**Aims & background material (student)**

For the project “Random testing of advanced programming language features in compilers”, two main directions are currently being considered. Firstly, I could attempt to extend the existing tool CSmith. Currently CSmith does not support a variety of higher-level aspects of C++ such as strings or floating-point arithmetic. We considered rebuilding CSmith, however, it was thought that this would working towards a redundant goal. To research this possible project direction, I have read the 2011 CSmith paper listed on the CSmith website. I have also downloaded their latest 2.3.0 version of CSmith and have had a shallow look at how it works. The second direction we thought the project could head towards was trying to produce a fuzzing testing program for Julia. We thought that C compilers are likely to have had heavy screening and bug finding is extremely challenging. Whereas, a new language like Julia is likely to have a more unstable compiler with more bugs to find. To research this direction, I have attempted to learn the Julia language and use it in some basic examples to.

**Student Summary of project deliverables, fallbacks & extensions (student)**

Roughly I think it best to spend time in the next couple of weeks examining the CSmith software to analysis how random code is generated. I would then like to begin work developing some random code generation for myself. If I pursue the CSmith extention I think it would be good to have one component working by 2nd term alongside my report. I then would like to have the rest of the code finished before I return for summer term. If I develop in Julia I would like to get some random code generation done by 2nd term and would like to have a basic model that can test for bugs done by the end of 2nd term. For both projects I would prefer to work more heavily in 2nd term.

**Summary of Risks (student)**

For the first project route several risks exist. Firstly, as the structure for extending CSmith is highly modular I could overcommit far too much time on one component. Alternatively, I could find that adding the extra component support is insufficient work and I would be left with no avenues to explore. As for results, I could find that due to the compilers all being highly targeted I can’t find any bugs. Specifically, for the Julia project route there also exists considerable risk. Firstly, as far as I can tell there are not multiple compilers so I would be unable to perform differential testing with different reliable compilers. Thus, I would either have to do differential testing at different compiler optimisation levels or perform mutation-based testing. It is unclear yet if this would yield the same quality of results. I would also have to develop considerably more code as I would be starting from scratch. As such I could find myself without a well performing final project. Finally, due to Julia being a more flexible language than C it might be the case that developing “correct” code is far more difficult.