~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

<exe-name> is a <version> host application.

Run with -? for options

-------------------------------------------------------------------------------

# A test name that starts with a #

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

with message:

yay

-------------------------------------------------------------------------------

#748 - captures with unexpected exceptions

outside assertions

-------------------------------------------------------------------------------

ExceptionTests.cpp:<line number>

...............................................................................

ExceptionTests.cpp:<line number>: FAILED:

due to unexpected exception with messages:

answer := 42

expected exception

-------------------------------------------------------------------------------

#748 - captures with unexpected exceptions

inside REQUIRE\_NOTHROW

-------------------------------------------------------------------------------

ExceptionTests.cpp:<line number>

...............................................................................

ExceptionTests.cpp:<line number>: FAILED:

REQUIRE\_NOTHROW( thisThrows() )

due to unexpected exception with messages:

answer := 42

expected exception

-------------------------------------------------------------------------------

#748 - captures with unexpected exceptions

inside REQUIRE\_THROWS

-------------------------------------------------------------------------------

ExceptionTests.cpp:<line number>

...............................................................................

ExceptionTests.cpp:<line number>:

PASSED:

REQUIRE\_THROWS( thisThrows() )

with message:

answer := 42

-------------------------------------------------------------------------------

#809

-------------------------------------------------------------------------------

CompilationTests.cpp:<line number>

...............................................................................

CompilationTests.cpp:<line number>:

PASSED:

REQUIRE( 42 == f )

with expansion:

42 == {?}

-------------------------------------------------------------------------------

#833

-------------------------------------------------------------------------------

CompilationTests.cpp:<line number>

...............................................................................

CompilationTests.cpp:<line number>:

PASSED:

REQUIRE( a == t )

with expansion:

3 == 3

CompilationTests.cpp:<line number>:

PASSED:

CHECK( a == t )

with expansion:

3 == 3

CompilationTests.cpp:<line number>:

PASSED:

REQUIRE\_THROWS( throws\_int(true) )

CompilationTests.cpp:<line number>:

PASSED:

CHECK\_THROWS\_AS( throws\_int(true), const int& )

CompilationTests.cpp:<line number>:

PASSED:

REQUIRE\_NOTHROW( throws\_int(false) )

CompilationTests.cpp:<line number>:

PASSED:

REQUIRE\_THAT( "aaa", Catch::EndsWith("aaa") )

with expansion:

"aaa" ends with: "aaa"

CompilationTests.cpp:<line number>:

PASSED:

REQUIRE( templated\_tests<int>(3) )

with expansion:

true

-------------------------------------------------------------------------------

#835 -- errno should not be touched by Catch

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>: FAILED:

CHECK( f() == 0 )

with expansion:

1 == 0

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( errno == 1 )

with expansion:

1 == 1

-------------------------------------------------------------------------------

#961 -- Dynamically created sections should all be reported

Looped section 0

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

with message:

Everything is OK

-------------------------------------------------------------------------------

#961 -- Dynamically created sections should all be reported

Looped section 1

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

with message:

Everything is OK

-------------------------------------------------------------------------------

#961 -- Dynamically created sections should all be reported

Looped section 2

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

with message:

Everything is OK

-------------------------------------------------------------------------------

#961 -- Dynamically created sections should all be reported

Looped section 3

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

with message:

Everything is OK

-------------------------------------------------------------------------------

#961 -- Dynamically created sections should all be reported

Looped section 4

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

with message:

Everything is OK

-------------------------------------------------------------------------------

'Not' checks that should fail

-------------------------------------------------------------------------------

ConditionTests.cpp:<line number>

...............................................................................

ConditionTests.cpp:<line number>: FAILED:

CHECK( false != false )

ConditionTests.cpp:<line number>: FAILED:

CHECK( true != true )

ConditionTests.cpp:<line number>: FAILED:

CHECK( !true )

with expansion:

false

ConditionTests.cpp:<line number>: FAILED:

CHECK\_FALSE( true )

with expansion:

!true

ConditionTests.cpp:<line number>: FAILED:

CHECK( !trueValue )

with expansion:

false

ConditionTests.cpp:<line number>: FAILED:

CHECK\_FALSE( trueValue )

with expansion:

!true

ConditionTests.cpp:<line number>: FAILED:

CHECK( !(1 == 1) )

with expansion:

false

ConditionTests.cpp:<line number>: FAILED:

CHECK\_FALSE( 1 == 1 )

-------------------------------------------------------------------------------

'Not' checks that should succeed

-------------------------------------------------------------------------------

ConditionTests.cpp:<line number>

...............................................................................

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( false == false )

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( true == true )

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( !false )

with expansion:

true

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE\_FALSE( false )

with expansion:

!false

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( !falseValue )

with expansion:

true

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE\_FALSE( falseValue )

with expansion:

!false

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( !(1 == 2) )

with expansion:

true

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE\_FALSE( 1 == 2 )

-------------------------------------------------------------------------------

(unimplemented) static bools can be evaluated

compare to true

-------------------------------------------------------------------------------

TrickyTests.cpp:<line number>

...............................................................................

TrickyTests.cpp:<line number>:

PASSED:

REQUIRE( is\_true<true>::value == true )

with expansion:

true == true

TrickyTests.cpp:<line number>:

PASSED:

REQUIRE( true == is\_true<true>::value )

with expansion:

true == true

-------------------------------------------------------------------------------

(unimplemented) static bools can be evaluated

compare to false

-------------------------------------------------------------------------------

TrickyTests.cpp:<line number>

...............................................................................

TrickyTests.cpp:<line number>:

PASSED:

REQUIRE( is\_true<false>::value == false )

with expansion:

false == false

TrickyTests.cpp:<line number>:

PASSED:

REQUIRE( false == is\_true<false>::value )

with expansion:

false == false

-------------------------------------------------------------------------------

(unimplemented) static bools can be evaluated

negation

-------------------------------------------------------------------------------

TrickyTests.cpp:<line number>

...............................................................................

TrickyTests.cpp:<line number>:

PASSED:

REQUIRE( !is\_true<false>::value )

with expansion:

true

-------------------------------------------------------------------------------

(unimplemented) static bools can be evaluated

double negation

-------------------------------------------------------------------------------

TrickyTests.cpp:<line number>

...............................................................................

TrickyTests.cpp:<line number>:

PASSED:

REQUIRE( !!is\_true<true>::value )

with expansion:

true

-------------------------------------------------------------------------------

(unimplemented) static bools can be evaluated

direct

-------------------------------------------------------------------------------

TrickyTests.cpp:<line number>

...............................................................................

TrickyTests.cpp:<line number>:

PASSED:

REQUIRE( is\_true<true>::value )

with expansion:

true

TrickyTests.cpp:<line number>:

PASSED:

REQUIRE\_FALSE( is\_true<false>::value )

with expansion:

!false

-------------------------------------------------------------------------------

A METHOD\_AS\_TEST\_CASE based test run that fails

-------------------------------------------------------------------------------

ClassTests.cpp:<line number>

...............................................................................

ClassTests.cpp:<line number>: FAILED:

REQUIRE( s == "world" )

with expansion:

"hello" == "world"

-------------------------------------------------------------------------------

A METHOD\_AS\_TEST\_CASE based test run that succeeds

-------------------------------------------------------------------------------

ClassTests.cpp:<line number>

...............................................................................

ClassTests.cpp:<line number>:

PASSED:

REQUIRE( s == "hello" )

with expansion:

"hello" == "hello"

-------------------------------------------------------------------------------

A TEST\_CASE\_METHOD based test run that fails

-------------------------------------------------------------------------------

ClassTests.cpp:<line number>

...............................................................................

ClassTests.cpp:<line number>: FAILED:

REQUIRE( m\_a == 2 )

with expansion:

1 == 2

-------------------------------------------------------------------------------

A TEST\_CASE\_METHOD based test run that succeeds

-------------------------------------------------------------------------------

ClassTests.cpp:<line number>

...............................................................................

ClassTests.cpp:<line number>:

PASSED:

REQUIRE( m\_a == 1 )

with expansion:

1 == 1

-------------------------------------------------------------------------------

A couple of nested sections followed by a failure

Outer

Inner

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

with message:

that's not flying - that's failing in style

-------------------------------------------------------------------------------

A couple of nested sections followed by a failure

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>: FAILED:

explicitly with message:

to infinity and beyond

-------------------------------------------------------------------------------

A failing expression with a non streamable type is still captured

-------------------------------------------------------------------------------

TrickyTests.cpp:<line number>

...............................................................................

TrickyTests.cpp:<line number>: FAILED:

CHECK( &o1 == &o2 )

with expansion:

0x<hex digits> == 0x<hex digits>

TrickyTests.cpp:<line number>: FAILED:

CHECK( o1 == o2 )

with expansion:

{?} == {?}

-------------------------------------------------------------------------------

Absolute margin

-------------------------------------------------------------------------------

ApproxTests.cpp:<line number>

...............................................................................

