Letters to the Editor

On Word Combinations

Dear Editor:

Mandalay Grems' recent article on word combination (Prac-niques, January 1963) was timely and interesting, but I wonder how several of the words in her list are to be pronounced. For example, autoindex (autoyndex?), perunit (perrunit?), preedit (?), pseudoop (soodupe?).

She says that we should have rigid spelling and combination rules, but she makes no attempt to give us any. Perhaps I might make a few suggestions which will meet aesthetic as well as coldly scientific requirements?

1. Spelling should be as given in the Concise Oxford or Webster's dictionary (one or the other exclusively, the first entry given in the dictionary to be used).
2. Accepted combinations should not be hyphenated, e.g. make-shift.
3. The two words to be combined should be hyphenated when
   (a) the first word ends with a vowel with which the second word begins,
   (b) the first word is (i) a verb or participle, (ii) a noun or adjective,
   (c) pronunciation is made more obvious where neither (a) nor (b) is applied.

auto-abstract
autocode
auto-index

back-up
back-space
boot-strap
branch-point
break-point
built-in

card-feed
card-punch
check-bit
check-digit
check-out
check-point
check-sum
cross-foot
cut-off

debug
decode
delimit
disable

enable
encode

face-down
face-up
fall-back
flow-chart

flow-line

gang-punch
gray-code
group-mark
hard-ware

inline
input
interface
interfix
interlace
interleave

key-board
key-punch
key-word

look-out
look-at
look-up

misfeed
multi-address
multiplex
multipoint
multiprocess
multiprogram
multisequence

nondestructive
nonequivalence
noneraseable

non-numeric
nonvolatile

offline
offpunch
offset
outline
output
overflow
overlap
overlay
overpunch
overview
overwrite

patch-board
percent
per-unit
piezo-electric
pin-board
pin-feed
plug-board
plug-in
postedit
postmortem
preanalysis
prearrange
pre-edit
prefix
preselect
pseudocode
pseudo-op
push-down
push-up

read-in
read-out
real-time
rearrange
rerun
reset
restart
rewind
rewrite
round-off

scale-factor
set-name
snap-shot
soft-ware
stand-by

throughput
thruput (deprecated as being ugly)
time-out
turn-around

unconditional
underflow
unlock
unpack
update

wait-out
word-length
word-mark
wrap-around

Miss Grems' word combinations will now read as in the accompanying list.

There can be no good reason for the dropping of the hyphen, since it takes very little time to write, type or set. The tendency for it to disappear in handwritten material is mainly a tendency to laziness in the writer.

Arthur S. Radford
Tarnock Farm,
Tarnock, Axbridge,
Somerset, England

On the Reference Counter Method

Dear Editor:

The reference counter method for solution of the erasure problem in list processing, as proposed by Collins[1] and incorporated in KLS[2] and SLIP[3] by Weizenbaum, is unsatisfactory. In the reference count system, each list carries a count of the number of references to the list in the system; i.e., this number is the number of words which point to the list. When a new word is created which refers to a list, the reference counter for that list is incremented by one; similarly, when any such word is destroyed, the reference counter of the list is decreased by one. When the reference counter becomes zero for a list the list itself is erased. Note that the reference counter for a list must be changed each time a word containing the name of the list is created, covered over, or destroyed.

If a list is used as a sublist within its own structure—that is, if any list structure is circular—the counter for that list can never become zero. This means that there is always the danger that entire list structures can take valuable memory space and yet be unaccessible within the KLS system. This holds also for SLIP.

Mr. Weizenbaum recognized this difficulty, and in his article[2] he mentioned the solution, as follows: "An operation is provided which determines whether placing the name of a particular list on a list structure will result in circularity. If the answer is yes [to the question of circularity], then that name is put on the structure nonresponsibly and the corresponding counter is not tallied." This operation must involve tracing the entire structure in question, for there is no other way to discover a circularity. Clearly, each time a word containing the name of any list L is inserted into a list, the entire structure of L must be traced, in order to determine whether to tally the counter. This necessitates the use of so much extra machine time that the system becomes prohibitively expensive; for this reason KLS and SLIP provide no such safeguard against the loss of circular structures.

A thorough revision of KLS has been made by the author of this note; this language, KLS II, avoids the difficulties here pointed out in KLS and SLIP, essentially by an application of McCarthy's garbage collection method. KLS II extends and streamlines KLS in other ways as well. A write-up is forthcoming.

J. Harold McBeth
General Electric Co., TEMPO
Santa Barbara, Calif.

[1] Collins, G. E. A method for overlapping and erasure of lists. *Comm. ACM 3* (Dec. 1960), 655–657.

[2] Weizenbaum, J. Knotted list structures. *Comm. ACM 5* (Mar. 1962), 161–165.

[3] Weizenbaum, J. Symmetric list processor. General Electric Co., Computer Dept., Sunnyvale, Calif., Feb. 1963; also to appear in *Comm. ACM 6* (Sept. 1963).