whether or not the candidate being eliminated is the highest choice remaining on that voting paper. This redistribution must be done only if the candidate being eliminated is the leading preference on that ballot paper and in that case his vote goes to the next preference, *if any*. This means that the whole loop from 5199 to 1150 can be simplified, as one has only to go through the votes once per candidate to transfer the right number.

Third, the redistribution procedure is applied whether or not there are alternative choices on the ballot paper. Subroutine EXTR has to be modified to give JUMP an initial value of 0, which is returned if no element of INP lies in the required range, as is possible when looking at ballot papers after several candidates have been eliminated. There is no compulsion to vote for the required number of candidates or even the right number of vacancies. On exit from EXTR it is then necessary to test the value of JUMP, in case it is 0.

Fourth, as a consequence of the last point, the absolute majority criterion has to be modified on each cycle, to be one more than half the votes still taking part.

It is perhaps worth pointing out that inspection of Tran van Hoa's original data already shows that his subroutine gives false counts. When filling the first vacancy, the votes of candidate A are transferred to candidate F after 6 counts. According to the preference matrix in Table 1 votes 14, 15, and 20 did not vote for candidate F at all and so their voters for other candidates cannot be transferred to him. His final tally should only be 17 instead of 20. Similarly, on the second vacancy counting, when A's votes are transferred to E it is not possible to transfer votes 14 and 20, so that E's maximum vote in this case would be 18. With a different random number generator and the original data I found that candidates were elected in the order F, C, D, A, with final totals of 17, 11, 11 and 20 respectively. This is different from van Hoa's claim of F, E, D, C and also from the manual example of Table 2, which elects in the order A, D, C, B, with 20, 11, 12 and 20 votes.

Perhaps I should add that I found out these things, as well as the points about RANF (especially that it cycled—on unwanted values!) and those noted by Parker and other points on efficiency, the hard way. I tried using the subroutine to count votes in an election, with 11 candidates and 5 vacancies, before their remarks were published. A rewritten version of the subroutine is being submitted separately.

P. A. Samet
Computer Centre
University College London
19 Gordon Street
London WC1H 0AH

Algorithm supplement—Statement of Policy

A contribution to the Supplement may consist of an Algorithm, a Note on a previous algorithm, or an item under the heading of Correspondence.

Algorithms must be submitted in one of the standard programming languages, namely ALGOL 60 (1), ALGOL 68 (2), FORTRAN (3), COBOL (4).

Algorithms in ALGOL 60 or ALGOL 68 must be in form of self-contained procedures. Algorithms in FORTRAN must be self-contained sub-programs. Algorithms in COBOL must be self-contained SUBROUTINEs with parameters or SECTIONs of the PROCEDURE DIVISION and DATA DIVISION. Self-contained means that the algorithm must not use any non-local identifiers other than standard function names, or any COMMON areas. Input/output will normally be through formal parameters, but where standard input/output functions are provided in the language, these are permitted.

The algorithm must be written for publication in the appropriate reference language, and preceded by an appropriate Author's Note. It must be submitted in duplicate and be typewritten double-spaced. Where material is to appear in bold-face it should be underlined in black. Where the appropriate character does not exist on a typewriter, it should be inserted neatly by hand in black and not be replaced by a similar composite character (e.g. $\leqslant$ should not be inserted as $\leq$)

An algorithm must be accompanied by a driver program incorporating it, test data and test results Moreover the tests must be carefully explained on a separate sheet, and all test documentation submitted in duplicate. Test documentation may be in a dialect of the language applicable to a particular computer and have been prepared on the editing equipment of that computer. It may be necessary to ask for paper tape or card source decks in order to check the test documentation independently, but this should not be sent in the first instance.

The Author's Note should include theory of the method, with references, and also explain any tests used to verify the algorithm.

The algorithm which follows the Note should have an opening comment section which briefly defines the parameters used. Comments should be used wherever it is appropriate to clarify the logic. The algorithm must be syntactically correct, produce the results claimed and use computer time and space as efficiently as possible. Avoid constructions whose results may depend on the compiler used (e.g. $y := x + f(x)$ where $f(b)$ is a function which alters the value of b). Cases of failure should be clearly anticipated by the use of appropriate label exits, and commented. Approximate numerical constants must be written as constants and given correct to 15 decimal digits, where appropriate.

Algorithms may be submitted which are translations of published material, but all conditions above must be met. The original Author's permission, where possible, must be obtained in writing and submitted to the Editor, along with the written permission of the copyright holder. Test documentation in such cases must include the original algorithm tests.

Every effort is made to see that published algorithms are completely reliable. In particular all algorithms are submitted to independent referees and extensively checked, so that certifications are not required. However Notes or Correspondence which point out defects in or suggest improvements to previously published algorithms are welcomed. To help in preventing printing mistakes, galley proofs will be sent to authors where possible.

Whilst every effort is made to publish correct algorithms, no liability is assumed by any contributor, the Editor or *The Computer Journal* in connection therewith.

In the event of the formation of a National Algorithm Library, all algorithms which have appeared in the Computer Journal will be made available to this Library.

References

PROGRAMMING LANGUAGE ALGOL, ISO/R/1538.
REPORT ON THE ALGORITHMIC LANGUAGE ALGOL 68. (1969). *Numerische Mathematik*, Vol. 14, pp. 79-218.
PROGRAMMING LANGUAGE FORTRAN, ISO/R/1539.
PROGRAMMING LANGUAGE COBOL, ISO/R/1989.
Documents 1, 3 and 4 above may be obtained from: British Standards Institution, Sales Branch, 101 Pentonville Road, London N.1.

Editor: R. F. Shepherd, Computing Centre, Chelsea College (University of London), Pulton Place, London SW6 5PR.