ApproxTests.cpp:<line number>:

PASSED:

REQUIRE( 104.0 != Approx(100.0) )

with expansion:

104.0 != Approx( 100.0 )

ApproxTests.cpp:<line number>:

PASSED:

REQUIRE( 104.0 == Approx(100.0).margin(5) )

with expansion:

104.0 == Approx( 100.0 )

ApproxTests.cpp:<line number>:

PASSED:

REQUIRE( 104.0 == Approx(100.0).margin(4) )

with expansion:

104.0 == Approx( 100.0 )

ApproxTests.cpp:<line number>:

PASSED:

REQUIRE( 104.0 != Approx(100.0).margin(3) )

with expansion:

104.0 != Approx( 100.0 )

ApproxTests.cpp:<line number>:

PASSED:

REQUIRE( 100.3 != Approx(100.0) )

with expansion:

100.3 != Approx( 100.0 )

ApproxTests.cpp:<line number>:

PASSED:

REQUIRE( 100.3 == Approx(100.0).margin(0.5) )

with expansion:

100.3 == Approx( 100.0 )

-------------------------------------------------------------------------------

AllOf matcher

-------------------------------------------------------------------------------

MatchersTests.cpp:<line number>

...............................................................................

MatchersTests.cpp:<line number>:

PASSED:

CHECK\_THAT( testStringForMatching(), AllOf( Catch::Contains( "string" ), Catch::Contains( "abc" ) ) )

with expansion:

"this string contains 'abc' as a substring" ( contains: "string" and

contains: "abc" )

-------------------------------------------------------------------------------

An expression with side-effects should only be evaluated once

-------------------------------------------------------------------------------

TrickyTests.cpp:<line number>

...............................................................................

TrickyTests.cpp:<line number>:

PASSED:

REQUIRE( i++ == 7 )

with expansion:

7 == 7

TrickyTests.cpp:<line number>:

PASSED:

REQUIRE( i++ == 8 )

with expansion:

8 == 8

-------------------------------------------------------------------------------

An unchecked exception reports the line of the last assertion

-------------------------------------------------------------------------------

ExceptionTests.cpp:<line number>

...............................................................................

ExceptionTests.cpp:<line number>:

PASSED:

CHECK( 1 == 1 )

ExceptionTests.cpp:<line number>: FAILED:

{Unknown expression after the reported line}

due to unexpected exception with message:

unexpected exception

-------------------------------------------------------------------------------

Anonymous test case 1

-------------------------------------------------------------------------------

VariadicMacrosTests.cpp:<line number>

...............................................................................

VariadicMacrosTests.cpp:<line number>:

PASSED:

with message:

anonymous test case

-------------------------------------------------------------------------------

AnyOf matcher

-------------------------------------------------------------------------------

MatchersTests.cpp:<line number>

...............................................................................

MatchersTests.cpp:<line number>:

PASSED:

CHECK\_THAT( testStringForMatching(), AnyOf( Catch::Contains( "string" ), Catch::Contains( "not there" ) ) )

with expansion:

"this string contains 'abc' as a substring" ( contains: "string" or contains:

"not there" )

MatchersTests.cpp:<line number>:

PASSED:

CHECK\_THAT( testStringForMatching(), AnyOf( Catch::Contains( "not there" ), Catch::Contains( "string" ) ) )

with expansion:

"this string contains 'abc' as a substring" ( contains: "not there" or

contains: "string" )

-------------------------------------------------------------------------------

Approx with exactly-representable margin

-------------------------------------------------------------------------------

ApproxTests.cpp:<line number>

...............................................................................

ApproxTests.cpp:<line number>:

PASSED:

CHECK( 0.25f == Approx(0.0f).margin(0.25f) )

with expansion:

0.25f == Approx( 0.0 )

ApproxTests.cpp:<line number>:

PASSED:

CHECK( 0.0f == Approx(0.25f).margin(0.25f) )

with expansion:

0.0f == Approx( 0.25 )

ApproxTests.cpp:<line number>:

PASSED:

CHECK( 0.5f == Approx(0.25f).margin(0.25f) )

with expansion:

0.5f == Approx( 0.25 )

ApproxTests.cpp:<line number>:

PASSED:

CHECK( 245.0f == Approx(245.25f).margin(0.25f) )

with expansion:

245.0f == Approx( 245.25 )

ApproxTests.cpp:<line number>:

PASSED:

CHECK( 245.5f == Approx(245.25f).margin(0.25f) )

with expansion:

245.5f == Approx( 245.25 )

-------------------------------------------------------------------------------

Approximate PI

-------------------------------------------------------------------------------

ApproxTests.cpp:<line number>

...............................................................................

ApproxTests.cpp:<line number>:

PASSED:

REQUIRE( divide( 22, 7 ) == Approx( 3.141 ).epsilon( 0.001 ) )

with expansion:

3.1428571429 == Approx( 3.141 )

ApproxTests.cpp:<line number>:

PASSED:

REQUIRE( divide( 22, 7 ) != Approx( 3.141 ).epsilon( 0.0001 ) )

with expansion:

3.1428571429 != Approx( 3.141 )

-------------------------------------------------------------------------------

Approximate comparisons with different epsilons

-------------------------------------------------------------------------------

ApproxTests.cpp:<line number>

...............................................................................

ApproxTests.cpp:<line number>:

PASSED:

REQUIRE( d != Approx( 1.231 ) )

with expansion:

1.23 != Approx( 1.231 )

ApproxTests.cpp:<line number>:

PASSED:

REQUIRE( d == Approx( 1.231 ).epsilon( 0.1 ) )

with expansion:

1.23 == Approx( 1.231 )

-------------------------------------------------------------------------------

Approximate comparisons with floats

-------------------------------------------------------------------------------

ApproxTests.cpp:<line number>

...............................................................................

ApproxTests.cpp:<line number>:

PASSED:

REQUIRE( 1.23f == Approx( 1.23f ) )

with expansion:

1.23f == Approx( 1.2300000191 )

ApproxTests.cpp:<line number>:

PASSED:

REQUIRE( 0.0f == Approx( 0.0f ) )

with expansion:

0.0f == Approx( 0.0 )

-------------------------------------------------------------------------------

Approximate comparisons with ints

-------------------------------------------------------------------------------

ApproxTests.cpp:<line number>

...............................................................................

ApproxTests.cpp:<line number>:

PASSED:

REQUIRE( 1 == Approx( 1 ) )

with expansion:

1 == Approx( 1.0 )

ApproxTests.cpp:<line number>:

PASSED:

REQUIRE( 0 == Approx( 0 ) )

with expansion:

0 == Approx( 0.0 )

-------------------------------------------------------------------------------

Approximate comparisons with mixed numeric types

-------------------------------------------------------------------------------

ApproxTests.cpp:<line number>

...............................................................................

ApproxTests.cpp:<line number>:

PASSED:

REQUIRE( 1.0f == Approx( 1 ) )

with expansion:

1.0f == Approx( 1.0 )

ApproxTests.cpp:<line number>:

PASSED:

REQUIRE( 0 == Approx( dZero) )

with expansion:

0 == Approx( 0.0 )

ApproxTests.cpp:<line number>:

PASSED:

REQUIRE( 0 == Approx( dSmall ).epsilon( 0.001 ) )

with expansion:

0 == Approx( 0.00001 )

ApproxTests.cpp:<line number>:

PASSED:

REQUIRE( 1.234f == Approx( dMedium ) )

with expansion:

1.234f == Approx( 1.234 )

ApproxTests.cpp:<line number>:

PASSED:

REQUIRE( dMedium == Approx( 1.234f ) )

with expansion:

1.234 == Approx( 1.2339999676 )

-------------------------------------------------------------------------------

Assertions then sections

-------------------------------------------------------------------------------

TrickyTests.cpp:<line number>

...............................................................................

TrickyTests.cpp:<line number>:

PASSED:

REQUIRE( Catch::alwaysTrue() )

with expansion:

true

-------------------------------------------------------------------------------

Assertions then sections

A section

-------------------------------------------------------------------------------

TrickyTests.cpp:<line number>

...............................................................................

TrickyTests.cpp:<line number>:

PASSED:

REQUIRE( Catch::alwaysTrue() )

with expansion:

true

-------------------------------------------------------------------------------

Assertions then sections

A section

Another section

-------------------------------------------------------------------------------

TrickyTests.cpp:<line number>

...............................................................................

TrickyTests.cpp:<line number>:

PASSED:

REQUIRE( Catch::alwaysTrue() )

with expansion:

true

-------------------------------------------------------------------------------

Assertions then sections

-------------------------------------------------------------------------------

TrickyTests.cpp:<line number>

...............................................................................

