Resilient Library Evolution Tests

This directory tests for the correctness of *resilience*, which is a broad term for Swift maximizing binary compatibility of a dependency while maintaining the freedom to do things that would normally break clients in other languages, such as changing the layout of nominal types. The detailed explanation of resilience is out of scope of this little readme and can be found in `docs/LibraryEvolution.rst`.

Each main test file should compile against "before" and "after" versions of the corresponding library file. The old and new versions are selected via the BEFORE or AFTER preprocessor variables.

In the library, going from BEFORE to AFTER must be an ABI-compatible change, as documented in the library evolution specification.

In the main program, going from BEFORE to AFTER does not have to be backward compatible.

In the main file, use your test library's `getVersion` helper function to know which version of the library to expect at runtime.

There are four valid combinations for each test:

1. Main file compiled against old library, linked against old library a.k.a. "before before"
2. Main file compiled against old library, linked against new library, a.k.a. "before after"
3. Main file compiled against new library, linked against old library, a.k.a. "after before"
4. Main file compiled against new library, linked against new library, a.k.a. "after after"

The version of the library available at compile time determines which serialized declarations and transparent function bodies are visible.

When adding a new test, see the boilerplate at the top of each file for the general pattern for this kind of test. Use the `StdlibUnittest` library for assertions.