COMMON LISP/CORE
--- a Common Lisp subset proposal ---

This proposal is the joint work of a collaboration with the following members.

Katsuhiko Yuura	(Hitachi, Ltd.)
Hideki Kato	(Fujitsu Laboratories Ltd.)
Yukiko Hashimoto	(NEC Corp.)
Kazusaku Kawagome	(SORD Computer Corp.)
Susumu Kawai	(Nihon Digital Equipment Corp.)
Shigeaki Harada	(Sharp Corp.)
Yoichi Yamamura	(Nippon UNIVAC Kaisha Ltd.)
Takeshi Shimizu	(Fuji Xerox Co., Ltd.)
Takashi Hamada	(Matsushita Electric Industrial Co., Ltd.)
Haruyuki Kawabe	(Nippon UNIVAC Kaisha Ltd.)
Akio Tanaka	(Toshiba Corp.)
Nobuyuki Saji	(NEC Corp.)
Masayuki Ida	(Aoyama Gakuin Univ.)

1. Goals of Common Lisp/Core

(1) To set up the standard specification for small sized personal computers and hand-held computers

Most of Japanese Lisp users on personal computers do not/will not use all functions of the full set Common Lisp, and they hope for good performance strongly. The authors want to propose a paying subset, which does not have functions that are seldom used and functions that make the system inefficient.

(2) To set up basic level of Common Lisp

Common Lisp should have two levels; one is full set which is growing up and the other is subset which is fixed.

2. Discussions for Common Lisp/Core

(1) As members of subset WG in Jeida Common Lisp Committee, the authors have discussed as to the following subjects since Dec. 10 1985.

i) a review of Ida's personal proposal for a subset.
ii) an examination of the necessities and the difficulties to implement each function.
iii) a decision on the basic issues for Common Lisp/Core.
iv) a choice of the functions based on the vote by the members.

(2) The authors had an open meeting for the announcement and the discussion of Common Lisp/Core on Jul. 8 '86. About twenty lisp and programming language researchers and about forty lisp users met and commented as follows.

i) From the implementor's point of view, it is supposed, the scale of Common Lisp/Core is not so much smaller than that of the full set.
ii) From the user's point of view, useful functions are selected all over sections and it is expected that more functions are selected from packages, streams or declarations.

3. Basic issues and decisions

(1) Arms and legs of Common Lisp are kept, because it is important to transfer programs in the subset to these in the full set easily, and it is also expected that the subset users grow into the full set users easily.

(2) It is aimed that the number of functions in Common Lisp/Core is about a half in the full set.

(3) The following features of Common Lisp are kept:

i) scope and extent rules including lexical closure.
ii) keyword parameters.
iii) the principles of type hierarchy and generic function.
iv) functional richness over Utilisp (Tokyo univ. '81; one of the most famous Lisp in Japan) or Franzlisp.
v) some useful or characteristic data types:
bignum, ratio, structure and readtable

(4) The choice of functions is based on the following rules in order.

i) functions related to the features (3).
ii) functions having high necessities.
iii) functions not having so many difficulties to implement

The following issues are also discussed, but these policies are not adopted.

(a) To keep the "language" oriented features and to leave out the "system" oriented features.

The authors think that Common Lisp features are classified into two categories; one is the "language" oriented features (e.g. control, number and list), the other is the "system" oriented features (e.g. package, stream and I/O).

(b) To keep the kernel part that users for themselves can not "defun" or "defmacro".

Some members of this WG opposed to the issues (3) and (4), argued for the Common Lisp/Kernel approach (b), and then they did not join this proposal.

4. Summary of Common Lisp/Core

Major deleted items are: most of system parameter constants, complex numbers, most of package features, local functions, adjustable arrays, hash-tables, and pathnames.

Common Lisp/Core summary

Notes: o...remained x...deleted

Chapter	Comment	Number of Func/Const Core	CLtL
2. Data Types	o type hierarchy x complex, pathname o ratio, structure	-	-
3. Scope and Extent	same as CLtL o lexical scoping	-	-
4. Type Specifiers	x complex type specifiers x deftype	2/-	3/-

5.	Program Structure	x	defparameter	4/0	5/2
6.	Predicates	x	equalp, complexp	26/2	33/2
7.	Control Structure	x	prog, prog*, prog2	49/0	67/2
		x	flet, labels, macrolet		
		o	tagbody, go		
8.	Macro			4/0	4/1
9.	Declarations	x	all declaration specifiers except "special"	2/-	4/-
		x	proclaim, locally		
10.	Symbols	x	getf, remf	9/-	13/-
11.	Packages	x	almost all	4/0	26/2
		o	intern, unintern, find-symbol, do-all-symbols		
12.	Numbers	x	complex numbers	49/1	96/44
		o	ratio, pi		
13.	Characters	x	bits, font attributes	28/0	36/7
14.	Sequences	x	xxx-if, -if-not	22/-	44/-
15.	Lists	x	c....r, xxx-if, -if-not	57/-	94/-
16.	Hash Tables	x	all	0/-	8/-
17.	Arrays	Only simple-arrays are available		7/0	31/3
		x	adjustable array, fill-pointer		
18.	Strings	o	almost all	25/-	25/-
19.	Structures	o	defstruct	1/-	1/-
20.	The Evaluator	x	hooks	2/4	4/12
21.	Streams	x	make-xx-stream functions	5/7	16/7
		x	:abort of "close"		
22.	Input/Output	x	write, y-or-n-p	28/4	41/14
23.	File System Interface	Pathname is a string		10/0	29/2
24.	Errors	o	error, warn, break	3/0	10/1
25.	Miscellaneous Features			19/2	32/2
			Total	356/20	622/101

5. Difference from Ida's personal proposal

(1) added features in Common Lisp/Core
keyword parameters, readtables, tagbody, go,
character predicates ignoring case, string functions and so on

(2) deleted features in Common Lisp/Core
list functions, file i/o functions and so on

(3) number of functions, variables and constant
Common Lisp/Core 376 (356,17,3)
Ida's personal proposal 341 (332, 7,2)