TrickyTests.cpp:<line number>:

PASSED:

REQUIRE( Catch::alwaysTrue() )

with expansion:

true

-------------------------------------------------------------------------------

Assertions then sections

A section

-------------------------------------------------------------------------------

TrickyTests.cpp:<line number>

...............................................................................

TrickyTests.cpp:<line number>:

PASSED:

REQUIRE( Catch::alwaysTrue() )

with expansion:

true

-------------------------------------------------------------------------------

Assertions then sections

A section

Another other section

-------------------------------------------------------------------------------

TrickyTests.cpp:<line number>

...............................................................................

TrickyTests.cpp:<line number>:

PASSED:

REQUIRE( Catch::alwaysTrue() )

with expansion:

true

-------------------------------------------------------------------------------

Capture and info messages

Capture should stringify like assertions

-------------------------------------------------------------------------------

ToStringGeneralTests.cpp:<line number>

...............................................................................

ToStringGeneralTests.cpp:<line number>:

PASSED:

REQUIRE( true )

with message:

i := 2

-------------------------------------------------------------------------------

Capture and info messages

Info should NOT stringify the way assertions do

-------------------------------------------------------------------------------

ToStringGeneralTests.cpp:<line number>

...............................................................................

ToStringGeneralTests.cpp:<line number>:

PASSED:

REQUIRE( true )

with message:

3

-------------------------------------------------------------------------------

Character pretty printing

Specifically escaped

-------------------------------------------------------------------------------

ToStringGeneralTests.cpp:<line number>

...............................................................................

ToStringGeneralTests.cpp:<line number>:

PASSED:

CHECK( tab == '\t' )

with expansion:

'\t' == '\t'

ToStringGeneralTests.cpp:<line number>:

PASSED:

CHECK( newline == '\n' )

with expansion:

'\n' == '\n'

ToStringGeneralTests.cpp:<line number>:

PASSED:

CHECK( carr\_return == '\r' )

with expansion:

'\r' == '\r'

ToStringGeneralTests.cpp:<line number>:

PASSED:

CHECK( form\_feed == '\f' )

with expansion:

'\f' == '\f'

-------------------------------------------------------------------------------

Character pretty printing

General chars

-------------------------------------------------------------------------------

ToStringGeneralTests.cpp:<line number>

...............................................................................

ToStringGeneralTests.cpp:<line number>:

PASSED:

CHECK( space == ' ' )

with expansion:

' ' == ' '

ToStringGeneralTests.cpp:<line number>:

PASSED:

REQUIRE( c == chars[i] )

with expansion:

'a' == 'a'

ToStringGeneralTests.cpp:<line number>:

PASSED:

REQUIRE( c == chars[i] )

with expansion:

'z' == 'z'

ToStringGeneralTests.cpp:<line number>:

PASSED:

REQUIRE( c == chars[i] )

with expansion:

'A' == 'A'

ToStringGeneralTests.cpp:<line number>:

PASSED:

REQUIRE( c == chars[i] )

with expansion:

'Z' == 'Z'

-------------------------------------------------------------------------------

Character pretty printing

Low ASCII

-------------------------------------------------------------------------------

ToStringGeneralTests.cpp:<line number>

...............................................................................

ToStringGeneralTests.cpp:<line number>:

PASSED:

CHECK( null\_terminator == '\0' )

with expansion:

0 == 0

ToStringGeneralTests.cpp:<line number>:

PASSED:

REQUIRE( c == i )

with expansion:

2 == 2

ToStringGeneralTests.cpp:<line number>:

PASSED:

REQUIRE( c == i )

with expansion:

3 == 3

ToStringGeneralTests.cpp:<line number>:

PASSED:

REQUIRE( c == i )

with expansion:

4 == 4

ToStringGeneralTests.cpp:<line number>:

PASSED:

REQUIRE( c == i )

with expansion:

5 == 5

-------------------------------------------------------------------------------

Comparing function pointers

-------------------------------------------------------------------------------

TrickyTests.cpp:<line number>

...............................................................................

TrickyTests.cpp:<line number>:

PASSED:

REQUIRE( a )

with expansion:

0x<hex digits>

TrickyTests.cpp:<line number>:

PASSED:

REQUIRE( a == &foo )

with expansion:

0x<hex digits> == 0x<hex digits>

-------------------------------------------------------------------------------

Comparing member function pointers

-------------------------------------------------------------------------------

TrickyTests.cpp:<line number>

...............................................................................

TrickyTests.cpp:<line number>:

PASSED:

CHECK( m == &S::f )

with expansion:

0x<hex digits>

==

0x<hex digits>

-------------------------------------------------------------------------------

Comparisons between ints where one side is computed

-------------------------------------------------------------------------------

ConditionTests.cpp:<line number>

...............................................................................

ConditionTests.cpp:<line number>:

PASSED:

CHECK( 54 == 6\*9 )

with expansion:

54 == 54

-------------------------------------------------------------------------------

Comparisons between unsigned ints and negative signed ints match c++ standard

behaviour

-------------------------------------------------------------------------------

ConditionTests.cpp:<line number>

...............................................................................

ConditionTests.cpp:<line number>:

PASSED:

CHECK( ( -1 > 2u ) )

with expansion:

true

ConditionTests.cpp:<line number>:

PASSED:

CHECK( -1 > 2u )

with expansion:

-1 > 2

ConditionTests.cpp:<line number>:

PASSED:

CHECK( ( 2u < -1 ) )

with expansion:

true

ConditionTests.cpp:<line number>:

PASSED:

CHECK( 2u < -1 )

with expansion:

2 < -1

ConditionTests.cpp:<line number>:

PASSED:

CHECK( ( minInt > 2u ) )

with expansion:

true

ConditionTests.cpp:<line number>:

PASSED:

CHECK( minInt > 2u )

with expansion:

-2147483648 > 2

-------------------------------------------------------------------------------

Comparisons with int literals don't warn when mixing signed/ unsigned

-------------------------------------------------------------------------------

ConditionTests.cpp:<line number>

...............................................................................

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( i == 1 )

with expansion:

1 == 1

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( ui == 2 )

with expansion:

2 == 2

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( l == 3 )

with expansion:

3 == 3

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( ul == 4 )

with expansion:

4 == 4

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( c == 5 )

with expansion:

5 == 5

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( uc == 6 )

with expansion:

6 == 6

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( 1 == i )

with expansion:

1 == 1

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( 2 == ui )

with expansion:

2 == 2

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( 3 == l )

with expansion:

3 == 3

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( 4 == ul )

with expansion:

4 == 4

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( 5 == c )

with expansion:

5 == 5

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( 6 == uc )

with expansion:

6 == 6

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( (std::numeric\_limits<unsigned long>::max)() > ul )

with expansion:

18446744073709551615 (0x<hex digits>)

>

4

-------------------------------------------------------------------------------

Contains string matcher

-------------------------------------------------------------------------------

MatchersTests.cpp:<line number>

...............................................................................

MatchersTests.cpp:<line number>: FAILED:

CHECK\_THAT( testStringForMatching(), Contains( "not there" ) )

with expansion:

"this string contains 'abc' as a substring" contains: "not there"

-------------------------------------------------------------------------------

Custom exceptions can be translated when testing for nothrow

-------------------------------------------------------------------------------

ExceptionTests.cpp:<line number>

...............................................................................

ExceptionTests.cpp:<line number>: FAILED:

REQUIRE\_NOTHROW( throwCustom() )

due to unexpected exception with message:

custom exception - not std

-------------------------------------------------------------------------------

Custom exceptions can be translated when testing for throwing as something else

-------------------------------------------------------------------------------

ExceptionTests.cpp:<line number>

...............................................................................

ExceptionTests.cpp:<line number>: FAILED:

REQUIRE\_THROWS\_AS( throwCustom(), std::exception )

due to unexpected exception with message:

custom exception - not std

-------------------------------------------------------------------------------

Custom std-exceptions can be custom translated

-------------------------------------------------------------------------------

ExceptionTests.cpp:<line number>

...............................................................................

ExceptionTests.cpp:<line number>: FAILED:

due to unexpected exception with message:

custom std exception

-------------------------------------------------------------------------------

Demonstrate that a non-const == is not used

-------------------------------------------------------------------------------

TrickyTests.cpp:<line number>

...............................................................................

TrickyTests.cpp:<line number>:

PASSED:

REQUIRE( t == 1u )

with expansion:

{?} == 1

-------------------------------------------------------------------------------

EndsWith string matcher

-------------------------------------------------------------------------------

MatchersTests.cpp:<line number>

...............................................................................

MatchersTests.cpp:<line number>: FAILED:

CHECK\_THAT( testStringForMatching(), EndsWith( "this" ) )

with expansion:

"this string contains 'abc' as a substring" ends with: "this"

-------------------------------------------------------------------------------

Equality checks that should fail

-------------------------------------------------------------------------------

ConditionTests.cpp:<line number>

...............................................................................

