Structure

* Title
* Check it is standard EEE coverpage
* Abstract
* Check the format
* Add conclusion part
* Acknowledgements
* John Wickerson add
* Introduction
* Formalize
* Make lay
* Convert project specification
* Background
* Re-read
* Look for more references
* Julia
* Body
* Method
* Environment and test
* Implementation philosophy
* Coverage
* Testing speed
* Implementation
* C draft
* Add line control
* Read the rest and implement
* Add multithreading
* Add functionality bulking
* Add maths optimisations checking
* Add control flow
* Add functions
* Static
* Self-defined
* Conclusion
* Many bugs with code production along way
* Div 0 – Overflow
* The variable definition in body issue **Investigate**
* Maths opt levels
* Why not many differences in opt levels **Investigate**
* Future work
* Smarter value checking
* List features
* Bibliography
* Recheck format (Use numeric [1] or AuthorDate [Cla06] keys)
* Appendix
* User Guide

REMOVED

As mutation testing has shown effective\cite{mutation\_2}, it would also be beneficial to implement some form of mutation-based testing.

BASH SCRIPT FLAG SECTION OR BASH IN GENERAL

As such, a review of the language was mandatory. The first and main source for information regarding the Julia language was the official Julia documentation\cite{Julia-Doc}. However, this documentation has observed flaws. The book “Think Julia: How to Think” like a computer scientist was also used as a reference and learning materials\cite{Julia-Book}. Helpful resources that provide hello world style introductions like Julia by example were used when running some random programs to gain more familiarity with the language\cite{Julia-by-example}. Key areas like File I/O and Random Number generation were then learned as these are fundamentally important for the AST and file generation. The complexities of creating the abstract portions of an AST in Julia were then attempted.