The authors will appreciate opinions from all the persons through network.

```
1   ------------------------------------------------------------------------------
2   2. Data Types
3   2.1 Numbers
4        NUMBER
5        INTEGER
6        FIXNUM
7        BIGNUM
8        RATIONAL
9        RATIO
10       FLOAT (SHORT-/SINGLE-/DOUBLE-/LONG-)
11   -- Deleted Item ----------------------------------------------------------
12       COMPLEX
13  ------------------------------------------------------------------------------
14  2.2 Characters
15       CHARACTER
16       STANDARD-CHARACTER
17       STRING-CHARACTER
18   -- Comment ----------------------------------------------------------------
19       -> bits-attribute and font-attribute may be zero.
20  ------------------------------------------------------------------------------
21  2.3 Symbols
22       SYMBOL
23  ------------------------------------------------------------------------------
24  2.4 Lists and Conses
25       LIST
26       CONS
27       NULL
28  ------------------------------------------------------------------------------
29  2.5 Arrays
30       ARRAY
31       SIMPLE-ARRAY
32       VECTOR
33       SIMPLE-VECTOR
34       STRING
35       SIMPLE-STRING
36   -- Deleted Item ----------------------------------------------------------
37       BIT-VECTOR
38   -- Comment ----------------------------------------------------------------
39       -> ARRAY means SIMPLE-ARRAY, and rank is restricted to max 3.
40  ------------------------------------------------------------------------------
41  2.6 Hash Tables
42   -- Deleted Item ----------------------------------------------------------
43       HASHTABLES
44  ------------------------------------------------------------------------------
45  2.7 Readtables
46       READTABLE
47  ------------------------------------------------------------------------------
48  2.8 Packages
49       PACKAGE
50   -- Comment ----------------------------------------------------------------
51       -> Package is restricted to LISP package and KEYWORD package.
52  ------------------------------------------------------------------------------
53  2.9 Pathnames
54   -- Deleted Item ----------------------------------------------------------
55       PATHNAME
56  ------------------------------------------------------------------------------
57  2.10 Streams
58       STREAM
59  ------------------------------------------------------------------------------
60  2.11 Random-States
61   -- Deleted Item ----------------------------------------------------------
62       RANDOM-STATES
63  ------------------------------------------------------------------------------
64  2.12 Structures
65       STRUCTURE
66  ------------------------------------------------------------------------------
67  2.13 Functions
68       FUNCTION
69       COMPILED-FUNCTION
70       LAMBDA-EXPRESSION
71       SYMBOL
72  ------------------------------------------------------------------------------
73  3. Scope and Extent
74       Same as CLtL
75   -- Comment ----------------------------------------------------------------
76       -> Modern Lisp must have the compiler.
77       -> The compatibility of compiler and interpreter is one of the
78          major goals of CL.
79       -> We evaluate the effort to make the semantics of Lisp clearer.
80  4 Type Specifiers
81  4.1 Type Specifier Symbols
82       The type symbols are the same as data types.
83  ------------------------------------------------------------------------------
84  4.2 Type Specifier Lists
85       Type specifier lists are allowed.
86  ------------------------------------------------------------------------------
87  4.3 Predicating Type Specifiers
88   --Deleted item ----------------------------------------------------------
89       SATISFIES predicate-name
90  ------------------------------------------------------------------------------
91  4.4 Type Specifiers That Combine
92       Type specifier combination is omitted.
93  ------------------------------------------------------------------------------
94  4.5 Type Specifier That Specialize
95       Type specifier specializations are omitted.
96  ------------------------------------------------------------------------------
97  4.6 Type Specifiers That Abbreviate
98       Type specifier abbreviations are omitted.
99  ------------------------------------------------------------------------------
100 4.7 Defining New Type Specifiers
```

```
 -- Deleted item ----------------------------------------------------------------
     DEFTYPE name lambda-list (declaration|doc-string)* (form)*          [Macro]
 ---------------------------------------------------------------------------------
 4.8 Type Conversion Function
     COERCE object result-type                                          [Function]
 ---------------------------------------------------------------------------------
 4.9 Determining the Type of an Object
     TYPE-OF object                                                     [Function]
 ---------------------------------------------------------------------------------
 5. Program Structure
 ---------------------------------------------------------------------------------
 5.1 Forms
   Same as CLtL except absence of some special forms.
 -- Deleted items ----------------------------------------------------------------
    Following special forms:
        PROGV
        COMPILER-LET
        FLET
        LABELS
        MACROLET
 ---------------------------------------------------------------------------------
 5.2 Functions
 -- Deleted items ----------------------------------------------------------------
    LAMBDA-LIST-KEYWORDS                                                [Constant]
    LAMBDA-PARAMETERS-LIST                                              [Constant]
    --> to keep the system compact.
 ---------------------------------------------------------------------------------
 5.3 Top-level Forms
    DEFUN name lambda-list {declaration | doc-string}* (form)*           [Macro]
    DEFVAR name [initial-value [documentation]]                          [Macro]
    DEFCONSTANT name initial-value [documentation]                       [Macro]
    EVAL-WHEN ((situation)*) (form)*                                [Special Form]
 -- Deleted items ----------------------------------------------------------------
   DEFPARAMETER name initial-value [documentation]                       [Macro]
   --> It is redundant.
 ---------------------------------------------------------------------------------
 6. Predicates
 ---------------------------------------------------------------------------------
 6.1. Logical Values
    NIL                                                                 [Constant]
    T                                                                   [Constant]
 ---------------------------------------------------------------------------------
 6.2. Data Type Predicates
 6.2.1. General Type Predicates
    TYPEP object type                                                   [Function]
    SUBTYPEP type1 type2                                                [Function]
 6.2.2. Specific Data Type Predicates
    NULL object                                                         [Function]
    SYMBOLP object                                                      [Function]
    ATOM object                                                         [Function]
    CONSP object                                                        [Function]
    LISTP object                                                        [Function]
    NUMBERP object                                                      [Function]
    INTEGERP object                                                     [Function]
    RATIONALP object                                                    [Function]
    FLOATP object                                                       [Function]
    STRINGP object                                                      [Function]
    VECTORP object                                                      [Function]
    SIMPLE-VECTOR-P object                                              [Function]
    SIMPLE-STRING-P object                                              [Function]
    ARRAYP object                                                       [Function]
    FUNCTIONP object                                                    [Function]
    COMPILED-FUNCTION-P object                                          [Function]
    STREAMP object                                                      [Function]
    COMMONP object                                                      [Function]
    -- see also ...
    STANDARD-CHAR-P, STRING-CHAR-P, READTABLEP
 -- Comment ----------------------------------------------------------------------
    Data type predicates' existence is depending on data type existence
       itself.
    Data type hierarchy remains.
    There is a recommend with using following predicates.
        VECTORP, STRINGP
    Use SIMPLE-xxx functions of these is much better, because data type ARRAY
        has no fill-pointer, is not adjustable, is not able to be displaced
        from another array.

 -- Deleted items ----------------------------------------------------------------
    BIT-VECTOR-P object                                                 [Function]
    SIMPLE-BIT-VECTOR-P object                                          [Function]
    RANDOM-STATE-P object                                               [Function]
    HASH-TABLE-P object                                                 [Function]
    PATHNAMEP object                                                    [Function]
    COMPLEXP object                                                     [Function]
    PACKAGEP object                                                     [Function]
    -> These functions are deleted because of deleting of their data types.
 ---------------------------------------------------------------------------------
 6.3. Equality Predicates
    EQ x y                                                              [Function]
    EQL x y                                                             [Function]
    EQUAL x y                                                           [Function]
 -- Deleted item ----------------------------------------------------------------
    EQUALP x y                                                          [Function]
    -> Other function can replace this. Use type specific function to compare
       rougher or more exactly.
 ---------------------------------------------------------------------------------
 6.4. Logical Operators
    NOT x                                                               [Function]
    AND (form)*                                                           [Macro]
    OR (form)*                                                            [Macro]
```

```
7 Control Structure
-------------------------------------------------------------------------------
7.1 Constants and Variables
7.1.1 Reference
     QUOTE object
     FUNCTION fn                                                  [Special Form]
     SYMBOL-VALUE symbol                                          [Special Form]
     SYMBOL-FUNCTION symbol                                           [Function]
     BOUNDP symbol                                                    [Function]
     FBOUNDP symbol                                                   [Function]
     SPECIAL-FORM-P symbol                                            [Function]
7.1.2 Assignment
     SETQ {var form}*                                             [Special Form]
     PSETQ {var form}*                                                    [Macro]
     SET symbol value                                                 [Function]
     MAKUNBOUND symbol                                                [Function]
     FMAKUNBOUND symbol                                               [Function]
 -- Comments ----------------------------------------------------------------
     -> These are primitive.
     -> FUNCTION is very important, because it returns "lexical closure".
-------------------------------------------------------------------------------
7.2 Generalized Variables
     SETF {place newvalue}*
     -> This is very important macro on CL.                               [Macro]
 -- Deleted Items -----------------------------------------------------------
     PSETF {place newvalue}*                                              [Macro]
     SHIFTF {place}+ newvalue                                             [Macro]
     ROTATEF {place}+                                                     [Macro]
     -> If necessary, these macros can be defined easily using SETF macro.
     DEFINE-MODIFY-MACRO name lambda-list function [doc-string]           [Macro]
     DEFSETF access-fn {update-fn [doc-string] |                          [Macro]
                        lambda-list (store-variable)
                        {declaration | doc-string}* {form}*)
     DEFINE-SETF-METHOD access-fn lambda-list                             [Macro]
                        {declaration | doc-string}* (form)*
     GET-SETF-METHOD form                                             [Function]
     GET-SETF-METHOD-MULTIPLE-VALUE form                              [Function]
     -> These macros and functions are provided to define macros
        like a SETF and to modify SETF.
 -- Forms of Place ----------------------------------------------------------
     1) The name of a variable.
     2) Functions.
          AREF              NTH               ELT
          REST              FIRST             SECOND
          THIRD             FOURTH            FIFTH
          SIXTH             SEVENTH           EIGHTH
          NINTH             TENTH             CAR
          CDR               C..R              C...R
          SVREF             GET               SYMBOL-PLIST
          SYMBOL-VALUE      SYMBOL-FUNCTION   MACRO-FUNCTION
     3) Selector function constructed by DEFSTRUCT.
     4) Functions.
          CHAR              SCHAR             SUBSEQ
     5) A THE type declaration form.
     6) A call to APPLY where the first argument form is of the form #'name.
     7) A macro call.
-- Deleted Forms of Place ---------------------------------------------------
     1) Functions.
          C....R            GETF
          GETHASH           DOCUMENTATION     FILL-POINTER
          BIT               SBIT              CHAR-BIT
          LDB               MASK-FIELD
     2) Any form for which a DEFSETF or DEFINE-SETF-METHOD declaration
        has been made.
     -> These functions are deleted on CL Core.
-------------------------------------------------------------------------------
7.3 Function Invocation
     APPLY function arg &rest more-args                               [Function]
     FUNCALL fn &rest argument                                        [Function]
     -> These functions are very primitive in LISP language.
 -- Deleted Item ------------------------------------------------------------
     CALL-ARGUMENT-LIMIT                                              [Constant]
-------------------------------------------------------------------------------
7.4 Simple Sequencing
     PROGN {form}*                                                [Special Form]
     PROG1 first {form}*                                                  [Macro]
 -- Deleted Item ------------------------------------------------------------
     PROG2 first second {form}*                                           [Macro]
     -> PROG2 is provided mostly for historical compatibility.
-------------------------------------------------------------------------------
7.5 Establishing New Variable Bindings
     LET ({var | (var value)}*) {declaration}* {form}*           [Special Form]
     LET* ({var | (var value)}*) {declaration}* {form}*          [Special Form]
     -> These are very important to establish new variable binding.
 -- Deleted Items -----------------------------------------------------------
     COMPILER-LET ({var | (var value)}*) {form}*                  [Special Form]
     PROGV symbols values {form}*                                 [Special Form]
     -> These are not important to ordinary users.
     FLET (((name lambda-list {declaration | doc-string}          [Special Form]
                              {form}*)}*) {form}*
     LABELS (((name lambda-list {declaration | doc-string}        [Special Form]
                                {form}*)}*) {form}*
     MACROLET (((name varlist {declaration | doc-string}          [Special Form]
                              {form}*)}*) {form}*
     -> In ordinary, local named functions and macros are not necessary.
7.6 Conditionals
     IF test then else                                            [Special Form]
     WHEN test {form}*                                                    [Macro]
     UNLESS test {form}*                                                  [Macro]
     COND ({test {form}*})*                                               [Macro]
```

```
      CASE keyform {{{{(key)*) | key}(form)*)}
  -- Deleted items ------------------------------------------------------------ [Macro]
      TYPECASE  keyform  {(type  (form)*)}*
  -- Comments ----------------------------------------------------------------- [Macro]
      -> #'IF is the primitive of the CL.
      -> #'COND is a  one  of  the  originator  of lisp.
         It is constructive but not harmful.
 ------------------------------------------------------------------------------
 7.7 Blocks and Exits
      BLOCK name (form)*
      RETURN-FROM name [result]                                    [Special Form]
      RETURN [result]                                              [Special Form]
  -- Comments ----------------------------------------------------------------- [Macro]
      -> These features are quite necessary for constructive programming.
 ------------------------------------------------------------------------------
 7.8 Iteration
 7.8.1 Infinite Iteration
      LOOP (form)*
 7.8.2 Infinite Iteration                                                      [Macro]
      DO    (((var [[init [step]]])*) (end-test (result)*
            (declaration)* (tag|statement)*                                  [Macro]
      DO*   (((var [[init [step]]])*) (end-test (result)*
            (declaration)* (tag|statement)*                                  [Macro]
 7.8.3 Simple Iteration Constructs
      DOLIST  (var listform  [resultform]) (declaration)* (tag|statement)*
                                                                             [Macro]
      DOTIMES (var countform [resultform]) (declaration)* (tag|statement)*
                                                                             [Macro]
 7.8.4 Mapping
      MAPCAR  function list &rest more-lists                       [Function]
      MAPLIST function list &rest more-lists                       [Function]
      MAPC    function list &rest more-lists                       [Function]
      MAPL    function list &rest more-lists                       [Function]
      MAPCAN  function list &rest more-lists                       [Function]
      MAPCON  function list &rest more-lists                       [Function]
  -- Comment ------------------------------------------------------------------ [Function]
      -> These are very primitive in LISP language.
 7.8.5 The "Program Feature"
      TAGBODY (tag|statement)*
      GO tag                                                       [Special Form]
  -- Deleted items ------------------------------------------------------------ [Special Form]
      PROG  ((var [init]))*) (declaration)* (tag|statement)*           [Macro]
      PROG* ((var [init]))*) (declaration)* (tag|statement)*           [Macro]
  -- Comment ------------------------------------------------------------------ [Macro]
      -> Old and Evil customs must be destroyed.TAGBODY and GO is needed
         to construct LOOP, DO and other iteration MACRO.
 ------------------------------------------------------------------------------
 7.9 Multiple Values
      VALUES &rest args
      MULTIPLE-VALUE-LIST  form                                    [Function]
      MULTIPLE-VALUE-CALL  function  (form)*                           [Macro]
      MULTIPLE-VALUE-PROG1 form      (form)*                       [Special Form]
      MULTIPLE-VALUE-BIND  form      (form)*                       [Special Form]
      MULTIPLE-VALUE-SETQ  variables  form                             [Macro]
  -- Deleted items ------------------------------------------------------------ [Macro]
      MULTIPLE-VALUES-LIMIT
      VALUES-LIST list                                             [Constant]
 ------------------------------------------------------------------------------ [Function]
 7.10 Dynamic Non-local Exits
      CATCH tag (form)*
      UNWIND-PROTECT protected-form (cleanup-form)*                [Special Form]
      THROW tag result                                             [Special Form]
  -- Comment ------------------------------------------------------------------ [Special Form]
      -> These features are quite necessary for constructive programming.
 ------------------------------------------------------------------------------
 8. Macro
 8.1 Macro Definition
      MACRO-FUNCTION  symbol
      DEFMACRO  name lambda-list (declaration|doc-string)*         [Function]
                (form)*
  -- Comments ----------------------------------------------------------------- [Macro]
      -> DEFMACRO var-list keywords: &optional,&rest,&key,
         &allow-other-keys and &aux are allowed, and &body,
         &whole and &environment are not allowed.
      -> &key, &allow-other-keys and &aux don't have so high necessity,
         but it leads to understand the specification clearly to have
         equality to DEFUN lambda-list keywords.
 ------------------------------------------------------------------------------
 8.2 Macro Expansion
      MACROEXPAND  form
      MACROEXPAND-1  form                                          [Function]
  -- Deleted Item ------------------------------------------------------------- [Function]
      *MACROEXPAND-HOOK*
