Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. Auxiliary tasks accompanying and related to programming include analyzing requirements, testing, debugging (investigating and fixing problems), implementation of build systems, and management of derived artifacts, such as programs' machine code. Integrated development environments (IDEs) aim to integrate all such help. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. However, with the concept of the stored-program computer introduced in 1949, both programs and data were stored and manipulated in the same way in computer memory. There are many approaches to the Software development process. For this purpose, algorithms are classified into orders using so-called Big O notation, which expresses resource use, such as execution time or memory consumption, in terms of the size of an input. In 1801, the Jacquard loom could produce entirely different weaves by changing the "program" – a series of pasteboard cards with holes punched in them. As early as the 9th century, a programmable music sequencer was invented by the Persian Banu Musa brothers, who described an automated mechanical flute player in the Book of Ingenious Devices. Following a consistent programming style often helps readability. Popular modeling techniques include Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA). Auxiliary tasks accompanying and related to programming include analyzing requirements, testing, debugging (investigating and fixing problems), implementation of build systems, and management of derived artifacts, such as programs' machine code. Programs were mostly entered using punched cards or paper tape. Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process. Programmable devices have existed for centuries. Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. Computer programmers are those who write computer software. Trade-offs from this ideal involve finding enough programmers who know the language to build a team, the availability of compilers for that language, and the efficiency with which programs written in a given language execute. Readability is important because programmers spend the majority of their time reading, trying to understand, reusing and modifying existing source code, rather than writing new source code. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users. In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages. These compiled languages allow the programmer to write programs in terms that are syntactically richer, and more capable of abstracting the code, making it easy to target varying machine instruction sets via compilation declarations and heuristics. The first computer program is generally dated to 1843, when mathematician Ada Lovelace published an algorithm to calculate a sequence of Bernoulli numbers, intended to be carried out by Charles Babbage's Analytical Engine.