David McGinnis
CM 2351

CSSE 376 Lab #5

1. The process of TDD involves the continual improvement of code in order to pass unit tests that are written to indicate what the code that will be written should do. These iterations are separated by phases of refactoring of the code, which ensures that the code is as efficient as possible.
2. While I would agree somewhat with the first point, the second point I don't believe is true. TDD only increases confidence of the programmer as long as the programmer can correctly identify all use cases for the piece of code, which may not be reasonable in a large piece of code with many use cases (such as a Compiler or textual interpreter). The code quality could very well be lowered, because of the approach which you take. Many algorithms are freely available online for most processes, and the process which TDD imposes may not be as effective as these algorithms. A good example of this is sorting, where the best general sorting algorithm (Quicksort) is not immediately obvious, and most likely would not be developed in a TDD fashion, unless the developer already knew the algorithm.
3. The major advantage of doing TDD is that you gain a much deeper understanding of the problem you are developing, and ensures that you have a suite of tests which can be run later for regression testing, as well as to help new people to your team understand individual methods, as well as perhaps the interactions between those methods. On the other hand, I think it is easy, as I said above, to get sucked into thinking about the problem in a certain way, which might not be the way which you should attack the given problem. It requires you to think about a problem a certain way, and then make that jump in logic to something completely different to actually get a reasonable answer that doesn't involve special cases, and that jump can be difficult to make if you had any more difficult of a problem you were trying to solve.