  -- Comment ------------------------------------------------------------------ [Variable]
      -> MACROEXPAND,MACROEXPAND-1: optional parameter env
         is not allowed.
 ------------------------------------------------------------------------------
 9 Declarations
 9.1 Declaration Syntax
      DECLARE (decl-spec)*
  -- Deleted items ------------------------------------------------------------ [Special form]
      LOCALLY  (declaration)* (form)*
      PROCLAIM decl-spec                                           [Macro]
  -- Comments ----------------------------------------------------------------- [Function]
      -> 'PROCLAIM' is equivalent to 'DEFVAR', because the declaration
         specifiers except 'special' are omitted.
 ------------------------------------------------------------------------------
 9.2 Declaration Specifiers
      special
  -- Deleted items -----------------------------------------------------------
```

```
401       TYPE, FTYPE, FUNCTION, INLINE, NOTINLINE, IGNORE, OPTIMIZE, DECLARATION
402  ------------------------------------------------------------------------------
403  9.3 Type Declaration for Forms
404       THE value-type form
405   -- Comments ------------------------------------------------------ [Special form]
406       -> The syntax of 'THE' must be acceptable,
407          but its function may not be interpreted.
408  ------------------------------------------------------------------------------
409  10.1 The Property List
410       GET symbol indicator &optional default
411       REMPROP symbol indicator                                          [Function]
412       SYMBOL-PLIST symbol                                               [Function]
413   -- Deleted items ------------------------------------------------------------
414       GETF place indicator &optional default                            [Function]
415       REMF place indicator                                                 [Macro]
416       -> Not necessary                                                     [Macro]
417       GET-PROPERTIES place indicator-list
418       -> This is not so necessary and can be implemented by             [Function]
419          SYMBOL-PLIST very easily.
420  10.2 The Print Name
421       SYMBOL-NAME sym
422  10.3 Creating Symbols                                                  [Function]
423       MAKE-SYMBOL print-name
424       COPY-SYMBOL sym &optional copy-props                              [Function]
425       GENSYM &optional x                                                [Function]
426       SYMBOL-PACKAGE sym                                                [Function]
427       KEYWORDP object                                                   [Function]
428   -- Deleted item -------------------------------------------------------------
429       GENTEMP &optional prefix package                                  [Function]
430   -- Comments -----------------------------------------------------------------
431       -> GET and REMPROP are very important primitive functions on plist.
432       -> SYMBOL-xxxs are the primitive functions on symbols.
433       -> GENSYM is, of course, necessary but GENTEMP is not.
434          (Someone said, however, that GENTEMP is more useful than GENSYM.)
435  ------------------------------------------------------------------------------
436  11. Packages
437  ------------------------------------------------------------------------------
438  11.1. Consistency Rules
439       -- Read-read consistency
440       -- Print-read consistency
441       -- Print-print consistency
442  ------------------------------------------------------------------------------
443  11.2. Package Names
444       -- #\: is a separator of package name and symbol name.
445  ------------------------------------------------------------------------------
446  11.3. Translating Strings to Symbols
447       -- :Bar is a external symbol in package KEYWORD package.
448       -- #:Bar is an uninterned symbol but accessible from current package.
449  -- Deleted items ---------------------------------------------------------------
450       Symbol *package* contains current package. *Package* is deleted.
451       Foo:bar is a external symbol BAR that accessible from package FOO.
452       Foo::bar is a internal or external symbol BAR
453            that accessible form package FOO.
454            System locally binds *package* to FOO and accesses the symbol BAR.
455  ------------------------------------------------------------------------------
456  11.4. Exporting and Importing Symbols
457       -- Function INTERN makes a symbol in current package.
458  -- Deleted items ---------------------------------------------------------------
459       Package using and symbol importing and exporting features.
460  -- Comment ---------------------------------------------------------------------
461     The following two features are considered:
462          exporting symbols
463          using packages
464       In exporting symbols in a package, it is necessary to be searched all
465       symbol tables to find the conflict. It causes the system inefficient,
466       especially in personal computers. Consequently, symbol exporting and
467       package using features are eliminated. As a natural course of event,
468       the symbol shadowing feature is also deleted.
469        Furthermore, the 'colon' notation is difficult to use correctly for
470       beginners. For example, 'USER::CAR' represents the symbol in package
471       USER but it does not mean the symbol locally defined in the package
472       USER. This is because package USER uses package LISP, so the symbol
473       is specified as the inherited one from package LISP (the above notation
474       represents the symbol which is ACCESSIBLE from package USER by the CLtL
475       definition). Symbol-shadowing feature is the only way to define CAR
476       locally in the package USER.
477  ------------------------------------------------------------------------------
478  11.5. Name Conflicts
479  -- Comment ---------------------------------------------------------------------
480       The symbol shadowing are deleted, because we have no other package except
481       LISP and KEYWORD package.
482  ------------------------------------------------------------------------------
483  11.6. Built-in Packages
484       -- Lisp system initially must have following packages.
485            lisp
486            keyword
487  -- Deleted items ---------------------------------------------------------------
488       Next packages are deleted.
489            user
490            system
491       And we can't make new packages.
492
493  ------------------------------------------------------------------------------
494  11.7. Package System Functions and Variables
495       INTERN string &OPTIONAL package                                   [Function]
496       FIND-SYMBOL string &OPTIONAL package                              [Function]
497       UNINTERN symbol &OPTIONAL package                                 [Function]
498       DO-ALL-SYMBOLS (var [result-form])                                   [Macro]
499               (declaration)*
500               (tag | statement)*
```

```
501  -- Restriction ----------------------------------------------------------------
502      Function INTERN returns two values, the first value is the symbol itself
503          and the second value is one of followings.
504              :internal ... When the symbol is already there.
505              nil ... When the symbol is newly created.
506      Function FIND-SYMBOL's second value is one of followings.
507              :internal ... When the symbol is already accessible.
508              nil ... There is no symbol with that name.
509  -- Deleted items -----------------------------------------------------------------
510      *PACKAGE*                                                          [Variable]
511      MAKE-PACKAGE package-name &KEY :nicknames :use                     [Function]
512      IN-PACKAGE package-name &KEY :nicknames :use                       [Function]
513      FIND-PACKAGE name                                                  [Function]
514      PACKAGE-NAME package                                               [Function]
515      PACKAGE-NICKNAMES package                                          [Function]
516      RENAME-PACKAGE package new-name &OPTIONAL new-nicknames            [Function]
517      PACKAGE-USE-LIST package                                           [Function]
518      PACKAGE-USED-BY-LIST package                                       [Function]
519      PACKAGE-SHADOWING-SYMBOLS package                                  [Function]
520      LIST-ALL-PACKAGES                                                  [Function]
521      EXPORT symbols &OPTIONAL package                                   [Function]
522      UNEXPORT symbols &OPTIONAL package                                 [Function]
523      IMPORT symbols &OPTIONAL package                                   [Function]
524      SHADOWING-IMPORT symbols &OPTIONAL package                         [Function]
525      SHADOW symbols &OPTIONAL package                                   [Function]
526      USE-PACKAGE package-to-use &OPTIONAL package                       [Function]
527      UNUSE-PACKAGE package-to-unuse &OPTIONAL package                   [Function]
528      FIND-ALL-SYMBOLS string-or-symbol                                  [Function]
529      DO-SYMBOLS (var [package [result-form]])                             [Macro]
530              {declaration}*
531              {tag | statement}*
532      DO-EXTERNAL-SYMBOLS (var [package [result]])                         [Macro]
533              {declaration}*
534              {tag | statement}*
535  ------------------------------------------------------------------------------
536  11.8. Modules
537  -- Deleted items -----------------------------------------------------------------
538      *MODULES*                                                          [Variable]
539      PROVIDE module-name                                                [Function]
540      REQUIRE module-name &OPTIONAL pathname                             [Function]
541  -- Comment -----------------------------------------------------------------------
542      We think it is very poor without package creation.
543  ------------------------------------------------------------------------------
544  12. Numbers
545      All features on complex numbers are omitted, since they are too
546     inefficient on PC environment.
547  ------------------------------------------------------------------------------
548  12.1 Precision, Contagion, and Coercion
549      Same as CLtL except absence of features on complex numbers
550  ------------------------------------------------------------------------------
551  12.2 Predicates on Numbers
552       ZEROP number
553       PLUSP number                                                      [Function]
554       MINUSP number                                                     [Function]
555       ODDP number                                                       [Function]
556       EVENP number                                                      [Function]
557  ------------------------------------------------------------------------------
558  12.3 Comparisons on Numbers
559       =  number &rest more-numbers
560       /= number &rest more-numbers                                      [Function]
561       <  number &rest more-numbers                                      [Function]
562       >  number &rest more-numbers                                      [Function]
563       <= number &rest more-numbers                                      [Function]
564       >= number &rest more-numbers                                      [Function]
565       MAX number &rest more-numbers                                     [Function]
566       MIN number &rest more-numbers                                     [Function]
567  ------------------------------------------------------------------------------
568  12.4 Arithmetic Operations
569       +  &rest numbers                                                  [Function]
570       -  number &rest more-numbers                                      [Function]
571       *  &rest numbers                                                  [Function]
572       /  number &rest more-numbers                                      [Function]
573       1+ number                                                         [Function]
574       1- number                                                         [Function]
575       INCF place &optional delta                                          [Macro]
576       DECF place &optional delta                                          [Macro]
577       GCD &rest integers                                                [Function]
578       LCM integer &rest more-integers                                   [Function]
579  ------------------------------------------------------------------------------
580  12.5 Irrational and Transcendental Functions
581       EXP number                                                        [Function]
582       EXPT base-number power-number                                     [Function]
583       LOG number &optional base                                         [Function]
584       SQRT number                                                       [Function]
585       ABS number                                                        [Function]
586       SIGNUM number                                                     [Function]
587       SIN number                                                        [Function]
588       COS number                                                        [Function]
589       TAN number                                                        [Function]
590       ATAN y &optional x                                                [Function]
591       PI                                                                [Constant]
592  --- Comments ----------------------------------------------------------------------
593      The function 'SQRT' must signal error, when a negative argument is passed.
594      The function 'LOG' must signal error, when a negative argument is passed.
595  --- Deleted items -----------------------------------------------------------------
596       ISQRT integer                                                     [Function]
597       PHASE number                                                      [Function]
598       ASIN number                                                       [Function]
599       ACOS number                                                       [Function]
600       SINH number                                                       [Function]
```

```
601         COSH number                                                    [Function]
602         TANH number                                                    [Function]
603         ASINH number                                                   [Function]
604         ACOSH number                                                   [Function]
605         ATANH number                                                   [Function]
606   ------------------------------------------------------------------------------
607   12.6 Type Conversions and Component Extractions on Numbers
608         FLOAT number &optional other
609         RATIONAL number                                                [Function]
610         NUMERATOR rational                                             [Function]
611         DENOMINATOR rational                                           [Function]
612         FLOOR number &optional divisor                                 [Function]
613         CEILING number &optional divisor                               [Function]
614         TRUNCATE number &optional divisor                              [Function]
615         ROUND number &optional divisor                                 [Function]
616         MOD number divisor                                             [Function]
617         REM number divisor                                             [Function]
618   -- Deleted items ----------------------------------------------------------
619         RATIONALIZE number
620         FFLOOR number &optional divisor                                [Function]
621         FCEILING number &optional divisor                              [Function]
622         FTRUNCATE number &optional divisor                             [Function]
623         FROUND number &optional divisor                                [Function]
624         DECODE-FLOAT float                                             [Function]
625         SCALE-FLOAT float                                              [Function]
626         FLOAT-RADIX float                                              [Function]
627         FLOAT-SIGN float                                               [Function]
628         FLOAT-DIGITS float                                             [Function]
629         FLOAT-PRECISION float                                          [Function]
630         INTEGER-DECODE-FLOAT float                                     [Function]
631         COMPLEX realpart &optional imagpart                            [Function]
632         REALPART number                                                [Function]
633         IMAGPART number                                                [Function]
634         --> They are meaningless.
635   ------------------------------------------------------------------------------
636   12.7 Logical Operations on Numbers
637         LOGIOR &rest integers
638         LOGXOR &rest integers                                          [Function]
639         LOGAND &rest integers                                          [Function]
640         LOGNOT integer                                                 [Function]
641         LOGBITP index integer                                          [Function]
642         ASH integer count                                              [Function]
643   -- Deleted items ----------------------------------------------------------
644         LOGEQV & rest integers                                         [Function]
645         LOGNAND integer1 integer2                                      [Function]
646         LOGNOR integer1 integer2                                       [Function]
647         LOGANDC1 integer1 integer2                                     [Function]
648         LOGANDC2 integer1 integer2                                     [Function]
649         LOGORC1 integer1 integer2                                      [Function]
650         LOGORC2 integer1 integer2                                      [Function]
651         BOOLE op integer1 integer2                                     [Function]
652         BOOL-CLR                                                       [Function]
653         BOOL-SET                                                       [Constant]
654         BOOLE-1                                                        [Constant]
655         BOOLE-2                                                        [Constant]
656         BOOLE-C1                                                       [Constant]
657         BOOLE-C2                                                       [Constant]
658         BOOLE-AND                                                      [Constant]
659         BOOLE-IOR                                                      [Constant]
660         BOOLE-XOR                                                      [Constant]
661         BOOLE-EQV                                                      [Constant]
662         BOOLE-NAND                                                     [Constant]
663         BOOLE-NOR                                                      [Constant]
664         BOOLE-ANDC1                                                    [Constant]
665         BOOLE-ANDC2                                                    [Constant]
666         BOOLE-ORC1                                                     [Constant]
667         BOOLE-ORC2                                                     [Constant]
668         LOGTEST integer1 integer2                                      [Constant]
669         LOGCOUNT integer                                               [Function]
670         INTEGER-LENGTH integer                                         [Function]
671   ------------------------------------------------------------------------------
672   12.8 Byte Manipulation Functions
673        All byte manipulation functions are omitted, since they are too
674       inefficient on PC environment.
675   -- Deleted items ----------------------------------------------------------
676       BYTE size position                                              [Function]
677       BYTE-SIZE bytespec                                              [Function]
678       BYTE-POSITION bytespec                                          [Function]
679       LDB bytespec integer                                            [Function]
680       LDB-TEST bytespec integer                                       [Function]
681       MASK-FIELD bytespec integer                                     [Function]
682       DPB newbyte bytespec integer                                    [Function]
683       DEPOSIT-FIELD newbyte bytespec integer                          [Function]
684   12.9 Random Numbers
685        All features on random-state are deleted.
686   -- Deleted items ----------------------------------------------------------
687       RANDOM number &optional state                                   [Function]
688       *RANDOM-STATE*                                                  [Variable]
689       MAKE-RANDOM-STATE &optional state                               [Function]
690       RANDOM-STATE-P object                                           [Function]
691   ------------------------------------------------------------------------------
692   12.10 Implementation Parameters
