Degree-Day Speech.

The Hatfield Polytechnic

23 November 1985

C. A. R. Hoare.
Twenty years ago I was working as a programmer for a small British Computer Manufacturer, Elliott Brothers of Borehamwood in Hertfordshire. My main achievement was to lead an implementation of a new high-level programming language ALGOL 60 for that manufacturer's best-selling computer, the Elliott 803. One of our customers was the Hatfield College of Advanced Technology, the predecessor of your illustrious Polytechnic. When the machine had been delivered, but before it had been commissioned on site, one of your students tried to run a program he had written. It did not work.
Not a very surprising outcome.

In those days it was quite common for computers not to work until they had been properly commissioned. And in those days, and perhaps even now, it was quite common that a student's first program might contain a programming error. But no; the program was quite correct, and the error persisted even when it was run on other working computers of the same make. So suspicion naturally fell, as it would now, on the manufacturer's software, and particularly the ALGOL 60 compiler, for which I was responsible.
But no; further investigation showed
gave the compiler a clean bill of health.
Attention then turned to the output routines.
The trouble was that the routine would quite happily obey the command

\[ \text{PRINT (3.0)} \]

but if it was asked to

\[ \text{PRINT (5.0 - 2.0)} \]

it would respond that this was

\[ \text{NOT A NUMBER}. \]

This error message indicated that the floating point number was not properly normalised, and the attempt to print it was a symptom of a programming error, like failing to initialise a variable.
But no; this was not a programming error. It was an error in the very design of the floating point unit. The original logic diagrams were fetched from their filing cabinets and examined by the original logic designer, who by that time had been promoted to be technical manager of the computer manufacturer. He found what appeared to be the correct logic for renormalising every number after every arithmetic operation. But on top of the original design, he found that in his own handwriting he had inserted a special logical circuit which prevented renormalisation in certain circumstances. But for the life...
to its source—a logic design error committed many years before.

The problem was carried out by the student computer program. After the student had been solved, it took exactly two weeks, from the original problem to its solution, just not normalized.

Well, the answer given by the software, "the hardware wasn't really wrong," was just not normalized. So the easiest solution was to change the software, which was just not normalized.

So it was much too risky to change the hardware. What was it to be done?
I tell this story now because it is a true and interesting one, and it brings it to light, and it brings to light a link between you and me, past. But on an occasion like this, I am expected to do more than just reminisce; I should be giving you moral advice and encouragement, that will be useful to you in your professional careers. What moral can I draw from this story?

The first moral is for all of you who are engineers: always write up and maintain full documentation for all your design decisions and designs, especially when you change your mind. I expect you have been told this often enough by your tutors and instructors; and nothing I can say will prevent you forgetting it as soon as you can.
As you may have realised, my own speciality is software, so the obvious next moral that I would like to draw is that hardware is unreliable and software is so sound and reliable that it has to cover up for errors in hardware design. But I'm afraid this wouldn't be moral, because it wouldn't even be true. One day I hope it will be; and those of you who are graduates in Information Technology will help to make it true, and graduates of other disciplines, the users of our software which use computers will insist that they do that we on our programs must be correct.
The true moral of the story I believe is even more important than that, and applies to all of us, no matter what discipline we have studied, and (profession we pursue) afterward. The fact is that in those days, none of us had any formal education or qualification for the jobs we were doing. The only qualified engineer—except the engineer who designed the logic—was a result we had to learn our jobs on the job, and take an interest in everything going on around us as well. So when a problem arises which cannot be pinned down to a single discipline, we were able to pursue it and solve it effectively and very quickly. The man who pursued the problem in my story left school after A-levels. But he took a great interest in everything...
both hardware and software. To this day
he still does—he is Managing Director
of Britain's second largest Computer Manufacturer.

All of you who have earned degrees
today—I congratulate you on your learning
and your success. You will start on
your professional careers with a great advantage.
But I hope you will regard your
advantage not just in what you have
learnt but also in the methods by
which you have learnt it. Maintain
your interest in learning not only to do
your own work, but also to find out
what else is going on round you, in
your Company, your Country and in
the World. Now is the time to
to take an interest in those options.
which you rejected as a student, and embark on
further study of subjects which you thought were-
beyond your capabilities. Regard it always
as your pleasant duty to know more-
than you need to know. You may do
this, if you wish, in ambition for promotion;
perhaps even to be Managing Director of your Company
but the real reward is greater Than that-
you will maintain your interest in life
at work and outside work, and go in
the fascinating discoveries and advances
in modern sciences and technology,