ConditionTests.cpp:<line number>: FAILED:

CHECK( data.int\_seven == 6 )

with expansion:

7 == 6

ConditionTests.cpp:<line number>: FAILED:

CHECK( data.int\_seven == 8 )

with expansion:

7 == 8

ConditionTests.cpp:<line number>: FAILED:

CHECK( data.int\_seven == 0 )

with expansion:

7 == 0

ConditionTests.cpp:<line number>: FAILED:

CHECK( data.float\_nine\_point\_one == Approx( 9.11f ) )

with expansion:

9.1f == Approx( 9.1099996567 )

ConditionTests.cpp:<line number>: FAILED:

CHECK( data.float\_nine\_point\_one == Approx( 9.0f ) )

with expansion:

9.1f == Approx( 9.0 )

ConditionTests.cpp:<line number>: FAILED:

CHECK( data.float\_nine\_point\_one == Approx( 1 ) )

with expansion:

9.1f == Approx( 1.0 )

ConditionTests.cpp:<line number>: FAILED:

CHECK( data.float\_nine\_point\_one == Approx( 0 ) )

with expansion:

9.1f == Approx( 0.0 )

ConditionTests.cpp:<line number>: FAILED:

CHECK( data.double\_pi == Approx( 3.1415 ) )

with expansion:

3.1415926535 == Approx( 3.1415 )

ConditionTests.cpp:<line number>: FAILED:

CHECK( data.str\_hello == "goodbye" )

with expansion:

"hello" == "goodbye"

ConditionTests.cpp:<line number>: FAILED:

CHECK( data.str\_hello == "hell" )

with expansion:

"hello" == "hell"

ConditionTests.cpp:<line number>: FAILED:

CHECK( data.str\_hello == "hello1" )

with expansion:

"hello" == "hello1"

ConditionTests.cpp:<line number>: FAILED:

CHECK( data.str\_hello.size() == 6 )

with expansion:

5 == 6

ConditionTests.cpp:<line number>: FAILED:

CHECK( x == Approx( 1.301 ) )

with expansion:

1.3 == Approx( 1.301 )

-------------------------------------------------------------------------------

Equality checks that should succeed

-------------------------------------------------------------------------------

ConditionTests.cpp:<line number>

...............................................................................

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( data.int\_seven == 7 )

with expansion:

7 == 7

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( data.float\_nine\_point\_one == Approx( 9.1f ) )

with expansion:

9.1f == Approx( 9.1000003815 )

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( data.double\_pi == Approx( 3.1415926535 ) )

with expansion:

3.1415926535 == Approx( 3.1415926535 )

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( data.str\_hello == "hello" )

with expansion:

"hello" == "hello"

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( "hello" == data.str\_hello )

with expansion:

"hello" == "hello"

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( data.str\_hello.size() == 5 )

with expansion:

5 == 5

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( x == Approx( 1.3 ) )

with expansion:

1.3 == Approx( 1.3 )

-------------------------------------------------------------------------------

Equals

-------------------------------------------------------------------------------

MatchersTests.cpp:<line number>

...............................................................................

MatchersTests.cpp:<line number>:

PASSED:

CHECK\_THAT( testStringForMatching(), Equals( "this string contains 'abc' as a substring" ) )

with expansion:

"this string contains 'abc' as a substring" equals: "this string contains

'abc' as a substring"

-------------------------------------------------------------------------------

Equals string matcher

-------------------------------------------------------------------------------

MatchersTests.cpp:<line number>

...............................................................................

MatchersTests.cpp:<line number>: FAILED:

CHECK\_THAT( testStringForMatching(), Equals( "something else" ) )

with expansion:

"this string contains 'abc' as a substring" equals: "something else"

-------------------------------------------------------------------------------

Exception messages can be tested for

exact match

-------------------------------------------------------------------------------

ExceptionTests.cpp:<line number>

...............................................................................

ExceptionTests.cpp:<line number>:

PASSED:

REQUIRE\_THROWS\_WITH( thisThrows(), "expected exception" )

-------------------------------------------------------------------------------

Exception messages can be tested for

different case

-------------------------------------------------------------------------------

ExceptionTests.cpp:<line number>

...............................................................................

ExceptionTests.cpp:<line number>:

PASSED:

REQUIRE\_THROWS\_WITH( thisThrows(), Equals( "expecteD Exception", Catch::CaseSensitive::No ) )

-------------------------------------------------------------------------------

Exception messages can be tested for

wildcarded

-------------------------------------------------------------------------------

ExceptionTests.cpp:<line number>

...............................................................................

ExceptionTests.cpp:<line number>:

PASSED:

REQUIRE\_THROWS\_WITH( thisThrows(), StartsWith( "expected" ) )

ExceptionTests.cpp:<line number>:

PASSED:

REQUIRE\_THROWS\_WITH( thisThrows(), EndsWith( "exception" ) )

ExceptionTests.cpp:<line number>:

PASSED:

REQUIRE\_THROWS\_WITH( thisThrows(), Contains( "except" ) )

ExceptionTests.cpp:<line number>:

PASSED:

REQUIRE\_THROWS\_WITH( thisThrows(), Contains( "exCept", Catch::CaseSensitive::No ) )

-------------------------------------------------------------------------------

Expected exceptions that don't throw or unexpected exceptions fail the test

-------------------------------------------------------------------------------

ExceptionTests.cpp:<line number>

...............................................................................

ExceptionTests.cpp:<line number>: FAILED:

CHECK\_THROWS\_AS( thisThrows(), std::string )

due to unexpected exception with message:

expected exception

ExceptionTests.cpp:<line number>: FAILED:

CHECK\_THROWS\_AS( thisDoesntThrow(), std::domain\_error )

because no exception was thrown where one was expected:

ExceptionTests.cpp:<line number>: FAILED:

CHECK\_NOTHROW( thisThrows() )

due to unexpected exception with message:

expected exception

-------------------------------------------------------------------------------

FAIL aborts the test

-------------------------------------------------------------------------------

MessageTests.cpp:<line number>

...............................................................................

MessageTests.cpp:<line number>: FAILED:

explicitly with message:

This is a failure

-------------------------------------------------------------------------------

FAIL does not require an argument

-------------------------------------------------------------------------------

MessageTests.cpp:<line number>

...............................................................................

MessageTests.cpp:<line number>: FAILED:

-------------------------------------------------------------------------------

FAIL\_CHECK does not abort the test

-------------------------------------------------------------------------------

MessageTests.cpp:<line number>

...............................................................................

MessageTests.cpp:<line number>: FAILED:

explicitly with message:

This is a failure

MessageTests.cpp:<line number>:

warning:

This message appears in the output

-------------------------------------------------------------------------------

Factorials are computed

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( Factorial(0) == 1 )

with expansion:

1 == 1

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( Factorial(1) == 1 )

with expansion:

1 == 1

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( Factorial(2) == 2 )

with expansion:

2 == 2

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( Factorial(3) == 6 )

with expansion:

6 == 6

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( Factorial(10) == 3628800 )

with expansion:

3628800 (0x<hex digits>) == 3628800 (0x<hex digits>)

-------------------------------------------------------------------------------

Generator over a range of pairs

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( i->first == i->second-1 )

with expansion:

0 == 0

-------------------------------------------------------------------------------

Generator over a range of pairs

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( i->first == i->second-1 )

with expansion:

