Dear Tony, dearest Jill, and other people, dear or not,

I would like all of you to realize that this is almost certainly Tony's last retirement because, now he is back in the folds of industry, he is all but certain to stay there until his dying breath. But when we address him then for the last time, at the side of his grave, when he can no longer hear what we say about him, tradition requires --"De mortuis nil nisi bene" and all that-- that we say nice things only. This creates obligations for now, as it means that this is our last opportunity to be nasty. But have no fear: to help me, Tony once wrote me a list of techniques by which I could make my pronouncements less offensive. It is for you to judge how successful a teacher he has been.

If I may, I would like to compare the two of us for a moment, because our professional lives have been somewhat intertwined. The first time we met was in 1961 and neither of us remembers it. It was at a lecture series on ALGOL 60 at Brighton. I can be forgiven that I don't remember Tony for I was one of the few speakers and he was only a participant in the audience. His amnesia is harder to forgive, but he has explained it to me by the presence of Peter Naur, the other bearded speaker with a continental accent, claiming that in his memory Peter and I had merged into one person. Clearly he was not impressed. My personal explanation is that Tony's attention was distracted by the presence in the audience of a charming lady by the name of Jill Pym, whom I don't remember either.

The overwhelming similarity between the two of us is the absolutely flimsy ground on which people have been willing to grant us our fame, viz. a juvenile programming exercise and a bon mot. In my case they were a graph algorithm and a disparaging remark about program testing, in Tony's case they were Quicksort and the remark that "there are two ways of constructing a software design: One way is to make it so simple that there are obviously no deficiencies and the other way is to make it so complicated that there are no obvious deficiencies". (End of comparison.)

A major question to be answered is how Tony could become such a successful professor of computing while his formal education --the usual Oxford stuff of those days-- was so defective: economics, some applied probability theory and an unhealthy dose of philosophy, the unscientific non-discipline par excellence. There are four answers to this question.

The first answer, of course, is that as an academic he did not succeed at all, that he just posed as a scholar, while deep in his heart he was no more than an industrialist, a politician and a manager. Indeed, in the form of Elliott Brothers and Microsoft, industry has formed the alpha and omega of his professional life. This would have been forgivable if, in between, he would have been a faultless member of the academic community, but during his Belfast & Oxford years, besides being director of the laboratory, he has been so heavily involved in IBM's PL/I, in the DoD's Ada and with Inmos in Occam, that it is a safe assumption that he could not combine this with his true professorial duties.

The second answer is that his seemingly impressive scientific output was not his own but was written by his friends. He has always been a great master in sending highly original but also totally immature drafts for debugging to a bunch of his cronies, who would then rewrite one out of every five sentences. He is also --as a true Anglo-Saxon-- an extremely iterative text composer, who would often circulate five successive versions before official publication of the final text; you may draw your conclusions. As you may have noticed, I essentially stopped correcting his writings in the mid-eighties.

The third answer is that, years before his Belfast appointment, he had already earned his educational laurels by the design of ALGOL W , which for a decade has been the preferred language for introductory programming courses at the most enlightened American universities. ALGOL W's distinguishing feature was the incorporation of the "record", a data-structuring device Tony had invented. When he sent me his proposal in 1965, I tested it with a new coding of the shortest-path algorithm, and the new text was a great improvement over its predecessors, an improvement which was enhanced by the absence of goto statements. We agreed with each other's simplification efforts. I think that correspondence marks the moment after which we viewed each other as colleagues.

The fourth answer is that his defective Oxford education forced him to be an autodidact for the rest of his life. That has required much intellectual effort and a lot of hard work, but has had one valuable consequence: he grew up without a scientific model to emulate and thus became as scientist absolutely unique.

This is not the place to give a complete enumeration of all the varied work he claims to have done, but we can give some idea of the influence he has had, i.e., of how he has impacted the world (as we Americans say), and also of how he has totally failed to have any noticeable influence.

Let us focus on some noticeable influence first, and let me state right at the beginning that it is widely regarded to have been, if not downright disastrous, at least very detrimental to the health of mathematics. I am referring to a development for which Tony may not be solely responsible but certainly can be considered the main cluprit, and which, for lack of a better name, I shall refer to as "calculational mathematics". It started with his 1969 paper on the axiomatic basis for computer programming, which presented programming as a mathematical task in which the predicate calculus played an essential, central role. The mathematicians at the time knew no more about the predicate calculus than that it was something logicians talked about and were in general horrified by the suggestion that it, or even should become an indispensable tool for daily reasoning. A confession of mine may underscore how revolutionary these thoughts were at the time. For one or two years , the paper on the axiomatic basis failed to excite me and when I then saw Tony's correctness proof of the program FIND, my reaction was this ballet of symbols was not my cup of tea! Obviously, tea is an acquired taste, and some mathematicians never get it. Tony, however, liked tea years before the others.