693        All implementation parameters on number are omitted to keep the system
694       compact.
695   -- Deleted items ----------------------------------------------------------
696       MOST-POSITIVE-FIXNUM                                            [Constant]
697       MOST-NEGATIVE-FIXNUM                                            [Constant]
698       MOST-POSITIVE-SHORT-FLOAT                                       [Constant]
699       LEAST-POSITIVE-SHORT-FLOAT                                      [Constant]
700       LEAST-NEGATIVE-SHORT-FLOAT                                      [Constant]
```

```
701       MOST-NEGATIVE-SHORT-FLOAT                                          [Constant]
702       MOST-POSITIVE-SINGLE-FLOAT                                         [Constant]
703       LEAST-POSITIVE-SINGLE-FLOAT                                        [Constant]
704       LEAST-NEGATIVE-SINGLE-FLOAT                                        [Constant]
705       MOST-NEGATIVE-SINGLE-FLOAT                                         [Constant]
706       MOST-POSITIVE-DOUBLE-FLOAT                                         [Constant]
707       LEAST-POSITIVE-DOUBLE-FLOAT                                        [Constant]
708       LEAST-NEGATIVE-DOUBLE-FLOAT                                        [Constant]
709       MOST-NEGATIVE-DOUBLE-FLOAT                                         [Constant]
710       MOST-POSITIVE-LONG-FLOAT                                           [Constant]
711       LEAST-POSITIVE-LONG-FLOAT                                          [Constant]
712       LEAST-NEGATIVE-LONG-FLOAT                                          [Constant]
713       MOST-NEGATIVE-LONG-FLOAT                                           [Constant]
714       SHORT-FLOAT-EPSILON                                                [Constant]
715       SINGLE-FLOAT-EPSILON                                               [Constant]
716       DOUBLE-FLOAT-EPSILON                                               [Constant]
717       LONG-FLOAT-EPSILON                                                 [Constant]
718       SHORT-FLOAT-NEGATIVE-EPSILON                                       [Constant]
719       SINGLE-FLOAT-NEGATIVE-EPSILON                                      [Constant]
720       DOUBLE-FLOAT-NEGATIVE-EPSILON                                      [Constant]
721       LONG-FLOAT-NEGATIVE-EPSILON                                        [Constant]
722    ------------------------------------------------------------------------------
723    13 Character
724     -- Comments ---------------------------------------------------------------
725         -> The CL Core's character doesn't have the font and bits attributes.
726         -> If necessary, the CL Core consider the font and bits attributes
727            as zero.
728    ------------------------------------------------------------------------------
729    13.1 Character Attributes
730     -- Deleted Items ------------------------------------------------------------
731         CHAR-CODE-LIMIT                                                  [Constant]
732         CHAR-FONT-LIMIT                                                  [Constant]
733         CHAR-BITS-LIMIT                                                  [Constant]
734         -> System constants should be omitted.
735    ------------------------------------------------------------------------------
736    13.2 Predicates on Characters
737         STANDARD-CHAR-P char                                             [Function]
738         GRAPHIC-CHAR-P char                                              [Function]
739         STRING-CHAR-P char                                               [Function]
740         ALPHA-CHAR-P char                                                [Function]
741         UPPER-CASE-P char                                                [Function]
742         LOWER-CASE-P char                                                [Function]
743         BOTH-CHAR-P char                                                 [Function]
744         DIGIT-CHAR-P char &optional (radix 10)                           [Function]
745         CHAR= character &rest more-characters                            [Function]
746         CHAR/= character &rest more-characters                           [Function]
747         CHAR< character &rest more-characters                            [Function]
748         CHAR> character &rest more-characters                            [Function]
749         CHAR<= character &rest more-characters                           [Function]
750         CHAR>= character &rest more-characters                           [Function]
751         -> These are primitive predicates for treating characters.
752         CHAR-EQUAL character &rest more-characters                       [Function]
753         CHAR-NOT-EQUAL character &rest more-characters                   [Function]
754         CHAR-LESSP character &rest more-characters                       [Function]
755         CHAR-GREATERP character &rest more-characters                    [Function]
756         CHAR-NOT-GREATERP character &rest more-characters                [Function]
757         CHAR-NOT-LESSP character &rest more-characters                   [Function]
758     -- Deleted Item -------------------------------------------------------------
759         ALPHANUMERICP char                                               [Function]
760         -> ALPHANUMERICP is equal to (OR (ALPHA-CHAR-P char) (DIGIT-CHAR-P char))
761    ------------------------------------------------------------------------------
762    13.3 Character Construction and Selection
763         CHAR-CODE char                                                   [Function]
764         CODE-CHAR code                                                   [Function]
765    -- Comment ------------------------------------------------------------------
766         -> CODE-CHAR's optional parameter (font, bits) is omitted.
767    -- Deleted Items ------------------------------------------------------------
768         CHAR-BITS char                                                   [Function]
769         CHAR-FONT char                                                   [Function]
770         MAKE-CHAR char &optional (bits 0) (font 0)                       [Function]
771         -> When the font and bits attributes are zero, MAKE-CHAR returns char
772            same as its argument char.
773    ------------------------------------------------------------------------------
774    13.4 Character Conversions
775         CHARACTER object                                                 [Function]
776         CHAR-UPCASE char                                                 [Function]
777         CHAR-DOWNCASE char                                               [Function]
778         DIGIT-CHAR weight &optional (radix 10)                           [Function]
779         CHAR-NAME char                                                   [Function]
780         NAME-CHAR name                                                   [Function]
781    -- Comment ------------------------------------------------------------------
782         -> DIGIT-CHAR's optional font parameter is omitted.
783    -- Deleted Items ------------------------------------------------------------
784         CHAR-INT char                                                    [Function]
785         INT-CHAR integer                                                 [Function]
786         -> CHAR-INT returns the same integer CHAR-CODE would, because
787            the font and bits attributes are zero.
788         -> And thus INT-CHAR is equal to CODE-CHAR.
789    ------------------------------------------------------------------------------
790    13.5 Character Control-Bit Functions
791    -- Deleted Items ------------------------------------------------------------
792         CHAR-CONTROL-BIT                                                 [Constant]
793         CHAR-META-BIT                                                    [Constant]
794         CHAR-SUPER-BIT                                                   [Constant]
795         CHAR-HYPER-BIT                                                   [Constant]
796         CHAR-BIT char name                                               [Function]
797         SET-CHAR-BIT char name newvalue                                  [Function]
798         -> Because the font and bits attributes are not implemented,
799            all character control-bit functions are omitted.
800    14 Sequences
```

```
301  --  Comment ----------------------------------------------------------------
302      Functions for the sequence is also important for the string,
303    Because CL does not have rich facility on string.
804  ------------------------------------------------------------------------------
805  14.1 Simple Sequence Function
806       ELT       sequence index                                          [Function]
807       SUBSEQ    sequence start &optional end                            [Function]
808       COPY-SEQ sequence                                                 [Function]
809       LENGTH    sequence                                                [Function]
810       REVERSE   sequence                                                [Function]
811       NREVERSE sequence                                                 [Function]
812    -- Deleted items ----------------------------------------------------------
813       MAKE-SEQUENCE type size &key :initial-element                     [Function]
814  ------------------------------------------------------------------------------
815  14.2 Concatenating, Mapping, and Reducing Sequences
816       CONCATENATE result-type &rest sequence
817       SOME      predicate sequence &rest more-sequence                  [Function]
818       EVERY     predicate sequence &rest more-sequence                  [Function]
819    -- Deleted items ----------------------------------------------------------
820       MAP result-type function sequence &rest more-sequence           [Function]
821       NOTANY    predicate sequence &rest more-sequence                  [Function]
822       NOTEVERY predicate sequence &rest more-sequence                   [Function]
823       REDUCE function sequence &key :from-end :start :end :initial-value
824                                                                         [Function]
825  ------------------------------------------------------------------------------
826  14.3 Modifying Sequences
827       FILL sequence item &key :start :end
828       REPLACE sequence1 sequence2 &key :start1 :end1 :start2 :end2      [Function]
829    -- Restricted -------------------------------------------------------------
830       Keyword parameters, :from-end, :test-not and :key are omitted from
831       following six functions.
832       REMOVE      item sequence &key :test :start :end :count           [Function]
833       DELETE      item sequence &key :test :start :end :count           [Function]
834       REMOVE-DUPLICATES sequence &key :test :start :end :count          [Function]
835       DELETE-DUPLICATES sequence &key :test :start :end :count          [Function]
836       SUBSTITUTE  newitem olditem sequence &key :test :start :end :count
837                                                                         [Function]
838       NSUBSTITUTE newitem olditem sequence &key :test :start :end :count
839                                                                         [Function]
840    -- Deleted items ----------------------------------------------------------
841       REMOVE-IF       test sequence &key :from-end :start :end :count :key
842                                                                         [Function]
843       REMOVE-IF-NOT   test sequence &key :from-end :start :end :count :key
844                                                                         [Function]
845       DELETE-IF       test sequence &key :from-end :start :end :count :key
846                                                                         [Function]
847       DELETE-IF-NOT   test sequence &key :from-end :start :end :count :key
848                                                                         [Function]
849       SUBSTITUTE-IF   newitem test sequence
850                       &key :from-end :start :end :count :key            [Function]
851       SUBSTITUTE-IF-NOT newitem test sequence
852                         &key :from-end :start :end :count :key          [Function]
853       NSUBSTITUTE-IF    newitem test sequence
854                         &key :from-end :start :end :count :key          [Function]
855       NSUBSTITUTE-IF-NOT newitem test sequence
856                         &key :from-end :start :end :count :key          [Function]
857    -- Comments ---------------------------------------------------------------
858       ->Treat following eight functions in a same manner
859           REMOVE, DELETE, SUBSTITUTE, NSUBSTITUTE, FIND, POSITION,
860         COUNT, SEARCH
861       ->Omit the key-word parameters such as :from-end, :test-not and :key.
862  ------------------------------------------------------------------------------
863  14.4 Searching Sequences for Items
864    -- Restricted -------------------------------------------------------------
865       Keyword parameters, :from-end, :test-not and :key are omitted from
866       the following four functions.
867       FIND        item sequence &key :test :start :end                  [Function]
868       POSITION    item sequence &key :test :start :end                  [Function]
869       COUNT       item sequence &key :test :start :end                  [Function]
870       SEARCH      sequence1 sequence2
871                   &key :test :start1 :start2 :end1 :end2                [Function]
872    -- Deleted items ----------------------------------------------------------
873       MISMATCH  sequence1 sequence2 &key :start1 :start2 :end1 :end2 [Function]
874       FIND-IF         test sequence &key :from-end :start :end :key  [Function]
875       FIND-IF-NOT     test sequence &key :from-end :start :end :key  [Function]
876       POSITION-IF     test sequence &key :from-end :start :end :key  [Function]
877       POSITION-IF-NOT test sequence &key :from-end :start :end :key  [Function]
878       COUNT-IF        test sequence &key :from-end :start :end :key  [Function]
879       COUNT-IF-NOT    test sequence &key :from-end :start :end :key  [Function]
880  ------------------------------------------------------------------------------
881  14.5 Sorting and Merging
882    -- Restricted -------------------------------------------------------------
883       Keyword parameter, :key is omitted from SORT.
884       SORT        sequence predicate
885    -- Deleted items ----------------------------------------------------------
886       STABLE-SORT sequence predicate &key :key                          [Function]
887       MERGE result-type sequence1 sequence2                             [Function]
888  ------------------------------------------------------------------------------
889  15. Lists
890  15.1 Conses
891       CAR  list
892       CDR  list                                                         [Function]
893       CAAR  list                                                        [Function]
894       CADR  list                                                        [Function]
895       CDAR  list                                                        [Function]
896       CDDR  list                                                        [Function]
897       CAAAR  list                                                       [Function]
898       CAADR  list                                                       [Function]
899       CADAR  list                                                       [Function]
900       CADDR  list                                                       [Function]
```

```
901       CDAAR  list                                                    [Function]
902       CDADR  list                                                    [Function]
903       CDDAR  list                                                    [Function]
904       CDDDR  list                                                    [Function]
905       CONS  x y                                                      [Function]
906    --  Deleted items ----------------------------------------------------------
907       CAAAAR  list                                                   [Function]
908       CAAADR  list                                                   [Function]
909       CAADAR  list                                                   [Function]
910       CAADDR  list                                                   [Function]
911       CADAAR  list                                                   [Function]
912       CADADR  list                                                   [Function]
913       CADDAR  list                                                   [Function]
914       CADDDR  list                                                   [Function]
915       CDAAAR  list                                                   [Function]
916       CDAADR  list                                                   [Function]
917       CDADAR  list                                                   [Function]
918       CDADDR  list                                                   [Function]
919       CDDAAR  list                                                   [Function]
920       CDDADR  list                                                   [Function]
921       CDDDAR  list                                                   [Function]
922       CDDDDR  list                                                   [Function]
923       TREE-EQUAL  x y &key :test :test-not                           [Function]
924    ------------------------------------------------------------------------------
925    15.2 Lists
926
927       ENDP  object                                                   [Function]
928       LIST-LENGTH  list                                              [Function]
929       NTH    list                                                    [Function]
930       FIRST  list                                                    [Function]
931       SECOND  list                                                   [Function]
932       THIRD  list                                                    [Function]
933       FOURTH  list                                                   [Function]
934       FIFTH  list                                                    [Function]
935       SIXTH  list                                                    [Function]
936       SEVENTH  list                                                  [Function]
937       EIGHTH  list                                                   [Function]
938       NINTH  list                                                    [Function]
939       TENTH  list                                                    [Function]
940       REST  list                                                     [Function]
941       NTHCDR  n list                                                 [Function]
942       LAST  list                                                     [Function]
943       LIST  &rest args                                               [Function]
944       LIST*  arg &rest others                                        [Function]
945       APPEND  &rest lists                                            [Function]
946       COPY-TREE  object                                              [Function]
947       NCONC  &rest lists                                             [Function]
948       PUSH  item place                                               [Function]
949       POP  place                                                     [Function]
950       LDIFF  list sublist                                            [Function]
951    --  Deleted items ----------------------------------------------------------
952       MAKE-LIST  size &key :initial-element                          [Function]
953       COPY-LIST  list                                                [Function]
954       COPY-ALIST list                                                [Function]
955       REVAPPEND  x y                                                 [Function]
956       NRECONC  x y                                                   [Function]
957       PUSHNEW  item place &key :test :test-not :key                  [Function]
958       BUTLAST  list &optional n                                      [Function]
959       NBUTLAST  list &optional n                                     [Function]
960    ------------------------------------------------------------------------------
961    15.3 Alteration of List Structure
962       RPLACA  x y                                                    [Function]
963       RPLACD  x y                                                    [Function]
964    ------------------------------------------------------------------------------
965    15.4 Substitution of Expressions
966       SUBST  new old tree &key :test                                 [Function]
967       NSUBST  new old tree &key :test                                [Function]
968    --  Deleted items ----------------------------------------------------------
969       SUBST-IF  new old tree &key :key                               [Function]
970       SUBST-IF-NOT  new old tree &key :key                           [Function]
971       SUBLIS  alist tree &key :test :test-not :key                   [Function]
972       NSUBLIS  alist tree &key :test :test-not :key                  [Function]
973    --  Comments ---------------------------------------------------------------
974       ->SUBST,NSUBST: lambda keyword :test-not and :key are
975       not allowed.
976       -> -IF, -IF-NOT and :test-not, :key are deleted as Sequences.
977    ------------------------------------------------------------------------------
978    15.5 Using Lists as Sets
979       MEMBER  item list &key :test                                   [Function]