2 == 2

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

2 == 2

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

200 == 200

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

4 == 4

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

200 == 200

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

6 == 6

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

200 == 200

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

8 == 8

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

200 == 200

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

10 == 10

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

200 == 200

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

30 == 30

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

200 == 200

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

40 == 40

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

200 == 200

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

42 == 42

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

200 == 200

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

72 == 72

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

200 == 200

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

2 == 2

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

202 == 202

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

4 == 4

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

202 == 202

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

6 == 6

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

202 == 202

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

8 == 8

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

202 == 202

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

10 == 10

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

202 == 202

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

30 == 30

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

202 == 202

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

40 == 40

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

202 == 202

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

42 == 42

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

202 == 202

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

72 == 72

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

202 == 202

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

2 == 2

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

204 == 204

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

4 == 4

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

204 == 204

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

6 == 6

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

204 == 204

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

8 == 8

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

204 == 204

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

10 == 10

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

204 == 204

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

30 == 30

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

204 == 204

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

40 == 40

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

204 == 204

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

42 == 42

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

204 == 204

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

72 == 72

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

204 == 204

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

2 == 2

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

206 == 206

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

4 == 4

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

206 == 206

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

6 == 6

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

206 == 206

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

8 == 8

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

206 == 206

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

10 == 10

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

206 == 206

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

30 == 30

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

206 == 206

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

40 == 40

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

206 == 206

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

42 == 42

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

206 == 206

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

72 == 72

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

206 == 206

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

2 == 2

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

208 == 208

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

4 == 4

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

208 == 208

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

6 == 6

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

208 == 208

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

8 == 8

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

208 == 208

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

10 == 10

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

208 == 208

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

30 == 30

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

208 == 208

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

40 == 40

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

208 == 208

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

42 == 42

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

208 == 208

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

72 == 72

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

208 == 208

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

2 == 2

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

210 == 210

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

4 == 4

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

210 == 210

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

6 == 6

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

210 == 210

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

8 == 8

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

210 == 210

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

10 == 10

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

210 == 210

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

30 == 30

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

210 == 210

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

40 == 40

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

210 == 210

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

42 == 42

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

210 == 210

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

72 == 72

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

210 == 210

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

2 == 2

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

212 == 212

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

4 == 4

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

212 == 212

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

6 == 6

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

212 == 212

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

8 == 8

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

212 == 212

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

10 == 10

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

212 == 212

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

30 == 30

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

212 == 212

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

40 == 40

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

212 == 212

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

42 == 42

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

212 == 212

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

72 == 72

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

212 == 212

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

2 == 2

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

214 == 214

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

4 == 4

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

214 == 214

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

6 == 6

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

214 == 214

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

8 == 8

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

214 == 214

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

10 == 10

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

214 == 214

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

30 == 30

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

214 == 214

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

40 == 40

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

214 == 214

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

42 == 42

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

214 == 214

-------------------------------------------------------------------------------

Generators over two ranges

-------------------------------------------------------------------------------

GeneratorTests.cpp:<line number>

...............................................................................

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( i, 2 ) == i\*2 )

with expansion:

72 == 72

GeneratorTests.cpp:<line number>:

PASSED:

CATCH\_REQUIRE( multiply( j, 2 ) == j\*2 )

with expansion:

214 == 214

-------------------------------------------------------------------------------

Greater-than inequalities with different epsilons

-------------------------------------------------------------------------------

ApproxTests.cpp:<line number>

...............................................................................

ApproxTests.cpp:<line number>:

PASSED:

REQUIRE( d >= Approx( 1.22 ) )

with expansion:

1.23 >= Approx( 1.22 )

ApproxTests.cpp:<line number>:

PASSED:

REQUIRE( d >= Approx( 1.23 ) )

with expansion:

1.23 >= Approx( 1.23 )

ApproxTests.cpp:<line number>:

PASSED:

REQUIRE\_FALSE( d >= Approx( 1.24 ) )

with expansion:

!(1.23 >= Approx( 1.24 ))

ApproxTests.cpp:<line number>:

PASSED:

REQUIRE( d >= Approx( 1.24 ).epsilon(0.1) )

with expansion:

1.23 >= Approx( 1.24 )

-------------------------------------------------------------------------------

INFO and WARN do not abort tests

-------------------------------------------------------------------------------

MessageTests.cpp:<line number>

...............................................................................

MessageTests.cpp:<line number>:

warning:

this is a message

this is a warning

-------------------------------------------------------------------------------

INFO gets logged on failure

-------------------------------------------------------------------------------

MessageTests.cpp:<line number>

...............................................................................

MessageTests.cpp:<line number>: FAILED:

REQUIRE( a == 1 )

with expansion:

2 == 1

with messages:

this message should be logged

so should this

-------------------------------------------------------------------------------

INFO gets logged on failure, even if captured before successful assertions

-------------------------------------------------------------------------------

MessageTests.cpp:<line number>

...............................................................................

MessageTests.cpp:<line number>:

PASSED:

CHECK( a == 2 )

with expansion:

2 == 2

with message:

this message may be logged later

MessageTests.cpp:<line number>: FAILED:

CHECK( a == 1 )

with expansion:

2 == 1

with messages:

this message may be logged later

this message should be logged

MessageTests.cpp:<line number>: FAILED:

CHECK( a == 0 )

with expansion:

2 == 0

with messages:

this message may be logged later

this message should be logged

and this, but later

MessageTests.cpp:<line number>:

PASSED:

CHECK( a == 2 )

with expansion:

2 == 2

with messages:

this message may be logged later

this message should be logged

and this, but later

but not this

-------------------------------------------------------------------------------

Inequality checks that should fail

-------------------------------------------------------------------------------

ConditionTests.cpp:<line number>

...............................................................................

ConditionTests.cpp:<line number>: FAILED:

CHECK( data.int\_seven != 7 )

with expansion:

7 != 7

ConditionTests.cpp:<line number>: FAILED:

CHECK( data.float\_nine\_point\_one != Approx( 9.1f ) )

with expansion:

9.1f != Approx( 9.1000003815 )

ConditionTests.cpp:<line number>: FAILED:

CHECK( data.double\_pi != Approx( 3.1415926535 ) )

with expansion:

3.1415926535 != Approx( 3.1415926535 )

ConditionTests.cpp:<line number>: FAILED:

CHECK( data.str\_hello != "hello" )

with expansion:

"hello" != "hello"

ConditionTests.cpp:<line number>: FAILED:

CHECK( data.str\_hello.size() != 5 )

with expansion:

5 != 5

-------------------------------------------------------------------------------

Inequality checks that should succeed

-------------------------------------------------------------------------------

ConditionTests.cpp:<line number>

...............................................................................

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( data.int\_seven != 6 )

with expansion:

7 != 6

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( data.int\_seven != 8 )

with expansion:

7 != 8

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( data.float\_nine\_point\_one != Approx( 9.11f ) )

with expansion:

9.1f != Approx( 9.1099996567 )

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( data.float\_nine\_point\_one != Approx( 9.0f ) )

with expansion:

9.1f != Approx( 9.0 )

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( data.float\_nine\_point\_one != Approx( 1 ) )

with expansion:

9.1f != Approx( 1.0 )

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( data.float\_nine\_point\_one != Approx( 0 ) )

with expansion:

9.1f != Approx( 0.0 )

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( data.double\_pi != Approx( 3.1415 ) )

with expansion:

3.1415926535 != Approx( 3.1415 )

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( data.str\_hello != "goodbye" )

with expansion:

"hello" != "goodbye"

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( data.str\_hello != "hell" )

with expansion:

"hello" != "hell"

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( data.str\_hello != "hello1" )

with expansion:

"hello" != "hello1"

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( data.str\_hello.size() != 6 )

with expansion:

5 != 6

-------------------------------------------------------------------------------

Less-than inequalities with different epsilons

-------------------------------------------------------------------------------

ApproxTests.cpp:<line number>

...............................................................................

ApproxTests.cpp:<line number>:

PASSED:

REQUIRE( d <= Approx( 1.24 ) )

with expansion:

1.23 <= Approx( 1.24 )

ApproxTests.cpp:<line number>:

PASSED:

REQUIRE( d <= Approx( 1.23 ) )

with expansion:

1.23 <= Approx( 1.23 )

ApproxTests.cpp:<line number>:

PASSED:

REQUIRE\_FALSE( d <= Approx( 1.22 ) )

with expansion:

!(1.23 <= Approx( 1.22 ))

ApproxTests.cpp:<line number>:

PASSED:

REQUIRE( d <= Approx( 1.22 ).epsilon(0.1) )

with expansion:

1.23 <= Approx( 1.22 )

-------------------------------------------------------------------------------

Long strings can be wrapped

plain string

No wrapping

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK( Text( testString, TextAttributes().setWidth( 80 ) ).toString() == testString )

with expansion:

"one two three four"

==

"one two three four"

TestMain.cpp:<line number>:

PASSED:

CHECK( Text( testString, TextAttributes().setWidth( 18 ) ).toString() == testString )

with expansion:

"one two three four"

==

"one two three four"

-------------------------------------------------------------------------------

Long strings can be wrapped

plain string

Wrapped once

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK( Text( testString, TextAttributes().setWidth( 17 ) ).toString() == "one two three\nfour" )

with expansion:

"one two three

four"

==

"one two three

four"

TestMain.cpp:<line number>:

PASSED:

CHECK( Text( testString, TextAttributes().setWidth( 16 ) ).toString() == "one two three\nfour" )

with expansion:

"one two three

four"

==

"one two three

four"

TestMain.cpp:<line number>:

PASSED:

CHECK( Text( testString, TextAttributes().setWidth( 14 ) ).toString() == "one two three\nfour" )

with expansion:

"one two three

four"

==

"one two three

four"

TestMain.cpp:<line number>:

PASSED:

CHECK( Text( testString, TextAttributes().setWidth( 13 ) ).toString() == "one two three\nfour" )

with expansion:

"one two three

four"

==

"one two three

four"

TestMain.cpp:<line number>:

PASSED:

CHECK( Text( testString, TextAttributes().setWidth( 12 ) ).toString() == "one two\nthree four" )

with expansion:

"one two

three four"

==

"one two

three four"

