Preface

Inspirations

There are many "popular" books on science that provide accessible accounts of the recent developments of modern science for the general reader. However, there are very few popular books about computer science – arguably the "science" that has changed the world the most in the last half century. This book is an attempt to address this imbalance and to present an accessible account of the origins and foundations of computer science. In brief, the goal of this book is to explain how computers work, how we arrived at where we are now, and where we are likely to be going in the future.

The key inspiration for writing this book came from Physics Nobel Prize recipient Richard Feynman. In his lifetime, Feynman was one of the few physicists well known to a more general public. There were three main reasons for this recognition. First, there were some wonderful British television programs of Feynman talking about his love for physics. Second, there was his best-selling book *"Surely You're Joking, Mr. Feynman!": Adventures of a Curious Character*, an entertaining collection of stories about his life in physics – from his experiences at Los Alamos and the Manhattan atomic bomb project, to his days as a professor at Cornell and Caltech. And third, when he was battling the cancer that eventually took his life, was his participation in the enquiry following the *Challenger* space shuttle disaster. His live demonstration, at a televised press conference, of the effects of freezing water on the rubber O-rings of the space shuttle booster rockets was a wonderfully understandable explanation of the origin of the disaster.

Among physicists, Feynman is probably best known for Feynman diagrams, the work that brought him his Nobel Prize in 1964. The diagrams constitute a calculational tool kit that enables physicists to make sense of not only Quantum Electrodynamics, the theory that underpins electricity and magnetism, but also of the relativistic quantum field theories believed to describe the weak and strong interactions of elementary particles. But Feynman was not only a great researcher: his *Feynman Lectures on Physics* are a masterly three-volume introduction to modern physics based on lectures that he gave at Caltech in the 1960s. He was also a visionary: Feynman's after-dinner talk "There's Plenty of Room at the Bottom" in 1959 first introduced the ideas of nanotechnology – the behavior of devices at length scales approaching atomic dimensions.

By the early 1980s, Feynman had become interested in computing, and for the last five years of his life, he gave a lecture course on computing. In the first two years, he collaborated on an ambitious computing course with two Caltech colleagues, Carver Mead and John Hopfield. In the third year, assisted by Gerry Sussman, a computer scientist on sabbatical from MIT, Feynman gave his own version of the course. The lectures contained a fascinating mixture of standard computer science material plus discussion of the thermodynamics of computing and an analysis of a quantum computer. Before he died, Feynman asked Tony Hey to edit his notes for publication, and the lectures eventually saw the light of day as *The Feynman Lectures on Computation*. Feynman also acted as a consultant to the Thinking Machines computer company, founded by MIT researcher Danny Hillis.

Feynman's introductory lectures on quantum mechanics were the inspiration for *The New Quantum Universe*, an earlier popular science book by one of the present authors. Feynman's computing lectures seemed to invite the creation of a similar popular treatment. Feynman had also given an introductory talk on computers in his "Computers from the Inside Out" lecture at the Esalen Institute – a "holistic learning and retreat center" – in Big Sur, California. In this talk he explained the essential working of a computer using the analogy of a "very dumb file clerk." Thus Feynman's lectures on computing were the original inspiration for our attempt at a popular book on computer science.

There were several other sources of inspiration for this book. One was *The Soul of a New Machine* by Tracy Kidder. Although that book is about the design and construction of a new mini-computer, it reads like a thriller – and even has a chapter titled "The Case of the Missing NAND Gate." Another inspiration was the book *The State of the Art*, a pictorial history of Moore's law by computer historian Stan Augarten. It is another book by Augarten, *Bit by Bit* – an illustrated history of computers, from calculating machines to the personal computer – that is closest in spirit to the present book. Other inspirations were *Algorithmics* by the Israeli computer scientist David Harel, originally given as a series of radio lectures, and *The Pattern in the Stone* by the computer architect Danny Hillis.

Digital literacy and computer science

At school, we take proficiency in the "3 Rs" – reading, writing, and arithmetic – to be an essential life skill. Now, in addition, we expect that all children should know how to use computers – to produce electronic documents, manipulate spreadsheets, make slide-show presentations, and browse the web. But such basic "digital literacy" skills are not what is meant by the term *computer science*. Computer science is the study of computers – how to build them, understand their limitations, and use their power to solve complex problems. Alan Turing, the English genius who was one of the first to explore these questions, developed a theoretical machine model by imitating how a "human computer" would go about solving a computational problem. Turing machines provide the essential mathematical basis for reasoning about the behavior of computers. But computer science is about more than mathematics; it is also about engineering – building complex systems that do useful things. In computer

engineering we also have the additional freedom to explore virtual systems – complex structures without the limitations of real physical systems.

