1. Check compiler warnings seen on the following builds:

    a. Linux GCC using options -Wall -Wextra
    b. OpenBSD GCC using options -Wall
    c. Windows VC++

2. Verify that documentation files are up-to-date:

    a. Latest release on the index.html page
    b. Release announcement in changes.in
    c. Release accouncement in news.html
    d. No unresolved hyperlinks in the documentation build
    e. The compile-time options are all up-to-date (compile.html)
    f. Database footprint is up-to-date (features.html)
    g. Test metrics are up-to-date (testing.html)
    h. Check new/changed documentation for spelling and gramatical mistakes
    i. Verify that this checklist agrees with planning documents

3. Review the differences between the release candidate and one or more prior releases to verify:

    a. No stray changes
    b. All significant changes are mentioned in the "changes" log of the documentation
    c. New code complies with style guidelines
    d. New code complies with design rules
    e. Comments have been updated to reflect code changes
    f. Variables and functions have been renamed to reflect changes in their use

4. Verify SQLite compiles cleanly with all OMIT, ENABLE, and DISABLE options:

    a. **tclsh ../tool/omittest.tcl -skip_run**

5. Run the following tests and verify output for platform *Linux x86*:

    a. **tclsh releasetest.tcl**
    b. **tclsh th3make cov.rc** *(verify 100% branch test coverage)*
    c. **tclsh fulltest.tcl fast.rc test.rc**
    d. **tclsh th3make fast-ex.rc**
    e. **tclsh th3make cov.rc -DHAVE_LOCALTIME_R=1** *(verify 100% coverage)*

6. Run the following tests and verify output for platform *Linux x86_64*:

    a. **tclsh releasetest.tcl**
    b. **tclsh fulltest.tcl fast.rc test.rc**
    c. **tclsh th3make memdebug.rc**
    d. **tclsh th3make test-ex.rc**

7. Run the following tests and verify output for platform *Mac OS-X x86*:

    a. **tclsh releasetest.tcl**
    b. **tclsh fulltest.tcl fast.rc test.rc**

8. Run the following tests and verify output for platform *Max OS-X PPC*:

    a. **tclsh th3make -Os min.rc**

9. Run the following tests and verify output for platform *Win32*:

    a. **make fulltest**
    b. **th3make test.rc**
    c. **th3make fast.rc**

10. Run the following tests and verify output for platform *Win64*:

    a. **make fulltest**
    b. **th3make test.rc** *(compiled using VC++)*
    c. **th3make fast.rc**

11. Run the following tests and verify output for platform *Android*:

    a. **th3make -Os min.rc**

12. Run the following tests and verify output for platform *OpenBSD x86*:

    a. **make test**
    b. **th3make -Os min.rc**

13. On one or more platforms of the tester's choice:

    a. **th3make alignment2.rc test.rc**

14. Run test suites on various compilers and at various optimization settings for each compiler and verify that identical results are obtained on each run. This checklist item is designed to verify that SQLite code does not use C code constructs that are undefined or implementation defined. In particular, make sure the code works the same with and without the **-ftrapv** option on GCC. *What other compilers and/or options can we list here? Clang? LLVM?*

15. Verify that everything still works with SQLITE_MAX_ATTACHED in the 50 to 60 range.

16. Update the amalgamation build for sqllogictest. Recompile and run the entire test suite in verification mode. Verify that there are no errors. Check-in the new amalgamation build in the sqllogictest Fossil repository.

17. The veryquick.test TCL tests run under valgrind with no unexplained errors or warnings. Note that some tests deliberately reference memory that has previously been freed which will cause valgrind warnings. Those tests are clearly marked.

18. Run TH3 tests **tclsh th3make -Os min.rc** under valgrind with no unexplained errors or warnings. Some TH3 tests deliberately do things that valgrind will complain about. Those tests are clearly marked.

19. Build using the configure/make in main source tree.

    a. Make sure that autoconf has been run to update the configure script.
    b. Run "make test" to verify that the build works.
    c. Verify the correct version numbers have been installed.
    d. Verify that the --disable-amalgamation option to configure works.

20. Build a library from the amalgamation using both SQLITE_ENABLE_FTS3 and SQLITE_ENABLE_RTREE and verify that no internal symbols are exported.

21. Update the SQLite amalgamation that is built into Fossil. Recompile and run a complete test.

22. Update the SQLite amalgamation that is built into the public Firefox sources. Recompile and verify that the new SQLite is being used (by checking the version mark in the header of one of the Firefox database files) and that Firefox still works correctly and efficiently.

23. Copy the SQLite amalgamation into CEROD. Compile CEROD and run the test scripts in the test/ subdirectory. Verify correct output.

24. Copy the SQLite amalgamation into SEE. Compile and test all five variations of SEE using a command similar to **tclsh th3make see.rc min.rc -Os -DSQLITE_DEBUG -DSQLITE_MEMDEBUG**, with or without the optional switches. Test controllers "test.rc" and "fast.rc" can be substituted for "min.rc"; each should be used at least once.

    a. see.c

    b. see-aes128-ofb.c
    c. see-aes256-ofb.c
    d. see-rc4.c
    e. see-aes128-ccm.c

25. Run the following ZIPVFS tests on at least one platform each:

    a. **testfixture ../zipvfs/test/zipvfs.test** *(See test/README.txt in the ZIPVFS source tree for details.)*
    b. **th3make zipvfs.rc test.rc**
    c. **th3make zipvfs.rc memdebug.rc**
    d. **th3make zipvfs.rc min.rc** *(Run under Valgrind with no unexplained errors)*

26. Build the amalgamation autoconf tarball. On Linux and Mac OS-X, unpack that tarball in a temporary directory and verify that it builds correctly.

27. Build the TEA archive. On Linux and Mac OS-X, unpack that TEA archive into a temporary directory and verify that it builds correctly.

28. Run performance tests comparing the new SQLite release against the previous version. Verify no performance regressions.

    a. The TCL speed*.test scripts
    b. The TH3 speed tests
    c. sqlite3 sqlite.fossil .dump >speed-c.sql
       fossil timeline -R sqlite.fossil -n 300 -sqltrace 2>>speed-c.sql
       fossil rebuild sqlite.fossil -sqltrace 2>>speed-c.sql

29. Verify that the amalgamation builds are byte-for-byte identical on all workstation platforms.

30. Verify that databases (including FTS3 and RTREE databases), rollback journals, and WAL files created by the release candidate are readable and writeable by historical versions of SQLite (as long as no unsupported features are used) and vice versa.

31. Verify that databases, rollback journals, and WAL files are portable across

    a. 32/64 bit platforms, and
    b. little/big-endian platforms.

32. Build and verify correct operation of sqlite3_analyzer on:

    a. Linux
    b. MacOSX
    c. Win32

33. Run the CLI tests found under the tool subfolder in the main source tree and verify correct output.

34. All changes in trunk have been merged into each of the following branches and those branches have been tested using TCL and TH3.

    a. sessions
    b. apple-osx

35. Verify that all branches that are no longer active have been closed.

36. Verify that the bug response checklist has been completed for all bugs found since the previous release.

37. Verify that the new feature checklist has been completed for all enhancements made since the previous release.