But Tony knew full well what he was doing and can pride himself on having been rejected by the establishement. One of his letters contains a long and infamous quotation from Paul Halmos, arguing the foolishness of this type of calculational mathematics, but unrepenting he introduced the relational calculus into computing science, and next designed a more calculational approach to category theory, thereby again horrifying a section of the mathematical world. But all this was done without ill feelings: after a bitter complaint of mine about the obscurity of mathematical writings, he reprimands me on 29 Dec 1983 with "You should have more sympathy with mathematicians -- they find their subject as hard to understand as you do.".

So much for influence he did have; let me now give you an example of a lot of hard work done by Tony that had no effect at all: that was whenever he tried to get others to keep their language designs clean and simple. The first example

dates from the 60s: he has been unable to prevent ALGOL 68 from being inflicted upon Mankind. (Fortunately, Mankind turned out to be rather resistant.) Then, for about 10 years, I think, he has been involved in trying to get PL/I clean enough for standardization, but to no effect, for the irrevocable harm had already been done. (In this connection, I am happy to report that PL/I has died: we now get graduate students who have never heard of PL/I. There seems to be some progress.) And finally, out of a sense of public duty, he has spent years trying to protect the DoD from making its overambitious mistakes. Again to no effect, and he suffered from that, as is shown in his letter of 25 March 1978, pointing out why IRONMAN, the next version of the requirement specifications for Ada, was so "misguided and dangerous":

> "You cannot improve a compromise by asking it to meet more exactly the irreconcilable objectives which it was trying to compromise between. And especially not if you add a few more irreconcilable objectives! Oh dear, how can we ever make them see sense?".

Well, we couldn't. The quoted cri de coeur was at the bottom of p. 6; on p. 3 he had explained why Philosophy had been his favourite subject in Oxford.

This was in the part of the letter that was concerned with the work of George Spencer Brown. As by now you may be suspecting, the range of topics covered and moods reflected is quite wide. For instance, on 16 July 1977 he writes:

> "I hope you had a pleasant holiday and that you are continuing to make a good recovery from it.".

while 3 years later he writes --on p. 8 , after 7 pages of technical material--

> "Last week I got a nice telephone call from the chairman of the Turing Award Committee, and I accepted.".

I must confess that I am not quite sure about my rights to quote from his letters to me: the law seems to be that, while the letters are mine, the copyright

is his. So, for safety's sake I take my final, somewhat longer quotation from one of Tony's public manuscripts. I shall first read the quotation and then tell you when it was written.

"So it is among the prophets of doom that I wish to enrol myself. I believe that the current situation in software design is bad, and that it is getting worse. I do not expect that this will be recognised by the designers, manufacturers, sellers, or even the users of software, who will regard the increase in complexity as a sign of progress, or at least an inevitable concomitant thereof; and may even welcome it as a tribute to their intelligence, or at least a challenge. The larger manufacturers are not only set in their ways, but they have actually profited from the increase in complexity of their software, and the resulting decrease in the efficiency and effectiveness of their customers' use of hardware. And everything is now so complicated, that any particular attack on the problem of low quality software design can always be evaded by appeals to tradition, to standards, to customer prejudice, to compensating advantages, and even to promises to mend the fault in future issues.

So it will be a long time before there is even any recognition of the problem which faces our profession. But even if the problem were widely recognized and deplored, its solution is going to present extremely difficult technical problems. The pursuit of complexity is easy, and the implementation of complexity can safely be delegated to competent managers. But the pursuit of simplicity is one of the most difficult and challenging activities of the human mind. Progress is likely to be extremely slow, where each complexity eliminated must be hailed as a breakthrough. We need not only brilliance of intellect but breadth of experience, nicety of judgement, excellence of taste, and even more than our fair share of good luck. And finally we need a puritanical rejection of the temptations of features and facilities, and a passionate devotion to the principles of purity, simplicity and elegance."

Tony wrote the above in 1974, and it reveals that already 25 years ago, he had a clear vision of the shape of the science that Academia would have to create, notwithstanding the somewhat discouraging circumstance that many would consider that creation a superfluous and futile exercise.

But we can also read these paragraphs differently. We can read them as a very personal statement of what he intended to do with the rest of his professional life, and then we can only admire his courage and his dedication, and marvel at the accuracy of his prediction: he has indeed displayed "a passionate devotion to the principles of purity, simplicity and elegance". Tony earns our deeply felt admiration and gets it, however not without the simultaneously expressed, equally deeply felt gratitude to Jill, who has made all this possible by sharing with him the joys (and sorrows) of Life.

prof.dr Edsger W.Dijkstra
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712 - 1188
USA