980       ADJOIN  item list                                              [Function]
981       UNION  list1 list2 &key :test                                  [Function]
982       NUNION  list1 list2 &key :test                                 [Function]
983       INTERSECTION  list1 list2 &key :test                           [Function]
984       NINTERSECTION  list1 list2 &key :test                          [Function]
985       SET-DIFFERENCE  list1 list2 &key :test                         [Function]
986       NSET-DIFFERENCE  list1 list2 &key :test                        [Function]
987       SET-EXCLUSIVE-OR  list1 list2 &key :test                       [Function]
988       NSET-EXCLUSIVE-OR  list1 list2 &key :test                      [Function]
989       SUBSETP  list1 list2 &key :test                                [Function]
990    --  Deleted items ----------------------------------------------------------
991       MEMBER-IF  new old tree &key :key                              [Function]
992       MEMBER-IF-NOT  new old tree &key :key                          [Function]
993       TAILP  sublist list                                            [Function]
994    --  Comments ---------------------------------------------------------------
995       ->MEMBER,UNION,NUNION,INTERSECTION,NINTERSECTION,
996       SET-DIFFERENCE,NSET-DIFFERENCE,SET-EXCLUSIVE-OR,
997       NSET-EXCLUSIVE-OR,SUBSETP:
998            lambda keyword :test-not and :key are not allowed.
999       -> using lists as sets non-destructively are very useful and using
1000      lists as sets destructively are useful and primitive.
```

```
15.6 Association lists
     ACONS  key datum a-list
     ASSOC  item a-list &key :test                                    [Function]
     RASSOC  item a-list &key :test                                   [Function]
 --  Deleted items ------------------------------------------------------------
     PAIRLIS   keys data &optional a-list                             [Function]
     ASSOC-IF  item a-list &key :test                                 [Function]
     ASSOC-IF-NOT  item a-list &key :test                             [Function]
     RASSOC-IF  item a-list &key :test                                [Function]
     RASSOC-IF-NOT  item a-list &key :test                            [Function]
 --  Comment ------------------------------------------------------------------
     ->ASSOC,RASSOC: lambda keyword :test-not and :key are
     not allowed.
     -> association lists don't have so many necessities now.
------------------------------------------------------------------------------
16 Hash Tables
 -- Deleted items -----------------------------------------------------------
     MAKE-HASH-TABLE &key :test :size :rehash-size
                          :rehash-threshold
     HASH-TABLE-P object                                              [Function]
     GETHASH key hash-table &optional default                         [Function]
     REMHASH key hash-table                                           [Function]
     MAPHASH function hash-table                                      [Function]
     CLRHASH hash-table                                               [Function]
     HASH-TABLE-COUNT hash-table                                      [Function]
     SXHASH object                                                    [Function]
 -- Comments ----------------------------------------------------------------
     -> the hash table is too complicated. and the necessity is low.
------------------------------------------------------------------------------
17. Arrays
     Only simple arrays are available.
     Bit-arrays are omitted.
 -- Comments ----------------------------------------------------------------
     -> Simple arrays are enough on PCs.
     -> Bit-arrays are not necessary in usual applications.
17.1 Array Creation
     MAKE-ARRAY dimensions &key :initial-element                      [Function]
                                :initial-contents
 -- Deleted items -----------------------------------------------------------
                                :element-type
                                :adjustable
                                :fill-pointer
                                :displaced-to
                                :displaced-index-offset
 -- Restricted --------------------------------------------------------------
     The maximum array rank available can be 3 (not 7).
 -- Comments ----------------------------------------------------------------
     -> Element-type "t" is enough on PCs.
        Restricted array element feature is for efficiency and is useful
        with foreign languages. These extensions should be left for venders.
     -> Adjustable-array is complexed feature and is not necessary on PCs.
     -> Fill-pointer may be useful but is not an essential feature on vectors.
     VECTOR &rest objects                                             [Function]
     -> VECTOR helps users write clearer code using sequences though
        it can be substituted by MAKE-ARRAY.
 -- Deleted items -----------------------------------------------------------
     ARRAY-RANK-LIMIT                                                 [Constant]
     ARRAY-DIMENSION-LIMIT                                            [Constant]
     ARRAY-TOTAL-SIZE-LIMIT                                           [Constant]
     -> System constants should be omitted (by our primary principle).
17.2 Array Access
     AREF array &rest subscripts                                      [Function]
     SVREF simple-vector index                                        [Function]
     -> SVREF helps users write clearer code accessing vectors though
        it can be substituted by AREF.
17.3 Array Information
     ARRAY-RANK array                                                 [Function]
     ARRAY-DIMENSION array axis-number                                [Function]
     ARRAY-IN-BOUNDS-P array &rest subscripts                         [Function]
 -- Comments ----------------------------------------------------------------
     -> These are the very primitive functions on arrays.
 -- Deleted items -----------------------------------------------------------
     ARRAY-ELEMENT-TYPE array                                         [Function]
     -> The element-type cannot be specified in MAKE-ARRAY.
     ARRAY-TOTAL-SIZE array                                           [Function]
     ARRAY-DIMENSIONS array                                           [Function]
     ARRAY-ROW-MAJOR-INDEX array &rest subscripts                     [Function]
     -> These are not essential and can be computed using other primitive
        functions very easily.
     ADJUSTABLE-ARRAY-P array                                         [Function]
     -> This is not necessary because only simple-arrays are available.
17.4 Functions on Arrays of Bits
 -- Deleted items -----------------------------------------------------------
     BIT bit-array &rest subscripts                                   [Function]
     SBIT simple-bit-array &rest subscripts                           [Function]
     BIT-AND bit-array1 bit-array2 &optional result-bit-array         [Function]
     BIT-IOR bit-array1 bit-array2 &optional result-bit-array         [Function]
     BIT-XOR bit-array1 bit-array2 &optional result-bit-array         [Function]
     BIT-EQV bit-array1 bit-array2 &optional result-bit-array         [Function]
     BIT-NAND bit-array1 bit-array2 &optional result-bit-array        [Function]
     BIT-NOR bit-array1 bit-array2 &optional result-bit-array         [Function]
     BIT-ANDC1 bit-array1 bit-array2 &optional result-bit-array       [Function]
     BIT-ANDC2 bit-array1 bit-array2 &optional result-bit-array       [Function]
     BIT-ORC1 bit-array1 bit-array2 &optional result-bit-array        [Function]
     BIT-ORC2 bit-array1 bit-array2 &optional result-bit-array        [Function]
     BIT-NOT bit-array &optional result-bit-array                     [Function]
     -> BIT-ARRAYs are not necessary in usual applications.
17.5 Fill Pointers
 -- Deleted items -----------------------------------------------------------
```

```
1101        ARRAY-HAS-FILL-POINTER-P array                                    [Function]
1102        FILL-POINTER vector                                               [Function]
1103        VECTOR-PUSH new-element vector                                    [Function]
1104        VECTOR-PUSH-EXTEND new-element vector &optional extension         [Function]
1105        VECTOR-POP vector                                                 [Function]
1106        -> Only simple-arrays are available.
1107     -- Comments ----------------------------------------------------------------------
1108        -> Fill-pointer might be useful in some applications
1109           but seems to be unnatural feature on the vector.
1110    17.6 Changing the Dimensions of an Array
1111     -- Deleted item -------------------------------------------------------------------
1112        ADJUST-ARRAY array new-dimensions &key                            [Function]
1113                                          :element-type
1114                                          :initial-element
1115                                          :initial-contents
1116                                          :fill-pointer
1117                                          :displaced-to
1118                                          :displaced-index-offset
1119        -> Only simple-arrays are available.
1120    17.6 Changing the Dimensions of an Array
1121     -- Deleted item -------------------------------------------------------------------
1122        ADJUST-ARRAY array new-dimensions &key                            [Function]
1123                                          :element-type
1124                                          :initial-element
1125    ------------------------------------------------------------------------------------
1126    18. Strings
1127    -- Comment -------------------------------------------------------------------------
1128        Strings' partial manipulation is handled by functions that treat sequence
1129        data type like SUBSEQ.
1130    ------------------------------------------------------------------------------------
1131    18.1. String Access
1132        CHAR string index                                                 [Function]
1133        SCHAR simple-string index                                         [Function]
1134    -- Comment -------------------------------------------------------------------------
1135        There is a recommend of these two functions. Use SCHAR rather than CHAR,
1136        because of transporting program to Common Lisp full set. SCHAR is faster
1137        than CHAR.
1138    ------------------------------------------------------------------------------------
1139    18.2. String Comparison
1140        STRING= string1 string2 &KEY :start1 :end1 :start2 :end2          [Function]
1141        STRING-EQUAL string1 string2 &KEY :start1 :end1 :start2 :end2
1142                                                                          [Function]
1143        STRING< string1 string2 &KEY :start1 :end1 :start2 :end2          [Function]
1144        STRING> string1 string2 &KEY :start1 :end1 :start2 :end2          [Function]
1145        STRING<= string1 string2 &KEY :start1 :end1 :start2 :end2         [Function]
1146        STRING>= string1 string2 &KEY :start1 :end1 :start2 :end2         [Function]
1147        STRING/= string1 string2 &KEY :start1 :end1 :start2 :end2         [Function]
1148        STRING-LESSP string1 string2 &KEY :start1 :end1 :start2 :end2
1149                                                                          [Function]
1150        STRING-GREATERP string1 string2 &KEY :start1 :end1 :start2 :end2
1151                                                                          [Function]
1152        STRING-NOT-GREATERP string1 string2 &KEY :start1 :end1 :start2 :end2
1153                                                                          [Function]
1154        STRING-NOT-LESSP string1 string2 &KEY :start1 :end1 :start2 :end2
1155                                                                          [Function]
1156        STRING-NOT-EQUAL string1 string2 &KEY :start1 :end1 :start2 :end2
1157                                                                          [Function]
1158    -- Comment -------------------------------------------------------------------------
1159        String comparisons remain with character comparisons.
1160
1161    ------------------------------------------------------------------------------------
1162    18.3. String Construction and Manipulation
1163        MAKE-STRING size &KEY :initial-element                            [Function]
1164        STRING-TRIM character-bag string                                  [Function]
1165        STRING-LEFT-TRIM character-bag string                             [Function]
1166        STRING-RIGHT-TRIM character-bag string                            [Function]
1167        STRING-UPCASE string &KEY :start :end                             [Function]
1168        STRING-DOWNCASE string &KEY :start :end                           [Function]
1169        STRING-CAPITALIZE string &KEY :start :end                         [Function]
1170        NSTRING-UPCASE string &KEY :start :end                            [Function]
1171        NSTRING-DOWNCASE string &KEY :start :end                          [Function]
1172        NSTRING-CAPITALIZE string &KEY :start :end                        [Function]
1173        STRING x                                                          [Function]
1174    -- Comment -------------------------------------------------------------------------
1175        We need MAKE-STRING to prepare the buffer of large string.  For
1176        example, prepare a buffer for screen of screen editor.
1177    -- Restrictions --------------------------------------------------------------------
1178        Cut keyword parameters :START and :END.
1179        Usually case conversion function will be applied to whole string.
1180        Please use SUBSEQ to modify the part of string.
1181    ------------------------------------------------------------------------------------
1182    19. Structures
1183        DEFSTRUCT name [doc-string] (slot-description)+                   [Macro]
1184        Legal syntax for the slot-descriptions:
1185        (slot-name [default-init])
1186        or
1187        slot-name
1188     -- Deleted Items -------------------------------------------------------------------
1189        slot options:
1190           :type
1191           :read-only
1192        defstruct options:
1193           :conc-name
1194           :constructor
1195           :copier
1196           :predicate
1197           :include
1198           :print-function
1199           :type
1200           :named
```

```
1201          :initial-offset
1202      --> to keep the system compact.
1203    ------------------------------------------------------------------------
1204    20 The Evaluator
1205    20.1 Run-Time Evaluation of Forms
1206         EVAL form                                                    [Function]
1207         -> EVAL is very important primitive function.
1208         CONSTANTP object                                             [Function]
1209         -> CONSTANTP is the only function to judge if object is constant.
1210    -- Deleted Items ----------------------------------------------------------
1211         *EVALHOOK*                                                   [Variable]
1212         *APPLYHOOK*                                                  [Variable]
1213         EVALHOOK form evalfookfn applyhookfn &optional env           [Function]
1214         APPLYHOOK function args evalhookfn applyhookfn               [Function]
1215                  &optional env
1216         -> These are not necessary for ordinary users, because hook feature
1217            is provided to make debugging tools.
1218         -> But any hook feature will be provided implicitly for making
1219            debugging tools on the CL Core.
1220    ------------------------------------------------------------------------
1221    20.2 The Top-Level Loop
1222         +                                                            [Variable]
1223         -                                                            [Variable]
1224         *                                                            [Variable]
1225         /                                                            [Variable]
1226         -> At least these are necessary for standard user interaction.
1227    -- Deleted Items ----------------------------------------------------------
1228         ++                                                           [Variable]
1229         +++                                                          [Variable]
1230         **                                                           [Variable]
1231         ***                                                          [Variable]
1232         //                                                           [Variable]
1233         ///                                                          [Variable]
1234    21 Streams
1235    21.1 Standard Streams
1236      *standard-input*                                                [Variable]
1237      *standard-output*                                               [Variable]
1238      *error-output*                                                  [Variable]
1239      *query-io*                                                      [Variable]
1240      *debug-io*                                                      [Variable]
1241      *terminal-io*                                                   [Variable]
1242      *trace-output*                                                  [Variable]
1243    ------------------------------------------------------------------------
1244    21.2 Creating New Streams
1245     -- Deleted items-------------------------------------------------------
1246      MAKE-SYNONYM-STREAM symbol                                      [Function]
1247      MAKE-BROADCAST-STREAM &rest streams                             [Function]
1248      MAKE-CONCATENATED-STREAM &rest streams                          [Function]
1249      MAKE-TWO-WAY-STREAM input-stream output-stream                  [Function]
1250      MAKE-ECHO-STREAM input-stream output-stream                     [Function]
1251      MAKE-STRING-INPUT-STREAM string &optional start end             [Function]
1252      MAKE-STRING-OUTPUT-STREAM                                       [Function]
1253      GET-OUTPUT-STREAM-STRING string-output-stream                   [Function]
1254      WITH-OPEN-STREAM       (var stream)   (declaration)* (form)*      [Macro]
1255      WITH-INPUT-FROM-STRING (var string (keyword value)*) (declaration)* (form)*
1256                                                                       [Macro]
1257      WITH-OUTPUT-TO-STRING  (var [string]) (declaration)* (form)*     [Macro]
1258    ---  Comments  ---------------------------------------------------------
1259        The Stream as an object in the CL data type is less needed, but
1260      it is very useful and important to abstract I/O.
1261    ------------------------------------------------------------------------
1262    21.3 Operations on Streams
1263         STREAMP              object                                  [Function]
1264         INPUT-STREAM-P       stream                                  [Function]
1265         OUTPUT-STREAM-P      stream                                  [Function]
1266         STREAM-ELEMENT-TYPE stream                                   [Function]
1267     -- Restricted ----------------------------------------------------------
1268         Keyword parameter, :abort is omitted from CLOSE.
1269         CLOSE             stream                                     [Function]
1270    ------------------------------------------------------------------------
1271    22. Input/Output
1272    22.1 Printed Representation of Lisp Objects
1273         macro characters
1274           (  )  '  ;  "  `  ,  @  #
1275         standard dispatch macro character syntax
1276           #\ #' #( #n( #: #B #O #nA #S #+ #- #! #<
1277         *READTABLE*                                                  [Variable]
1278         COPY-READTABLE  &optional from-readtable to-readtable
1279                                                                      [Function]
1280         READTABLEP  readtable                                        [Function]
1281         SET-SYNTAX-FROM-CHAR  to-char from-char &optional
1282                  from-readtable to-read-table                        [Function]
1283         SET-MACRO-CHARACTER  char function  &optional
1284                  non-terminating-p readtable                         [Function]
1285         GET-MACRO-CHARACTER  char &optional readtable                [Function]
1286         MAKE-DISPATCH-MACRO-CHARACTER  char &optional
1287                  non-terminating-p readtable                         [Function]
1288         SET-DISPATCH-MACRO-CHARACTER  disp-char sub-char function
1289                  &optional readtable                                 [Function]
1290         GET-DISPATCH-MACRO-CHARACTER  disp-char sub-char
1291                  &optional readtable                                 [Function]
1292         *PRINT-LENGTH*                                               [Variable]
1293         *PRINT-LEVEL*                                                [Variable]
1294         *PRINT-CASE*                                                 [Variable]
1295      --  Deleted items ---------------------------------------------------
1296         standard dispatch macro character syntax
1297           #* #n* #. #, #nR #n= #n# #) #C
1298         *READ-BASE*                                                  [Variable]
1299         *READ-SUPPRESS*                                              [Variable]
1300         *PRINT-ESCAPE*                                               [Variable]
```