-------------------------------------------------------------------------------

Long strings can be wrapped

plain string

Wrapped twice

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK( Text( testString, TextAttributes().setWidth( 9 ) ).toString() == "one two\nthree\nfour" )

with expansion:

"one two

three

four"

==

"one two

three

four"

TestMain.cpp:<line number>:

PASSED:

CHECK( Text( testString, TextAttributes().setWidth( 8 ) ).toString() == "one two\nthree\nfour" )

with expansion:

"one two

three

four"

==

"one two

three

four"

TestMain.cpp:<line number>:

PASSED:

CHECK( Text( testString, TextAttributes().setWidth( 7 ) ).toString() == "one two\nthree\nfour" )

with expansion:

"one two

three

four"

==

"one two

three

four"

-------------------------------------------------------------------------------

Long strings can be wrapped

plain string

Wrapped three times

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK( Text( testString, TextAttributes().setWidth( 6 ) ).toString() == "one\ntwo\nthree\nfour" )

with expansion:

"one

two

three

four"

==

"one

two

three

four"

TestMain.cpp:<line number>:

PASSED:

CHECK( Text( testString, TextAttributes().setWidth( 5 ) ).toString() == "one\ntwo\nthree\nfour" )

with expansion:

"one

two

three

four"

==

"one

two

three

four"

-------------------------------------------------------------------------------

Long strings can be wrapped

plain string

Short wrap

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK( Text( "abcdef", TextAttributes().setWidth( 4 ) ).toString() == "abc-\ndef" )

with expansion:

"abc-

def"

==

"abc-

def"

TestMain.cpp:<line number>:

PASSED:

CHECK( Text( "abcdefg", TextAttributes().setWidth( 4 ) ).toString() == "abc-\ndefg" )

with expansion:

"abc-

defg"

==

"abc-

defg"

TestMain.cpp:<line number>:

PASSED:

CHECK( Text( "abcdefgh", TextAttributes().setWidth( 4 ) ).toString() == "abc-\ndef-\ngh" )

with expansion:

"abc-

def-

gh"

==

"abc-

def-

gh"

TestMain.cpp:<line number>:

PASSED:

CHECK( Text( testString, TextAttributes().setWidth( 4 ) ).toString() == "one\ntwo\nthr-\nee\nfour" )

with expansion:

"one

two

thr-

ee

four"

==

"one

two

thr-

ee

four"

TestMain.cpp:<line number>:

PASSED:

CHECK( Text( testString, TextAttributes().setWidth( 3 ) ).toString() == "one\ntwo\nth-\nree\nfo-\nur" )

with expansion:

"one

two

th-

ree

fo-

ur"

==

"one

two

th-

ree

fo-

ur"

-------------------------------------------------------------------------------

Long strings can be wrapped

plain string

As container

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

REQUIRE( text.size() == 4 )

with expansion:

4 == 4

TestMain.cpp:<line number>:

PASSED:

CHECK( text[0] == "one" )

with expansion:

"one" == "one"

TestMain.cpp:<line number>:

PASSED:

CHECK( text[1] == "two" )

with expansion:

"two" == "two"

TestMain.cpp:<line number>:

PASSED:

CHECK( text[2] == "three" )

with expansion:

"three" == "three"

TestMain.cpp:<line number>:

PASSED:

CHECK( text[3] == "four" )

with expansion:

"four" == "four"

-------------------------------------------------------------------------------

Long strings can be wrapped

plain string

Indent first line differently

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK( text.toString() == " one two\n three\n four" )

with expansion:

" one two

three

four"

==

" one two

three

four"

-------------------------------------------------------------------------------

Long strings can be wrapped

With newlines

No wrapping

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK( Text( testString, TextAttributes().setWidth( 80 ) ).toString() == testString )

with expansion:

"one two

three four"

==

"one two

three four"

TestMain.cpp:<line number>:

PASSED:

CHECK( Text( testString, TextAttributes().setWidth( 18 ) ).toString() == testString )

with expansion:

"one two

three four"

==

"one two

three four"

TestMain.cpp:<line number>:

PASSED:

CHECK( Text( testString, TextAttributes().setWidth( 10 ) ).toString() == testString )

with expansion:

"one two

three four"

==

"one two

three four"

-------------------------------------------------------------------------------

Long strings can be wrapped

With newlines

Trailing newline

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK( Text( "abcdef\n", TextAttributes().setWidth( 10 ) ).toString() == "abcdef" )

with expansion:

"abcdef" == "abcdef"

TestMain.cpp:<line number>:

PASSED:

CHECK( Text( "abcdef", TextAttributes().setWidth( 6 ) ).toString() == "abcdef" )

with expansion:

"abcdef" == "abcdef"

TestMain.cpp:<line number>:

PASSED:

CHECK( Text( "abcdef\n", TextAttributes().setWidth( 6 ) ).toString() == "abcdef" )

with expansion:

"abcdef" == "abcdef"

TestMain.cpp:<line number>:

PASSED:

CHECK( Text( "abcdef\n", TextAttributes().setWidth( 5 ) ).toString() == "abcd-\nef" )

with expansion:

"abcd-

ef"

==

"abcd-

ef"

-------------------------------------------------------------------------------

Long strings can be wrapped

With newlines

Wrapped once

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK( Text( testString, TextAttributes().setWidth( 9 ) ).toString() == "one two\nthree\nfour" )

with expansion:

"one two

three

four"

==

"one two

three

four"

TestMain.cpp:<line number>:

PASSED:

CHECK( Text( testString, TextAttributes().setWidth( 8 ) ).toString() == "one two\nthree\nfour" )

with expansion:

"one two

three

four"

==

"one two

three

four"

TestMain.cpp:<line number>:

PASSED:

CHECK( Text( testString, TextAttributes().setWidth( 7 ) ).toString() == "one two\nthree\nfour" )

with expansion:

"one two

three

four"

==

"one two

three

four"

-------------------------------------------------------------------------------

Long strings can be wrapped

With newlines

Wrapped twice

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK( Text( testString, TextAttributes().setWidth( 6 ) ).toString() == "one\ntwo\nthree\nfour" )

with expansion:

"one

two

three

four"

==

"one

two

three

four"

-------------------------------------------------------------------------------

Long strings can be wrapped

With wrap-before/ after characters

No wrapping

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK( Text( testString, TextAttributes().setWidth( 80 ) ).toString() == testString )

with expansion:

"one,two(three) <here>"

==

"one,two(three) <here>"

TestMain.cpp:<line number>:

PASSED:

CHECK( Text( testString, TextAttributes().setWidth( 24 ) ).toString() == testString )

with expansion:

"one,two(three) <here>"

==

"one,two(three) <here>"

-------------------------------------------------------------------------------

Long strings can be wrapped

With wrap-before/ after characters

Wrap before

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK( Text( testString, TextAttributes().setWidth( 11 ) ).toString() == "one,two\n(three)\n<here>" )

with expansion:

"one,two

(three)

<here>"

==

"one,two

(three)

<here>"

-------------------------------------------------------------------------------

Long strings can be wrapped

With wrap-before/ after characters

Wrap after

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK( Text( testString, TextAttributes().setWidth( 6 ) ).toString() == "one,\ntwo\n(thre-\ne)\n<here>" )

with expansion:

"one,

two

(thre-

e)

<here>"

==

"one,

two

(thre-

e)

<here>"

TestMain.cpp:<line number>:

PASSED:

CHECK( Text( testString, TextAttributes().setWidth( 5 ) ).toString() == "one,\ntwo\n(thr-\nee)\n<her-\ne>" )

with expansion:

"one,

two

(thr-

ee)

<her-

e>"

==

"one,

two

(thr-

ee)

<her-

e>"

TestMain.cpp:<line number>:

PASSED:

CHECK( Text( testString, TextAttributes().setWidth( 4 ) ).toString() == "one,\ntwo\n(th-\nree)\n<he-\nre>" )

with expansion:

"one,

two

(th-

ree)

<he-

re>"

==

"one,

two

(th-

ree)

<he-

re>"

-------------------------------------------------------------------------------

Long text is truncated

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK\_THAT( t.toString(), EndsWith( "... message truncated due to excessive size" ) )

with expansion:

"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-

\*\*-

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

... message truncated due to excessive size

-------------------------------------------------------------------------------

ManuallyRegistered

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

with message:

was called

-------------------------------------------------------------------------------

Matchers can be (AllOf) composed with the && operator

-------------------------------------------------------------------------------

MatchersTests.cpp:<line number>

...............................................................................

MatchersTests.cpp:<line number>:

PASSED:

CHECK\_THAT( testStringForMatching(), Contains( "string" ) && Contains( "abc" ) && Contains( "substring" ) && Contains( "contains" ) )

with expansion:

"this string contains 'abc' as a substring" ( contains: "string" and

contains: "abc" and contains: "substring" and contains: "contains" )

-------------------------------------------------------------------------------

Matchers can be (AnyOf) composed with the || operator

-------------------------------------------------------------------------------

MatchersTests.cpp:<line number>

...............................................................................

