPLACD(X,≥(D E)));
```

```
        PRINT(X);
    WOULD PRINT≥
        (A D E)
        (A D E)

SETPROP(NAME,PROP,VAL)
    GIVES THE PROPERTY PROP WITH VALUE VAL TO THE NAME NAME.  THE
    PREVIOUS VALUE OF THIS PROPERTY (IF PRESENT) IS PUSHED DOWN, AND
    NOT LOST.  THE NAME IS RETURNED.

SETPROPD(NAME,PROP,VAL)
    GIVES THE PROPERTY PROP WITH VALUE VAL TO THE NAME NAME.  THE
    PREVIOUS VALUE OF THIS PROPERTY (IF PRESENT) IS DESTROYED.  IF A
    PREVIOUS VALUE, PVAL, IS PRESENT, THEN PROP:PVAL IS RETURNED,
    OTHERWISE NIL.

STRING(STR1,STR2,...,STRN)
    RETURNS STR1,STR2,...,STRN CONCATENATED.
    STR1,STR2,...,STRN MUST BE EITHER STRINGS OR LISTS OF SHORT
    STRINGS.  A STRING IS RETURNED.  FOR EXAMPLE:
    STRING(<A STRING>,< AND >,<ANOTHER>)   RETURNS
    <A STRING AND ANOTHER>

STRINGP(ARG)
    RETURNS TRUE IF ARG IS A STRING, NIL OTHERWISE.

SUBST(NEW,OLD,LST)
    RETURNS THE LIST LST WITH ALL OCCURENCES OF OLD REPLACED BY NEW.
    FOR EXAMPLE X=SUBST(≥[X Y],≥(A(B C)),≥(X Y ((A (B C))) Z);
    PRINT(X);
    WILL PRINT (X Y ([X Y]) Z)

SUBSTR(STR,N,LEN)
    RETURNS THE SUBSTRING OF STRING STR STARTING FROM THE N TH
    CHARACTER AND CONTINUING FOR LEN CHARACTERS.  IF LEN IS ZERO OR
    IF N IS GREATER THAN THE LENGTH OF STR, THEN THE NULL STRING <>
    IS RETURNED.  IF LEN IS NEGATIVE OR IF N+LEN IS GREATER THAN THE
    LENGTH OF STR, THEN A STRING FORMED FROM THE NTH CHARACTER UNTIL
    THE END OF STR IS RETURNED.
    EXAMPLES:
    PRINT(SUBSTR(<ABCDE>,2,3));   PRINTS <BCD>
    PRINT(SUBSTR(<ABCDE>,2,-1));  PRINTS <BCDE>
    PRINT(SUBSTR(<ABCDE>,3,0));   PRINTS <>

TERPRI()
    WILL PRINT AND FLUSH THE PRINT BUFFER, AND RETURN NIL.

TIME()
    RETURNS THE ELAPSED CP TIME IN SECONDS SINCE THE START OF THE
    JOB AS A REAL NUMBER.

TRANSLATE(LST)
    LST IS A LIST CONTAINING A BALM STATEMENT WITHOUT THE
    TERMINATING SEMI-COLON.  TRANSLATE WILL RETURN THE STATEMENT
    TRANSLATED INTO BALM INTERNAL FORM, SUITABLE FOR EVALUATION BY
    EVAL ($) .  (SEE SEC. 5)
```

UNARY(OP,PREC,PROCNAM)
DEFINES OP AS A UNARY OPERATOR WITH PRECEDENCE PREC AND ASSOCIATED PROCEDURE NAME PROCNAM. (SEE SEC. 5)

VARNAME(VAR)
RETURNS A STRING WHOSE CHARACTERS SPECIFY THE NAME OF THE VARIABLE VAR.

VECTOR(ARG1,ARG2,...,ARGN)
RETURNS A VECTOR CONTAINING ARG1,ARG2,...,ARGN.

VERBOS(ARG)
IF ARG IS TRUE, SUBSEQUENT GARBAGE COLLECTIONS WILL PRODUCE A MESSAGE ON THE STANDARD OUTPUT FILE. IF ARG IS NIL, THEN THESE MESSAGES ARE SUSPENDED. THE VALUE OF ARG IN THE PREVIOUS CALL IS RETURNED.

WRITE(ARG,TAPE)
INTRODUCES ARG INTO THE CURRENT LINE OF FILE TAPE AND WRITES THE CURRENT LINE. IF TAPE IS OMITTED, THEN THE STANDARD OUTPUT FILE IS ASSUMED.

WRITELINE(STRNG,TAPE)
WRITES THE CHARACTERS OF STRNG ON FILE TAPE AS A SINGLE LINE. THE LENGTH OF STRNG MUST BE LESS THAN OR EQUAL TO THE LINE LENGTH OF FILE TAPE. IF TAPE IS OMITTED, THEN THE STANDARD OUTPUT FILE IS ASSUMED.

WRITENF(ARG,TAPE)
ARG IS INTRODUCED INTO THE CURRENT LINE OF FILE TAPE. THE CURRENT LINE IS NOT WRITTEN. IF TAPE IS OMITTED THEN THE STANDARD OUTPUT FILE IS ASSUMED.

ZEROP(NUMB)
RETURNS TRUE IF NUMB IS ZERO, OTHERWISE NIL.

9. BALM CONTROL CARD

THE PARAMETER LIST OF THE BALM CONTROL CARD IS IN A POSITION INDEPENDENT FORMAT SIMILAR TO THE COMPASS AND UPDATE CONTROL CARDS. IT IS AS FOLLOWS:

```
BALM(P1,P2,...,PN)
```

WHERE THE PI ARE ANY OF THE FOLLOWING PARAMETERS:

I=FNAM1	FNAM1 IS THE NAME OF THE FILE TO BE USED FOR INPUT OF COMMANDS. IT IS THE STANDARD INPUT FILE.
O=FNAM2	FNAM2 IS THE NAME OF THE FILE TO BE USED FOR BALM OUTPUT. IT IS THE STANDARD OUTPUT FILE.
T3=FNAM3	WHERE FNAM3 IS THE NAME OF THE FILE TO BE USED WITHIN BALM AS TAPE 3.
T4=FNAM4	FNAM4 IS THE NAME OF THE FILE TO BE USED WITHIN BALM AS TAPE 4
FS=P1	P1 IS AN INTEGER REPRESENTING THE PERCENTAGE OF AVAILABLE SPACE (FIELD LENGTH MINUS LENGTH OF BALM SYSTEM) TO BE USED AS FREE SPACE (LIST, STRING AND NUMBER STORAGE)
ST=P2	P2 IS AN INTEGER REPRESENTING THE PERCENTAGE OF AVAILABLE SPACE TO BE USED AS A STACK.
TTY	TAPE 1 AND TAPE 2 GO TO A TELETYPE. SHOULD NOT BE PRESENT FOR BATCH PROCESSING.

DEFAULT VALUES ARE ASSUMED FOR MISSING PARAMETERS. THEY ARE

```
I=INPUT,O=OUTPUT,FS=63,ST=20
```

PARAMETERS MAY NOT BE REPEATED.

10. NOTES ON CURRENT STATUS

A FL OF 60000 IS SUFFICIENT FOR MOST BALM APPLICATIONS

IF YOU HAVE INFALLIBLE EVIDENCE OF BUGS, REPORT THEM TO ROOM 329.

THE SCOPE VERSION IS IN OVERLAY FORM, WITHOUT LOADER TABLES, SO USE OF ENTRY-POINTS ON INPUT WILL NOT WORK.

COME TO ROOM 329 IF YOU WANT THE COMPILER.