PRINT-PRETTY [Variable]
PRINT-CIRCLE [Variable]
PRINT-BASE [Variable]
PRINT-RADIX [Variable]
PRINT-GENSYM [Variable]
PRINT-ARRAY [Variable]
-- Comment ---------- [Variable]
-> readtables are very useful to extend input format.

22.2 Input Functions
READ &optional input-stream eof-error-p eof-value recursive-p [Function]
READ-LINE &optional input-stream eof-error-p eof-value recursive-p [Function]
READ-CHAR &optional input-stream eof-error-p eof-value recursive-p [Function]
UNREAD-CHAR character &optional input-stream [Function]
LISTEN &optional input-stream [Function]
READ-CHAR-NO-HANG &optional in-stream eof-error-p eof-value recursive-p [Function]
CLEAR-INPUT &optional input-stream [Function]
READ-FROM-STRING string &optional eof-error-p eof-value &key :start :end [Function]
READ-BYTE binary-input-stream &optional eof-error-p eof-value [Function]
-- Deleted items ---------- [Function]
READ-DEFAULT-FLOAT-FORMAT [Variable]
READ-PRESERVING-WHITESPACE &optional in-stream eof-error-p eof-value recursive-p [Function]
READ-DELIMITED-LIST char &optional input-stream recursive-p [Function]
PEEK-CHAR &optional peek-type input-stream eof-error-p eof-value recursive-p [Function]
PARSE-INTEGER string &key :start :end :radix :junk-allowed [Function]
-- Comments ---------- [Function]
->READ-FROM-STRING keyword parameter :preserve-whitespace is not allowed.
-> string i/o is useful for the lisp internal editor and binary i/o is necessary for handling alien structures producted by other language.

