LIST

LIST (FIRST=n, LAST=n, PREFIX=name) *list name* [=] *list of variable names* ;
or
LIST (DROP) *list name* *list of variable names* ;
or
LIST (PRINT, DELETE) *list of listnames* ;

Function:

LIST gives a single name to a list of TSP variables for use later in the program wherever that list of variables is needed. It provides a convenient way to handle long lists of variables repeatedly used. After you define a list, any time it appears in a command (except within equation code), the contents of the list replace the listname. Lists may also be subscripted or lagged. PROC arguments are also treated as lists (the same type of replacement occurs).

Usage:

The LIST name can be any legal TSP variable names; followed by any number of legal TSP variable names (they may include lags or leads) including the names of other lists.

Lists may be nested indefinitely, that is, the names following the LIST name may be LISTs. The only limitation on length is that on the ultimate list of variables. LISTs may not be defined recursively -- the list name cannot appear in its own list when a list is first defined. An example of an illegal recursive definition is LIST LL A B LL; (if LL has not already been defined as a list).

A LIST of variables may be specified as a range, i. e., VAR1-VAR20. A list like this is interpreted as VAR1, VAR2, VAR3, ..., VAR19, VAR20. The range also works for numbers (1-20) and lags (X(-1)-X(-20)). Such an implicit list can also appear directly in a command (see the DOT command below). You can combine this type of list with a DOT loop to operate on individual elements of the list very conveniently:

```
LIST VARLIST VAR1-VAR20 ;
DOT 1-20 ;
------- computations on VAR.  -------
ENDDOT ;
------- operations on VARLIST -------
```

This enables you to use the variables as a group or as individual variables in GENR (since listnames are not allowed in GENR, SET, or equation specifications).

Options:

DELETE/**NODELETE** deletes existing lists.

DROP/**NODROP** drops variables from an existing list.

FIRST= starting integer for a sequence of names in a list. The default is 1. FIRST can be larger than last, to make a list in decreasing order. Cannot be negative.

LAST= ending integer for a sequence of names in a list. The default is 1, and it cannot be negative.

LIST

PREFIX= name to be used as the prefix in constructing a list in the form: namefirst-namelast. See the examples. PREFIX can also be used to append a list of different names to a common prefix.

PRINT/**NOPRINT** prints the contents (i.e. included names) of existing lists.

Examples:

A few simple examples:

```
LIST EQ EQ1-EQ4 ;
LIST INSTVAR C TIME LM G ;
LIST ENDOGVAR GNP CONS I R LP ;
LIST ALLVARS INSTVAR ENDOGVAR ;
LIST LAGS X(-1)-X(-20) ;
LIST STATES 1-50 ;
```

The list ALLVARS consists of the series C, TIME, LM, G, GNP, CONS, I, R, and LP.

The following example shows the power of implicit lists when combined with DOT loops to eliminate repetitive typing:

```
PRINT PAT72-PAT76 RND72-RND76 ;
PARAM A72-A76 DELT72-DELT76 BETA ;
DOT 72-76 ;
     FRML EQ. PAT. = EXP(A.+BETA*RND.+DELT72*RND72 + DELT73*RND73 +
          DELT74*RND74 + DELT75*RND75 +DELT76*RND76) ;
     PARAM A. 1.0 DELT. ;
ENDDOT ;
LSQ (NOPRINT,STEP=BARDB) EQ72-EQ76;
```

In this example, the series, equations, and parameters for a panel data model (5 years of data on each of several hundred units) can all be referred to by their LIST names to save the repetitive typing of each year's variables. Obviously, LISTs defining more than one variable cannot be used within equations, because they do not reduce to algebraic expressions, but it is useful in some applications to use EQSUB in a DOT loop to reproduce similar equations.

Suppose you do not know the number of variables in a LIST until runtime; you can use the options to construct a variable length list in this case:

```
BEGYR=72 ;  ENDYR=76 ;
LIST(PREFIX=EQ,FIRST=BEGYR,LAST=ENDYR) EQS ;
```

creates a list called EQS consisting of EQ72, EQ73, EQ74, EQ75, and EQ76.

More simple examples:

```
LIST(FIRST=5,LAST=1) DECR;          ? 5 4 3 2 1
LIST(DROP)  DECR 3 2;               ? 5 4 1
LIST(PREFIX=B)  BS X Y Z(-1);       ? BX BY BZ(-1)
LIST BSMID BS(2);                   ? Y   (Subscripted list)
PRINT BS(-1);                       ? X(-1) Y(-1) Z(-2)    (Lagged list)
LIST(PRINT) BS;                     ? prints the names X Y Z(-1), but not their values
LIST(DELETE) BS;                    ? removes the list definition for BS
```

Examples of lists which do not contain variable names:

```
LIST STRING “A title string”;
TITLE STRING;                         ? same as TITLE “A title string”;

LIST OPTS MEAN=5;
RANDOM(OPTS) X;                       ? same as RANDOM(MEAN=5) X;
```

Output:

LIST produces no printed output. A TSP variable list is stored in data storage. SHOW LIST; can be used to see the currently defined lists, and LIST(PRINT) can be used to view the contents of one or more lists.