MatchersTests.cpp:<line number>:

PASSED:

CHECK\_THAT( testStringForMatching(), Contains( "string" ) || Contains( "different" ) || Contains( "random" ) )

with expansion:

"this string contains 'abc' as a substring" ( contains: "string" or contains:

"different" or contains: "random" )

MatchersTests.cpp:<line number>:

PASSED:

CHECK\_THAT( testStringForMatching2(), Contains( "string" ) || Contains( "different" ) || Contains( "random" ) )

with expansion:

"some completely different text that contains one common word" ( contains:

"string" or contains: "different" or contains: "random" )

-------------------------------------------------------------------------------

Matchers can be composed with both && and ||

-------------------------------------------------------------------------------

MatchersTests.cpp:<line number>

...............................................................................

MatchersTests.cpp:<line number>:

PASSED:

CHECK\_THAT( testStringForMatching(), ( Contains( "string" ) || Contains( "different" ) ) && Contains( "substring" ) )

with expansion:

"this string contains 'abc' as a substring" ( ( contains: "string" or

contains: "different" ) and contains: "substring" )

-------------------------------------------------------------------------------

Matchers can be composed with both && and || - failing

-------------------------------------------------------------------------------

MatchersTests.cpp:<line number>

...............................................................................

MatchersTests.cpp:<line number>: FAILED:

CHECK\_THAT( testStringForMatching(), ( Contains( "string" ) || Contains( "different" ) ) && Contains( "random" ) )

with expansion:

"this string contains 'abc' as a substring" ( ( contains: "string" or

contains: "different" ) and contains: "random" )

-------------------------------------------------------------------------------

Matchers can be negated (Not) with the ! operator

-------------------------------------------------------------------------------

MatchersTests.cpp:<line number>

...............................................................................

MatchersTests.cpp:<line number>:

PASSED:

CHECK\_THAT( testStringForMatching(), !Contains( "different" ) )

with expansion:

"this string contains 'abc' as a substring" not contains: "different"

-------------------------------------------------------------------------------

Matchers can be negated (Not) with the ! operator - failing

-------------------------------------------------------------------------------

MatchersTests.cpp:<line number>

...............................................................................

MatchersTests.cpp:<line number>: FAILED:

CHECK\_THAT( testStringForMatching(), !Contains( "substring" ) )

with expansion:

"this string contains 'abc' as a substring" not contains: "substring"

-------------------------------------------------------------------------------

Mismatching exception messages failing the test

-------------------------------------------------------------------------------

ExceptionTests.cpp:<line number>

...............................................................................

ExceptionTests.cpp:<line number>:

PASSED:

REQUIRE\_THROWS\_WITH( thisThrows(), "expected exception" )

ExceptionTests.cpp:<line number>: FAILED:

REQUIRE\_THROWS\_WITH( thisThrows(), "should fail" )

with expansion:

expected exception

-------------------------------------------------------------------------------

Nice descriptive name

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

warning:

This one ran

-------------------------------------------------------------------------------

Non-std exceptions can be translated

-------------------------------------------------------------------------------

ExceptionTests.cpp:<line number>

...............................................................................

ExceptionTests.cpp:<line number>: FAILED:

due to unexpected exception with message:

custom exception

-------------------------------------------------------------------------------

NotImplemented exception

-------------------------------------------------------------------------------

ExceptionTests.cpp:<line number>

...............................................................................

ExceptionTests.cpp:<line number>:

PASSED:

REQUIRE\_THROWS( thisFunctionNotImplemented( 7 ) )

-------------------------------------------------------------------------------

Objects that evaluated in boolean contexts can be checked

-------------------------------------------------------------------------------

TrickyTests.cpp:<line number>

...............................................................................

TrickyTests.cpp:<line number>:

PASSED:

CHECK( True )

with expansion:

1

TrickyTests.cpp:<line number>:

PASSED:

CHECK( !False )

with expansion:

true

TrickyTests.cpp:<line number>:

PASSED:

CHECK\_FALSE( False )

with expansion:

!0

-------------------------------------------------------------------------------

Operators at different namespace levels not hijacked by Koenig lookup

-------------------------------------------------------------------------------

TrickyTests.cpp:<line number>

...............................................................................

TrickyTests.cpp:<line number>:

PASSED:

REQUIRE( 0x<hex digits> == o )

with expansion:

3221225472 (0x<hex digits>) == {?}

-------------------------------------------------------------------------------

Ordering comparison checks that should fail

-------------------------------------------------------------------------------

ConditionTests.cpp:<line number>

...............................................................................

ConditionTests.cpp:<line number>: FAILED:

CHECK( data.int\_seven > 7 )

with expansion:

7 > 7

ConditionTests.cpp:<line number>: FAILED:

CHECK( data.int\_seven < 7 )

with expansion:

7 < 7

ConditionTests.cpp:<line number>: FAILED:

CHECK( data.int\_seven > 8 )

with expansion:

7 > 8

ConditionTests.cpp:<line number>: FAILED:

CHECK( data.int\_seven < 6 )

with expansion:

7 < 6

ConditionTests.cpp:<line number>: FAILED:

CHECK( data.int\_seven < 0 )

with expansion:

7 < 0

ConditionTests.cpp:<line number>: FAILED:

CHECK( data.int\_seven < -1 )

with expansion:

7 < -1

ConditionTests.cpp:<line number>: FAILED:

CHECK( data.int\_seven >= 8 )

with expansion:

7 >= 8

ConditionTests.cpp:<line number>: FAILED:

CHECK( data.int\_seven <= 6 )

with expansion:

7 <= 6

ConditionTests.cpp:<line number>: FAILED:

CHECK( data.float\_nine\_point\_one < 9 )

with expansion:

9.1f < 9

ConditionTests.cpp:<line number>: FAILED:

CHECK( data.float\_nine\_point\_one > 10 )

with expansion:

9.1f > 10

ConditionTests.cpp:<line number>: FAILED:

CHECK( data.float\_nine\_point\_one > 9.2 )

with expansion:

9.1f > 9.2

ConditionTests.cpp:<line number>: FAILED:

CHECK( data.str\_hello > "hello" )

with expansion:

"hello" > "hello"

ConditionTests.cpp:<line number>: FAILED:

CHECK( data.str\_hello < "hello" )

with expansion:

"hello" < "hello"

ConditionTests.cpp:<line number>: FAILED:

CHECK( data.str\_hello > "hellp" )

with expansion:

"hello" > "hellp"

ConditionTests.cpp:<line number>: FAILED:

CHECK( data.str\_hello > "z" )

with expansion:

"hello" > "z"

ConditionTests.cpp:<line number>: FAILED:

CHECK( data.str\_hello < "hellm" )

with expansion:

"hello" < "hellm"

ConditionTests.cpp:<line number>: FAILED:

CHECK( data.str\_hello < "a" )

with expansion:

"hello" < "a"

ConditionTests.cpp:<line number>: FAILED:

CHECK( data.str\_hello >= "z" )

with expansion:

"hello" >= "z"

ConditionTests.cpp:<line number>: FAILED:

CHECK( data.str\_hello <= "a" )

with expansion:

"hello" <= "a"

-------------------------------------------------------------------------------

Ordering comparison checks that should succeed

-------------------------------------------------------------------------------

ConditionTests.cpp:<line number>

...............................................................................

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( data.int\_seven < 8 )

with expansion:

7 < 8

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( data.int\_seven > 6 )

with expansion:

7 > 6

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( data.int\_seven > 0 )

with expansion:

7 > 0

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( data.int\_seven > -1 )

with expansion:

7 > -1

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( data.int\_seven >= 7 )

with expansion:

7 >= 7

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( data.int\_seven >= 6 )

with expansion:

7 >= 6

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( data.int\_seven <= 7 )

with expansion:

7 <= 7

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( data.int\_seven <= 8 )

with expansion:

7 <= 8

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( data.float\_nine\_point\_one > 9 )

with expansion:

9.1f > 9

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( data.float\_nine\_point\_one < 10 )

with expansion:

9.1f < 10

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( data.float\_nine\_point\_one < 9.2 )

with expansion:

9.1f < 9.2

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( data.str\_hello <= "hello" )

with expansion:

"hello" <= "hello"

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( data.str\_hello >= "hello" )

with expansion:

"hello" >= "hello"

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( data.str\_hello < "hellp" )

with expansion:

"hello" < "hellp"

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( data.str\_hello < "zebra" )

with expansion:

"hello" < "zebra"

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( data.str\_hello > "hellm" )

with expansion:

"hello" > "hellm"

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( data.str\_hello > "a" )

with expansion:

"hello" > "a"

-------------------------------------------------------------------------------

Output from all sections is reported

one

-------------------------------------------------------------------------------

MessageTests.cpp:<line number>

...............................................................................

