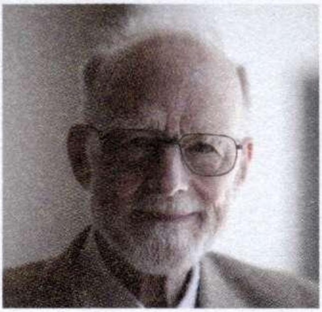
50 Years of Computing at the University of Colombo



A Message of Congratulation
from
Professor Sir Tony Hoare
FRS FREng MA DSc(mult).
University of Oxford

Sir Charles Antony Richard Hoare FRS FREng is a pioneer in the truce sense of the word, in the field of Computer Science and is well known for his development of the sorting algorithm quick-sort and Hoare logic, a first order logic for verifying program correctness, and the formal language communicating sequential processes as a concurrent process algebra and the inspiration for the occam programming language. Born in Colombo, Ceylon to British parents, Tony Hoare's father was a colonial civil servant and his mother was the daughter of a tea planter. He led the Programming Research Group at Oxford University as the Professor and is now an Emeritus Professor there, and is also a principal researcher at Microsoft Research in Cambridge, England. In 1982, Hoare was elected a Fellow of the Royal Society and in 2005 a Fellow of the Royal Academy of Engineering. He became the recipient of the ACM Turing Award (known as the Nobel Prize in Computer Science) in 1980 and in 2011, the IEEE John Von Neumann Medal, and numerous other awards of honour.

Hearty congratulations to the University of Colombo School of Computing on the fiftieth anniversary of the delivery of the first lecture to the University on computer programming.

As a native of Sri Lanka, I much regret that I am not with you on this happy occasion. I was born in 1934 at the Fraser Nursing Home in city of Colombo, and spent my happy childhood there. To celebrate my 70th birthday, I returned to my birthplace in Sri Lanka in 2004 with my children and grandchildren. It was a memorable trip, including a pilgrimage to the top of Sri Pada at dawn.

My family left Sri Lanka in 1945 to return to their own country of origin, England. I was educated here, and received and MA in 1956 in the Humanities, including the Latin and Greek languages, together with their classical literature; the ancient history of Rome and Greece; and finally, Philosophy, both classical and modern.

Since my schooldays, my favourite academic subject has been Philosophy, and it still is. My special interest is in the study of Human Knowledge (Epistemology). This knowledge is acquired both by inductive reasoning which learns from experience and observation of the real world, and by deductive reasoning which derives true and useful conclusions from what has been learnt. Any theory of Knowledge must surely include both these aspects.

I therefore studied statistics and probability theory as the foundation for inductive logic, and formal deductive logic as the basis of mathematical reasoning. When I heard about computers, and I was fascinated by the prospect that they could be capable of simulating important aspects of Human Intelligence. That was why in 1960 I accepted an offer of employment as a programmer from a small British computer manufacturer. I thus obtained on-the-job experience in computer programming and programming languages, in their compilers and in operating systems, which exploit and allow concurrent operation of many programs.

My first major assignment was to lead a small team to implement a compiler for ALGOL 60, a programming language designed in early 1960 by an international committee of scientists. The language specification included an elegant formal definition of the correctness of the syntax of programs written in the language. This stimulated my interest in formalisation of the meaning (aka semantics) of the language, defining precisely the behaviour of a program when executed by computer.

Then the behaviour of each program written in the language could be predicted correctly by deductive reasoning. As a result, the correctness of a program could be proved with mathematical certainty. I realised that achievement of such a goal was a very long-term prospect, one that would occupy me beyond the date of my retirement, which was still thirty-three years ahead. There was no prospect of the research finding practical application in industry within that timescale.

These considerations led me in 1967 to apply for a Professorship in Computing Science at the Queen's University in Belfast. My first research paper was entitled 'An Axiomatic Basis for Computer Programming. From then until now I have worked consistently at exploring the foundations of a Theory of Programming, following the example of the foundations of mathematics, explored by the great logicians early in the last century. Next year is the fiftieth anniversary of start of my lifetime's research in this field.

What about the next fifty years? What are the prospects for further development of Computing and of Computer Science? My prediction is for a dramatic increase in the current rate of progress of our research, and in the scale of its beneficial application. Current indications are much more favourable than they were fifty years ago.

For example, verification of critical properties of critical parts of widely used programs has been recognised as essential in all computer applications where a single software error after delivery and installation is completely unacceptable. Such applications abound in many industries – financial, transport, nuclear, defence, space, security, entertainment, and above all in the software industry itself. The urgent need for absolute reliability is driving the evolution of powerful mechanical proof assistants, which have already displayed their power by assisting the proof of longstanding conjectures in Mathematics, for example the four-colour problem and the Kepler conjecture.

More generally, the cost of software errors made by programmers during the development of programs is being dramatically reduced by exploitation of Integrated Software Development Environments. These are already widely used in the software industry to detect many common errors by symbolic execution at compile time, even before the first unit tests are conducted. Automatic test case generators are used to evoke further errors. Finally, advanced display systems will permit rapid identification of the places where the program needs to be corrected.