After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. Whatever the approach to development may be, the final program must satisfy some fundamental properties. 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. A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. It is very difficult to determine what are the most popular modern programming languages. Popular modeling techniques include Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA). Normally the first step in debugging is to attempt to reproduce the problem. Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation. Techniques like Code refactoring can enhance readability. The Unified Modeling Language (UML) is a notation used for both the OOAD and MDA. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. Various visual programming languages have also been developed with the intent to resolve readability concerns by adopting non-traditional approaches to code structure and display. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. 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. However, because an assembly language is little more than a different notation for a machine language, two machines with different instruction sets also have different assembly languages. FORTRAN, the first widely used high-level language to have a functional implementation, came out in 1957, and many other languages were soon developed—in particular, COBOL aimed at commercial data processing, and Lisp for computer research. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. Whatever the approach to development may be, the final program must satisfy some fundamental properties. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. Later a control panel (plug board) added to his 1906 Type I Tabulator allowed it to be programmed for different jobs, and by the late 1940s, unit record equipment such as the IBM 602 and IBM 604, were programmed by control panels in a similar way, as were the first electronic computers. Many applications use a mix of several languages in their construction and use. In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. It is usually easier to code in "high-level" languages than in "low-level" ones.