22.3 Output Functions
PRIN1 object &optional output-stream [Function]
PRINT object &optional output-stream [Function]
PPRINT object &optional output-stream [Function]
PRINC object &optional output-stream [Function]
PRIN1-TO-STRING object [Function]
PRINC-TO-STRING object [Function]
WRITE-CHAR object &optional output-stream [Function]
TERPRI &optional output-stream [Function]
FRESH-LINE &optional output-stream [Function]
WRITE-BYTE integer binary-output-stream [Function]
FORMAT destination control-string &rest arguments [Function]
format directives [Function]
~A ~S ~D ~B ~O ~X ~C ~F ~E ~% ~& ~| ~~ ~<NEWLINE> ~T
-- Deleted items ----------
WRITE object &key :stream :escape :radix :base :circle :pretty :level :length :case :gensym :array
WRITE-TO-STRING object &key :escape :radix :base :circle :pretty :level :length :case :gensym :array [Function]
WRITE-STRING string &optional output-stream &key :start :end [Function]
WRITE-LINE string &optional output-stream &key :start :end [Function]
FINISH-OUTPUT &optional output-stream [Function]
FORCE-OUTPUT &optional output-stream [Function]
CLEAR-OUTPUT &optional output-stream [Function]
format directives [Function]
~nR ~P ~G ~$ ~* ~? ~(~) ~[~] ~;
~{ ~} ~< ~> ~^
-- Comment ----------
-> PRIN1, PPRINT and PRINC are more friendly than generic WRITE.

