Correspondence

To the Editor
The Computer Journal

Sir
In a recent article in this *Journal* (Vol. 17, No. 2) Presser and Benson imply that 'arithmetic expression in logical IF' and 'illegal statement label in arithmetic IF' should *both* be detectable error types in FORTRAN. However, the mistaken inclusion of an arithmetic expression in a logical IF statement, for example IF(A – B)GOTO 10, transforms that statement into an arithmetic IF statement with an error of the form 'illegal statement label in arithmetic IF'. No compiler could be expected to determine which of the two mistakes the programmer had actually made, but the latter diagnostic message might be expected.

In the same way the statement IF(A . EQ . B)10, 20, 30 for example would probably give rise to the diagnostic message 'illegal statement following logical IF' rather than 'logical expression in arithmetic IF', but Presser and Benson do not explicitly consider this particular case.

Yours faithfully,
S. E. WALLIS

Division of Computing
Thames Polytechnic
Wellington Street
London SE18 6PF
22 November 1974

To the Editor
The Computer Journal

Sir
As one who learnt PL/1 before any Algol, and who is now deeply involved in teaching Algol 68, I should like to comment on S. H. Valentine's excellent comparison of these two languages.

The default processes in PL/1 are so obscure that they often defeat sophisticated programming; practically anything compiles and practically nothing runs as expected. Most PL/1 programs really seem to be FORTRAN or COBOL (perhaps this is a plus ?).

The conclusion of the article contradicts itself in stating that Algol 68 looks well in publications, but is harder to read than PL/1. Bold and ordinary type are much easier to read than endless capitals.

Finally, the Revised Report has been available for some time, and it is a pity that the examples were not taken from it.

Yours faithfully,
A. LEARNER

Mathematical Sciences
Queen Mary College
University of London
London E1 4NS
26 November 1974

To the Editor
The Computer Journal

Sir
We sympathise with Mr. H. N. Coates' call for more business papers to be submitted to *The Computer Journal*. In the same issue as Mr. Coates' letter (Vol. 17, No. 1, February 1974) appeared Algorithm 81, Dendrogram Plot. Although, as we have since discovered, in the biological world this is a familiar topic, it was completely foreign to us. However we were able to recognise the pictures and Dendrograms are what we would call 'Family trees.' It seems that the routines we have developed may be of use in Numerical Taxonomy. How often is lack of interest caused by inability to interpret another discipline's terminology? This then is an industrial application of a similar technique to Dendrogram plots.

FAMILY TREES OF COMPONENT PART NUMBERS

At Rolls-Royce (1971) Limited Derby we are generating every week several hundred new part numbers or new applications of existing parts for our gas turbine engines. Certain components undergo many changes over the period of several years that it takes to develop the engine. Each change is instructed as the deletion or addition of features to a previously designed part number e.g. by altering the angle of incidence of a blade part number A, we create part number B. To keep track of the features embodied in every part number of the high change rate components we draw 'Family Trees' of the design progression.

In November 1972 we implemented a system on an IBM 1130 to plot family trees of components from cards in random order containing the following data:

New Part Number, Modification Number, Replaced Part Number.

The method adopted was the result of only a short analysis but proved to be sufficient as only minor modifications have been introduced. One additional requirement arose that caused a new routine to be inserted in the program, this requirement was that where a modification is applied to more than one previous part number then the new part numbers are to appear in a line i.e. part number siblings to be at the same level.

PROCESS

Read cards into table A.

Routine 1 — Find base part number (level 1) i.e. part without a replaced part number.

Routine 2 — From the base part number compute the levels and numbers of sons for each part number. Insert these into tables B and C.

Routine 3 — Starting with level 2 Insert dummy elements in table A so that all applications of a modification are on the same level. (This step may be omitted if mod lining up is not required.)

Routine 2

lowest level reached—no (return to Routine 3)

Routine 4 — Starting with the lowest level insert the reverse levels into table D.

Routine 5 — Starting with the longest chain insert chain numbers into table E. In reverse level sequence allocate chain numbers. Where two or more are of equal length take the chain with least width i.e. least number of sons.

Routine 6 — From tree width and levels compute which of four layouts will fit an A4 or A3 size sheet.

Plots chains in sequence.

See Fig. 1 for an example of typical output.

Recently the Air Transport Association have ruled that where modifications (post introduction of the engine into airline use) to a component exceeds six then a family tree must be published with the Modification Bulletin to the airline. The system is written in Fortran