Management of lists (types `list_T` and `listitem_T` from vim) was changed in https://github.com/neovim/neovim/pull/7708/. There is a lint against the "old" usage, but here is a list (pun not intended) of the most important changes.

Declarations for the table

- `list_T list` : a list
- `listitem_T li` : an item of `list`
- `int val` a value for `lv_copyID`

Old	New	Comment
`list->lv_first`	`tv_list_first(list)`	
`list->lv_last`	`tv_list_last(list)`	
`li->li_next`	`TV_LIST_ITEM_NEXT(list, li)`	To be avoided if possible, must use list which li belongs to.
`li->li_prev`	`TV_LIST_ITEM_PREV(list, li)`	To be avoided if possible, must use list which li belongs to.
	Suggestion by @ZyX-I:	Use `TV_LIST_ITER` or indexing instead of the previous two calls.
`list->lv_len`	`tv_list_len(list)`	
`list->lv_lock`	`tv_list_locked(list)`	
`&li->li_tv`	`TV_LIST_ITEM_TV(li)`	
`list->lv_refcount++`	`tv_list_ref(list)`	
`val = list->lv_copyID`	`val = tv_list_copyid(list)`	
`list->lv_copyID = val`	`tv_list_set_copyid(list, val)`	
`for (li = list->lv_first; li != NULL && another_cond; li = li->li_next) code`	`TV_LIST_ITER_CONST(list, li, { if (another_cond) {break;} code})`	Use `TV_LIST_ITER(...)` if you need to modify list items (note: assigning copyID is also modification and this happens always when recursively traversing a list).

For more details and some more advanced usage, the doxygen documentation on `typval.h` and `typval.c` .