I began writing these specifications believing that it would take no more than 10 hours. I now estimate that about 150 hours were spent, over 9 months, in choosing the style of presentation, discovering the required functions and specifying the behaviour of the program. (The time spent in writing this paper is not included in the estimate.) In contrast, the original program was designed, written and tested in a total of 30 hours. Two revised versions of the program, eliminating many 'minor bugs' in the original, were written during the development of the specifications. A correctness proof of the last version appears in a companion paper.[4] I estimate that the two revisions were done in 20 hours. Thus in my experience, the effort required in specifying a program I thought I understood well was 3 to 4 times more than that required in designing and writing it. I believe that this factor would have been considerably higher if I had less training in this field.

One wonders if the low level specifications could have been written without a certain program in mind, or if they are needed at all. I did have a certain program in mind, and perhaps some of the inelegance is due to this fact. However, writing down these specifications exhibited the subtle errors and inelegant ways of the program that escaped my attention before. In contrast, the writing of high level specifications helped only to explain to others how this program indents. It was important to write the low level specifications because these defined how invalid input would be dealt with. Indeed, half the correctness proof of the indenting program consist of showing that low level specifications coincide with high level specifications.

It is not unfair to say that few practising programmers would be comfortable with the level of formalism used here. While there is certainly scope for improving the notations used in the paper, I believe there will be significant loss of precision with any further decrease in the level of formalism and rigour. The complexity of our specifications, however, truly reflects the complexity of any program meeting them.

This experience has been both delightful and frustrating, at times. I recommend that everyone who writes programs conduct similar experiments as often as possible. As such experimenters are well aware, specifications can and often do contain bugs just as programs do.

REFERENCES

1. P. Grogono, 'On layout, identifers and semicolons in Pascal programs', *SIGPLAN Notices*, **14**(4), 35–40 (1979).
2. H. Ledgard, A. Singer and J. Hueras, 'A basis for executing Pascal programmers', *SIGPLAN Notices*, **12**(7), 101–105 (1977).
3. A. Sale, 'Stylistics in languages with compound statements', *The Australian Computer Journal*, **10**(2), 58–59 (1978).
4. P. Mateti and J. Jaffar, 'A correctness proof of an indenting program' *Software—Practice and Experience*, **13**, (3) (1983) (to be published).
5. P. Mateti, 'Documentation of program *indent*: a model for the complete documentation of computer programs', unpublished class notes, Department of Computer Science, University of Melbourne, Parkville 3052, Australia, 1980.
6. B. H. Liskov and V. Berzins, 'An appraisal of program specifications', in P. Wegner (ed.), *Research Directions in Software Technology*, M.I.T. Press, Massachusetts, pp. 276–301, 1979.
7. N. Wirth, 'Syntax of Pascal in extended BNF', *Pascal News*, **12**, 52–53 (1978).