siderable annoyance to find that much of one's previous programming effort is of no further use, because the original programming language is not implemented at the new university or that the implementation, although allegedly of the same language, differs in important aspects. The same problem exists if a university has more than one computer available to its staff, or if it replaces a machine. Can we do no more than shrug our shoulders and say 'Well, that's life'? I would claim that agreed language standards are of the utmost importance to university users. By the same token, purely local languages—the provision of which seems to be a favourite university pastime—should not be used as the main working language of a university computing centre.

Just as important as the language compatibility is a certain amount of hardware compatibility. There is considerable interchange of programs and experimental data between workers at different universities. This makes it important for us to have ways of passing this information in machine readable form. Often this means compatible magnetic tapes but it also shows the need for compatible input/output equipment. The present jungle of mutually incompatible codes for cards and paper tape is to no-one's advantage and is totally unnecessary. It is also extremely costly, as new data preparation equipment may be needed when a university brings another machine into operation. For example, London University currently has Atlas, IBM 7094, IBM 360 and I.C.T. 1905 computers in operation, and will shortly add a CDC 6600. Four incompatible card codes will be in use, including three of the new 64-character sets. As all of these machines are available to users within the university there is considerable waste in having equipment that cannot punch cards for all the machines.

6. Concluding comments

I have tried to outline the characteristics of the computing load in a typical university and what software is needed to cope with them. In my opinion, manufacturers do not at present fill this need and so it is left to universities to write much of their own software. I suggest, however, that it is the manufacturers' own interest to look carefully at what the universities want. It is at college that young men and women get their first real introduction to computers. The impression they carry away with them from college can be most important in later years, when they are in a position to influence the installation of machines.

References

Irons, E. T. (1965). A rapid turnaround multiprogramming system, *Comm. ACM*, Vol. 8, p. 152.

Lynch, W. C. (1966). Description of a high capacity, fast turnaround university computing centre, *Comm. ACM*, Vol. 9, p. 117.

Rosen, S., Spurgeon, R. A., Donnelly, J. K. (1965). PUFFT—The Purdue University Fast FORTRAN Translator, *Comm. ACM*, Vol. 8, p. 661.

Shantz, P. W., German, R. A., Mitchell, J. G., Shirley, R. S. K., Zarnke, C. R. (1967). WATFOR—The University of Waterloo FORTRAN IV Compiler, *Comm. ACM*, Vol. 10, p. 41. (This describes an earlier version of the compiler, not that for the System/360.)

Correspondence

To the Editor
The Computer Journal

Some computational notes on the shortest route problem

Sir,

There is a minor mistake in the procedures in the paper by Aarni Perko, published in the April 1965 issue of this *Journal.*

With regard to Bellman's optimisation principle, the iterative application of the procedures at each step gives an optimal solution for the nodes already considered. In general, in each iterative cycle the length of the route to a node i is calculated in an optimal way. If, however, the network contains isolated nodes, i.e. nodes which are not attainable from any other node, it can then be the case that after some iterations no nodes will be assigned to the array p. Considering this, the last two instructions with the data of the preceding iterative cycle will be executed wrongly.

In both procedures the lines

```
d[jm] := x; p[jm] := im;
```

should be replaced by

```
if x < 99 999 999 then
begin d[jm] := x; p[jm] := im;
end;
```

Provided that the arrays p and d are declared from $0 - n$ instead of $1 - n$, the additional instruction

```
jm := 0;
```

is sufficient.

As far as the computational refinement is concerned, it is useful as well to introduce an auxiliary variable for loc $[i + 1] - 1$ in the **for** clause.

Yours faithfully,

Uwe Pape

334 Wolfenbüttel,
Fontaneweg 4,
Germany.
3 February 1968

(*Further correspondence appears on pp. 172 and* 194)