LFind Obtain the address and length of a cell's data

#include <Lists.h> **List Manager Package**

```
void        LFind(offset, len, theCell, theList );
short       *offset ;       receives offset from start of list data
short       *len ;          receives length of cell data
Cell        theCell ;       cell to query
ListHandle  theList ;       handle leading to a ListRec
```

LFind is an alternative to **LGetCell**. It obtains the length of a cell's contents and a value you can use to calculate the address of that data.

offset is the address of a short integer. Upon return, it contains the offset from the start of the data area of *theList* at which a cell's data may be found (or -1 if *theCell* is invalid). Using this value requires quadruple indirection (see the examples, below).

len is the address of a short integer; upon return, it contains the length of the data associated with *theCell* (or -1 if *theCell* is invalid).

theCell specifies the cell whose data you wish to access.

theList is a handle leading to a variable-length ListRec structure. It is a value previously obtained via **LNew**.

Returns: none

Notes: The data area of a list is identified by a handle stored in the cells field of the ListRec. That handle leads to an unstructured array, no larger than 32K, containing the cell contents in no special order.

Here's a version of the code you can use to access the data:

```
DataHandle dh;                      /* defined in Lists.h */
DataPtr    dp;
char       *cp;
short      offset, len;

LFind( &offset, &len, theCell, theList );
if ( offset == -1 ) {/* . . . bad value in theCell , otherwise. . . */}

dh = (*theList)->cells;             /* get the handle */
dp = *dh;                           /* address of start of data area */
cp = (char *) dp;                   /* coerce to char * */
cp += offset;                       /* now cp  points to data of theCell
*/
if (*cp == 'X' ) { /*... etc ...*/ }    /* compare, examine, print, etc.*/
```

Here's the terse version of the final 5 lines above:

```
cp = ( **(*theList)->cells ) + offset;
```

```
if ( *cp == 'X' ) {... etc ...}          /* compare, examine, etc.*/
```

If you just want to draw the text, you can use the following:

```
DrawText( **(*theList)->cells, offset, length );
```

Note: the entire list data area can move around unless you lock it down; e.g.,

```
HLock( (*theList)->cells );
```

Furthermore, the storage for any individual cell can be moved around within the data area by such functions as **LSetCell**, **LDelRow**, etc. Therefore: **LFind** should NOT be used to pre-calculate pointers to cell data; the only safe time to use such cell data addresses is directly after calling **LFind**.

The **LGetCell** function may be easier to use. But since it copies cell data to a local buffer, it is inevitably slower than examining the data directly.