Expert programmers are familiar with a variety of well-established algorithms and their respective complexities and use this knowledge to choose algorithms that are best suited to the circumstances. After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. While these are sometimes considered programming, often the term software development is used for this larger overall process – with the terms programming, implementation, and coding reserved for the writing and editing of code per se. 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. Programming languages are essential for software development. Trial-and-error/divide-and-conquer is needed: the programmer will try to remove some parts of the original test case and check if the problem still exists. The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. 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. Normally the first step in debugging is to attempt to reproduce the problem. The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). Programs were mostly entered using punched cards or paper tape. Techniques like Code refactoring can enhance readability. Programmable devices have existed for centuries. Programming languages are essential for software development. In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages. Code-breaking algorithms have also existed for centuries. One approach popular for requirements analysis is Use Case analysis. The academic field and the engineering practice of computer programming are both largely concerned with discovering and implementing the most efficient algorithms for a given class of problems. Expert programmers are familiar with a variety of well-established algorithms and their respective complexities and use this knowledge to choose algorithms that are best suited to the circumstances. Allen Downey, in his book How To Think Like A Computer Scientist, writes: Many computer languages provide a mechanism to call functions provided by shared libraries. Normally the first step in debugging is to attempt to reproduce the problem. Use of a static code analysis tool can help detect some possible problems.