Fatal vs non-fatal:

Fatal	Non-Fatal	What it tests
`ASSERT_TRUE(condition);`	`EXPECT_TRUE(condition);`	`condition is true`
`ASSERT_FALSE(condition);`	`EXPECT_FALSE(condition);`	`condition is not true`

Basic assertions:

Fatal	Non-Fatal	What it tests
`ASSERT_EQ(x, y);`	`EXPECT_EQ(x, y);`	`x == y`
`ASSERT_NE(x, y);`	`EXPECT_NE(x, y);`	`x != y`
`ASSERT_LT(x, y);`	`EXPECT_LT(x, y);`	`x < y`
`ASSERT_LE(x, y);`	`EXPECT_LE(x, y);`	`x <= y`
`ASSERT_GT(x, y);`	`EXPECT_GT(x, y);`	`x > y`
`ASSERT_GE(x, y);`	`EXPECT_GE(x, y);`	`x >= y`

Assertions on C strings (char*):

Fatal	Non-Fatal	What it tests
`ASSERT_STREQ(x,y);`	`EXPECT_STREQ(x,y);`	x and y have the same content
`ASSERT_STRNE(x,y);`	`EXPECT_STRNE(x,y);`	x and y have different contents
`ASSERT_STRCASEEQ(x,y);`	`EXPECT_STRCASEEQ(x,y);`	x and y have the same content, ignoring case
`ASSERT_STRCASENE(x,y);`	`EXPECT_STRCASENE(x,y);`	x and y have different contents, ignoring case

Assertions on exceptions:

Fatal	Non-Fatal	What it tests
ASSERT_THROW(some_statement, exceptionType);	EXPECT_THROW(some_statement, exceptionType);	some_statement throws an exception of the **exact** given type
ASSERT_ANY_THROW(some_statement);	EXPECT_ANY_THROW(some_statement);	some_statement throws an exception of **any** type
ASSERT_NO_THROW(some_statement);	EXPECT_NO_THROW(some_statement);	some_statement throws no exception