22.4 Querying the Users
-- Deleted items ----------
Y-OR-N-P &optional format-string &rest arguments
YES-OR-NO-P &optional format-string &rest arguments [Function]
[Function]

23 File System Interface
23.1 File Names
File names can be expressed as strings.
23.1.1 Pathnames
Pathname is deleted.
23.1.2 Pathname Functions
-- Deleted items ----------
PATHNAME pathname
TRUENAME pathname [Function]
PARSE-NAMESTRING thing &optional host defaults [Function]
&key :start :end :junk-allowed
MERGE-PATHNAMES pathname &optional defaults default-version [Function]
DEFAULT-PATHNAME-DEFAULTS [Function]
MAKE-PATHNAME &key :host :device :directory :name [Variable]
:type :version :defaults
PATHNAMEP object [Function]
PATHNAME-HOST pathname [Function]
[Function]

```
1401        PATHNAME-DEVICE pathname
1402        PATHNAME-DIRECTORY pathname                                    [Function]
1403        PATHNAME-NAME pathname                                         [Function]
1404        PATHNAME-TYPE pathname                                         [Function]
1405        PATHNAME-VERSION pathname                                      [Function]
1406        NAMESTRING pathname                                            [Function]
1407        FILE-NAMESTRING pathname                                       [Function]
1408        DIRECTORY-NAMESTRING pathname                                  [Function]
1409        HOST-NAMESTRING pathname                                       [Function]
1410        ENOUGH-NAMESTRING pathname &optional defaults                  [Function]
1411        USER-HOMEDIR-PATHNAME &optional host                           [Function]
1412   ------------------------------------------------------------------------------
1413   23.2 Opening and Closing Files
1414        open filename &key :direction :element-type
1415        keyword parameters                                             [Function]
1416         :direction
1417           :input, :output
1418         :element-type
1419           string-char, unsigned-byte
1420        with-open-file (stream filename (options)*)
1421                       (declaration)* (form)*                          [Macro]
1422    --Deleted items ---------------------------------------------------------
1423        keyword parameters
1424         :direction
1425           :io, :probe
1426         :element-type
1427           signed-byte, character, bit, (mod n), :default
1428         :if-exists
1429         :if-dose-not-exist
1430   ------------------------------------------------------------------------------
1431   23.3 Renaming, Deleting, and Other File Operations
1432        rename-file      file new-name
1433        delete-file      file                                          [Function]
1434        probe-file       file                                          [Function]
1435        file-write-date  file                                          [Function]
1436        file-position    file-stream &optional position                [Function]
1437        file-length      file-stream                                   [Function]
1438    -- Deleted item ----------------------------------------------------------
1439        file-author file                                               [Function]
1440   ------------------------------------------------------------------------------
1441   23.4 Loading Files
1442        load filename &key :verbose :print
1443    -- Deleted items ----------------------------------------------------------
1444        keyword parameter                                              [Function]
1445         :if-dose-not-exist
1446        *load-verbose*
1447   ------------------------------------------------------------------------------
1448   23.5 Accessing Directories                                          [Variable]
1449        DIRECTORY filename &key
1450    -- Comments ---------------------------------------------------------------
1451        -> A list of strings is returned.                              [Function]
1452   ------------------------------------------------------------------------------
1453   24. Errors
1454   24.1 General Error-Signalling Function
1455        ERROR formal-string &rest args
1456        WARN format-string &rest args                                  [Function]
1457        BREAK &optional format-string &rest args                       [Function]
1458    -- Deleted items ----------------------------------------------------------
1459        *BREAK-ON-WARNINGS*                                            [Function]
1460        CERROR continue-format-string error-format-string &rest args   [Function]
1461    -- Comments ---------------------------------------------------------------
1462        -> The three functions, error, warn, and break are enough for PCs.
1463   24.2 Specialized Error-Signalling Forms and Macros
1464    -- Deleted items ----------------------------------------------------------
1465        CHECK-TYPE place typespec &optional string                     [Variable]
1466        ASSERT test-form [((place*)) [string (arg)*]]                  [Function]
1467        -> These are not so necessary.                                 [Macro]
1468        ETYPECASE keyform ((type (form)*))*                            [Macro]
1469        CTYPECASE keyplace ((type (form)*))*                           [Macro]
1470        -> These are not so necessary and can be implemented very easily
1471           using TYPECASE.
1472        ECASE keyform ((((key}*) | key) (form)*))*                     [Macro]
1473        CCASE keyform ((((key}*) | key) (form)*))*                     [Macro]
1474        -> Same reason as E/CTYPECASE
1475     ASSERT test-form [((place*)) [string (arg)*]]                     [Macro]
1476        -> These are not so necessary.
1477        ETYPECASE keyform ((type (form}*))*
1478        CTYPECASE keyplace ((type (form)*))*                           [Macro]
1479   ------------------------------------------------------------------------------
1480   25. Miscellaneous Features
1481   ------------------------------------------------------------------------------
1482   25.1. The Compiler
1483       COMPILE name &OPTIONAL definition                               [Function]
1484       COMPILE-FILE input-pathname &KEY :output-file                   [Function]
1485   -- Deleted item ----------------------------------------------------------
1486       DISASSEMBLE name-or-compiled-function                           [Function]
1487   ------------------------------------------------------------------------------
1488   25.2. Documentation
1489   -- Deleted item ----------------------------------------------------------
1490       DOCUMENTATION symbol doc-type                                   [Function]
1491       -- doc-type is one of followings.
1492            variable
1493            function
1494            structure
1495            type
1496            setf
1497   -- Comment ----------------------------------------------------------------
1498       This function is very useful. But if there is not this function, we will
1499            see the manual. And this function deletion will reducing space of lisp
1500            system.
```

```
1501  ------------------------------------------------------------------------
1502  25.3. Debugging Tools
1503      TRACE (function-name)*                                        [Macro]
1504      UNTRACE (function-name)*                                      [Macro]
1505      STEP form                                                     [Macro]
1506      TIME form                                                  [Function]
1507      DESCRIBE object                                            [Function]
1508      ED &OPTIONAL x                                             [Function]
1509      APROPOS string &OPTIONAL package                           [Function]
1510      APROPOS-LIST string &OPTIONAL package                      [Function]
1511  -- Comment ----------------------------------------------------------------
1512      We think the common lisp is a system. So implementor have to deliver
1513          these debugging tools with Common Lisp.
1514      Following functions may be defined.
1515          GC ... Performs garbage collection.
1516          EXIT ... Exit from lisp system if it can.
1517  -- Deleted Items -------------------------------------------------------
1518      INSPECT object                                             [Function]
1519      ROOM &OPTIONAL x                                           [Function]
1520      DRIBBLE &OPTIONAL pathname                                 [Function]
1521      We think these functions are not so important with ordinary debugging.
1522
1523  ------------------------------------------------------------------------
1524  25.4. Environment Inquiries
1525  ------------------------------------------------------------------------
1526  25.4.1. Time Functions
1527      GET-DECODED-TIME                                           [Function]
1528      INTERNAL-TIME-UNITS-PER-SECOND                             [Constant]
1529      GET-INTERNAL-RUN-TIME                                      [Function]
1530      GET-INTERNAL-REAL-TIME                                     [Function]
1531      SLEEP seconds                                              [Function]
1532  -- Comment ----------------------------------------------------------------
1533      RUN-TIME and REAL-TIME may be equal on some kind of the system that can't
1534          calculate CPU time only.
1535  -- Deleted items -------------------------------------------------------
1536      GET-UNIVERSAL-TIME                                         [Function]
1537      DECODE-UNIVERSAL-TIME universal-time &OPTIONAL time-zone   [Function]
1538      ENCODE-UNIVERSAL-TIME second minute hour date month year
1539          &OPTIONAL time-zone                                    [Function]
1540      Function GET-DECODED-TIME will be enough, we think. So the concept of
1541          universal time is deleted.
1542  25.4.2. Other Environment Inquiries
1543      LISP-IMPLEMENTATION-TYPE                                   [Function]
1544      LISP-IMPLEMENTATION-VERSION                                [Function]
1545      SHOT-SITE-NAME                                             [Function]
1546      LONG-SITE-NAME                                             [Function]
1547      *FEATURES*                                                 [Variable]
1548  -- Comment ----------------------------------------------------------------
1549      If lisp read the return value of LISP-IMPLEMENTATION-VERSION, we want
1550          reader return number.
1551  -- Deleted items -------------------------------------------------------
1552      MACHINE-TYPE                                               [Function]
1553      MACHINE-VERSION                                            [Function]
1554      MACHINE-INSTANCE                                           [Function]
1555      SOFTWARE-TYPE                                              [Function]
1556      SOFTWARE-VERSION                                           [Function]
1557      Function LISP-IMPLEMENTATION-TYPE and LISP-IMPLEMENTATION-VERSION will be
1558      enough.
1559  ------------------------------------------------------------------------
1560  25.5. Identity Function
1561      IDENTITY object                                            [Function]
1562  -- Comment ----------------------------------------------------------------
1563      This function is the default value of :KEY keyword parameter.
```