Test Assertions Checklist

If you know there is a problem, then write a test for it. Otherwise, don’t bother.

Test anything that can possibly break.

Test core functions.

Avoid multiple asserts.

Only provide as much information as possible to pass test.

Use arrange (your objects), act (on object), assert (something is as expected).

Use a code coverage tool to make sure your tests exercise all the logic paths in your code.

Test getters and setters.

In inheritance hierarchies, test for Liskov Substitution Principle.

Does a value fall within its expected range?

File stream is open or closed?

That a file stream is at the beginning when routine executes?

That a file, class, routine, stream is set to the right access modifier?

That the value of an input variable is not changed by a routine?

That a pointer is non-null?

That an array or other container only contains the X allowed number of elements?

That a table has been initialized to contain real values?

That a container is empty when a routine begins executing?

That the time it takes to run a routine is within performance requirements?

Do methods and properties return expected values?

Are appropriate exceptions thrown when invalid argument is supplied?

Test anything that is broken.

Good values of different types (i.e. positive, negative, zero)

Boundary conditions

Maximum, minimum

Outside of max and min

Gaps in domain (i.e. prime numbers, even numbers, etc).

Delimiter problems (missing or too many)

Mixed case (hello, Hello, HeLlo)

Input is too long for string

Input has white space or other delimiter

File exists and contains correct data

File exists but data is wrong type/format

File exists but is empty

File exists but is corrupt

File does not exist

Boolean values of 0/false, 1/true, and something else (e.g. 7/Hello)

Ensure nested statements are tested thoroughly

Case statements should test all conditions (including ELSE clause)

Ensure entering condition of loop is true

are exit values what are expected?

Is the loop exited at the correct iteration

Loop body executes zero, once, or multiple times

Ordered data structure

First element added/removed

Middle element added/removed

Last element added/removed

Unordered data structure

Empty structure

Single item

Multiple items

Full structure

Duplicate items

Pointers

Nil

Pointer is not nil (i.e. points to object)

Two pointers pointing to same object (e.g. pointers A and B point to object X)

Pointer to a list of multiple objects

Test functional requirements.

Smoke Tests – Test call chain.