Computer scientist Jeannette Wing defines *computational thinking* as the ability to use the fundamental concepts of computer science to solve difficult problems, design complex systems, and understand human behavior. She believes that education in computational thinking will be as essential in the twenty-first century as the 3 Rs have been in all previous centuries. Computational thinking includes techniques of abstraction and decomposition that assist in the development of algorithms to attack complex tasks or to design complex systems. It also gives new insights on system concepts such as prevention, protection, and recovery by thinking in terms of corresponding computer science concepts such as redundancy, damage containment, and error correction. Computational thinking can also help apply ideas from machine learning and Bayesian statistics to everyday problems. In many areas of life we are faced with the problem of planning and learning in the presence of uncertainty, and computational thinking ideas applied to "big data" have application in both science and commerce.

How do we instruct a computer to solve a particular problem? First we must write down our *algorithm* – a sequence of steps to the solution rather like a cooking recipe – in a specialized *programming language*. The specific sequence of instructions is called a *program* and this constitutes part of the *computer software* required to solve the problem on the computer. The programming language instructions can then be translated into operations that can be carried out by the low-level components of the *computer hardware*. Running the program requires more software, the *operating system* that manages the input and output of data and the use of storage devices and printers. Programming is the skill required to translate our computer science algorithms into programs that computers can understand. Like digital literacy skills, the ability to program is certainly a vital skill for a future career in the information technology industry but constitutes only a small part of Jeannette Wing's computational thinking agenda.

The goal of this book

Our goal in writing this book is not to produce another textbook on computers, or a book on the history of computing. Rather, the book is intended to be intelligible to both high school and first-year university students and to stimulate their interest in the wide range of opportunities of a career in computer science. We also hope that it will provide general readers with an understandable and accessible account of how computers work, as well as a glimpse of the vast scope of activities that have been enabled by networks of interconnected computers. In order to make the book more readable and entertaining we have included brief biographies and anecdotes about the scientists and engineers who helped create this computing universe.

It is curious that schoolchildren are taught the names and achievements of great mathematicians, physicists, chemists, and biologists but not about the great computer pioneers. In part then, one goal of the book is to make a start at correcting this imbalance by highlighting the contributions of the pioneers

of computing. These include the early theoretical ideas of Alan Turing and John von Neumann as well as the achievements of the first computer engineers such as Presper Ekert and John Mauchly in the United States and Maurice Wilkes and Konrad Zuse in Europe. The story follows the rise of IBM and Digital to the computing legends at Xerox PARC with their incredible Alto computer, created by Alan Kay, Chuck Thacker, and Butler Lampson. In a very real sense, the story of computing follows the evolution of Moore's law and the rise of the semiconductor industry. It was the microprocessor – "a computer on a chip" – that made possible the personal computing revolution with pioneers like Steve Jobs and Steve Wozniak from Apple and Bill Gates and Paul Allen from Microsoft.

If the first thirty years of computers were about using computers for computing, the second thirty years have been about using computers for communicating. The story takes us from the earliest speculations about interactive computing and the Internet by J. C. R. Licklider; to the packet-switching ideas of Paul Baran and Donald Davies; to the ARPANET of Bob Taylor, Larry Roberts, and BBN; to the Internet protocols of Bob Kahn and Vint Cerf. Early ideas about hypertext and linked documents of Vannevar Bush, Ted Nelson, and Doug Engelbart evolved into the now ubiquitous World Wide Web, created by Tim Berners-Lee. Similarly, the PageRank algorithm, invented by Stanford graduate students Sergey Brin and Larry Page, led to the rise of Internet search engines like Google, Bing, and Baidu.

Today, we are able to forecast the weather with reasonable accuracy; access vast amounts of information; talk to anybody over the Internet; play games, collaborate, and share information easily with others; and, if we wish, broadcast our thoughts to the entire world. Already, the opportunities of our present computing universe seem endless, yet we are only at the beginning of what will be possible in the future. According to Turing Award recipient Butler Lampson, the next thirty years will see us enter the *Third Age of Computing* in which computers become able to act intelligently on our behalf. These developments will bring with them serious issues concerning ethics, security, and privacy, which are beyond the scope of this book. Instead, the book ends with a look at some possible computing technologies and computer science challenges for the future.