MessageTests.cpp:<line number>: FAILED:

explicitly with message:

Message from section one

-------------------------------------------------------------------------------

Output from all sections is reported

two

-------------------------------------------------------------------------------

MessageTests.cpp:<line number>

...............................................................................

MessageTests.cpp:<line number>: FAILED:

explicitly with message:

Message from section two

-------------------------------------------------------------------------------

Parse test names and tags

Empty test spec should have no filters

-------------------------------------------------------------------------------

CmdLineTests.cpp:<line number>

...............................................................................

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.hasFilters() == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcA ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcB ) == false )

with expansion:

false == false

-------------------------------------------------------------------------------

Parse test names and tags

Test spec from empty string should have no filters

-------------------------------------------------------------------------------

CmdLineTests.cpp:<line number>

...............................................................................

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.hasFilters() == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches(tcA ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcB ) == false )

with expansion:

false == false

-------------------------------------------------------------------------------

Parse test names and tags

Test spec from just a comma should have no filters

-------------------------------------------------------------------------------

CmdLineTests.cpp:<line number>

...............................................................................

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.hasFilters() == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcA ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcB ) == false )

with expansion:

false == false

-------------------------------------------------------------------------------

Parse test names and tags

Test spec from name should have one filter

-------------------------------------------------------------------------------

CmdLineTests.cpp:<line number>

...............................................................................

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.hasFilters() == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcA ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcB ) == true )

with expansion:

true == true

-------------------------------------------------------------------------------

Parse test names and tags

Test spec from quoted name should have one filter

-------------------------------------------------------------------------------

CmdLineTests.cpp:<line number>

...............................................................................

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.hasFilters() == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcA ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcB ) == true )

with expansion:

true == true

-------------------------------------------------------------------------------

Parse test names and tags

Test spec from name should have one filter

-------------------------------------------------------------------------------

CmdLineTests.cpp:<line number>

...............................................................................

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.hasFilters() == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcA ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcB ) == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcC ) == false )

with expansion:

false == false

-------------------------------------------------------------------------------

Parse test names and tags

Wildcard at the start

-------------------------------------------------------------------------------

CmdLineTests.cpp:<line number>

...............................................................................

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.hasFilters() == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcA ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcB ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcC ) == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcD ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( parseTestSpec( "\*a" ).matches( tcA ) == true )

with expansion:

true == true

-------------------------------------------------------------------------------

Parse test names and tags

Wildcard at the end

-------------------------------------------------------------------------------

CmdLineTests.cpp:<line number>

...............................................................................

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.hasFilters() == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcA ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcB ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcC ) == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcD ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( parseTestSpec( "a\*" ).matches( tcA ) == true )

with expansion:

true == true

-------------------------------------------------------------------------------

Parse test names and tags

Wildcard at both ends

-------------------------------------------------------------------------------

CmdLineTests.cpp:<line number>

...............................................................................

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.hasFilters() == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcA ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcB ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcC ) == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcD ) == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( parseTestSpec( "\*a\*" ).matches( tcA ) == true )

with expansion:

true == true

-------------------------------------------------------------------------------

Parse test names and tags

Redundant wildcard at the start

-------------------------------------------------------------------------------

CmdLineTests.cpp:<line number>

...............................................................................

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.hasFilters() == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcA ) == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcB ) == false )

with expansion:

false == false

-------------------------------------------------------------------------------

Parse test names and tags

Redundant wildcard at the end

-------------------------------------------------------------------------------

CmdLineTests.cpp:<line number>

...............................................................................

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.hasFilters() == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcA ) == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcB ) == false )

with expansion:

false == false

-------------------------------------------------------------------------------

Parse test names and tags

Redundant wildcard at both ends

-------------------------------------------------------------------------------

CmdLineTests.cpp:<line number>

...............................................................................

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.hasFilters() == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcA ) == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcB ) == false )

with expansion:

false == false

-------------------------------------------------------------------------------

Parse test names and tags

Wildcard at both ends, redundant at start

-------------------------------------------------------------------------------

CmdLineTests.cpp:<line number>

...............................................................................

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.hasFilters() == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcA ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcB ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcC ) == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcD ) == true )

with expansion:

true == true

-------------------------------------------------------------------------------

Parse test names and tags

Just wildcard

-------------------------------------------------------------------------------

CmdLineTests.cpp:<line number>

...............................................................................

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.hasFilters() == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcA ) == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcB ) == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcC ) == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcD ) == true )

with expansion:

true == true

-------------------------------------------------------------------------------

Parse test names and tags

Single tag

-------------------------------------------------------------------------------

CmdLineTests.cpp:<line number>

...............................................................................

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.hasFilters() == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcA ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcB ) == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcC ) == false )

with expansion:

false == false

-------------------------------------------------------------------------------

Parse test names and tags

Single tag, two matches

-------------------------------------------------------------------------------

CmdLineTests.cpp:<line number>

...............................................................................

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.hasFilters() == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcA ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcB ) == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcC ) == true )

with expansion:

true == true

-------------------------------------------------------------------------------

Parse test names and tags

Two tags

-------------------------------------------------------------------------------

CmdLineTests.cpp:<line number>

...............................................................................

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.hasFilters() == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcA ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcB ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcC ) == true )

with expansion:

true == true

-------------------------------------------------------------------------------

Parse test names and tags

Two tags, spare separated

-------------------------------------------------------------------------------

CmdLineTests.cpp:<line number>

...............................................................................

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.hasFilters() == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcA ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcB ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcC ) == true )

with expansion:

true == true

-------------------------------------------------------------------------------

Parse test names and tags

Wildcarded name and tag

-------------------------------------------------------------------------------

CmdLineTests.cpp:<line number>

...............................................................................

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.hasFilters() == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcA ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcB ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcC ) == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcD ) == false )

with expansion:

false == false

-------------------------------------------------------------------------------

Parse test names and tags

Single tag exclusion

-------------------------------------------------------------------------------

CmdLineTests.cpp:<line number>

...............................................................................

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.hasFilters() == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcA ) == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcB ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcC ) == true )

with expansion:

true == true

-------------------------------------------------------------------------------

Parse test names and tags

One tag exclusion and one tag inclusion

-------------------------------------------------------------------------------

CmdLineTests.cpp:<line number>

...............................................................................

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.hasFilters() == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcA ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcB ) == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcC ) == false )

with expansion:

false == false

-------------------------------------------------------------------------------

Parse test names and tags

One tag exclusion and one wldcarded name inclusion

-------------------------------------------------------------------------------

CmdLineTests.cpp:<line number>

...............................................................................

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.hasFilters() == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcA ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcB ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcC ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcD ) == true )

with expansion:

true == true

-------------------------------------------------------------------------------

Parse test names and tags

One tag exclusion, using exclude:, and one wldcarded name inclusion

-------------------------------------------------------------------------------

CmdLineTests.cpp:<line number>

...............................................................................

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.hasFilters() == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcA ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcB ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcC ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcD ) == true )

with expansion:

true == true

-------------------------------------------------------------------------------

Parse test names and tags

name exclusion

-------------------------------------------------------------------------------

CmdLineTests.cpp:<line number>

...............................................................................

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.hasFilters() == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcA ) == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcB ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcC ) == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcD ) == true )

with expansion:

true == true

-------------------------------------------------------------------------------

Parse test names and tags

wildcarded name exclusion

-------------------------------------------------------------------------------

CmdLineTests.cpp:<line number>

...............................................................................

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.hasFilters() == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcA ) == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcB ) == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcC ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcD ) == false )

with expansion:

false == false

-------------------------------------------------------------------------------

Parse test names and tags

wildcarded name exclusion with tag inclusion

-------------------------------------------------------------------------------

CmdLineTests.cpp:<line number>

...............................................................................

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.hasFilters() == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcA ) == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcB ) == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcC ) == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcD ) == false )

with expansion:

false == false

-------------------------------------------------------------------------------

Parse test names and tags

wildcarded name exclusion, using exclude:, with tag inclusion

-------------------------------------------------------------------------------

CmdLineTests.cpp:<line number>

...............................................................................

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.hasFilters() == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcA ) == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcB ) == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcC ) == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcD ) == false )

with expansion:

false == false

-------------------------------------------------------------------------------

Parse test names and tags

two wildcarded names

-------------------------------------------------------------------------------

CmdLineTests.cpp:<line number>

...............................................................................

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.hasFilters() == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcA ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcB ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcC ) == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcD ) == false )

with expansion:

false == false

-------------------------------------------------------------------------------

Parse test names and tags

empty tag

-------------------------------------------------------------------------------

CmdLineTests.cpp:<line number>

...............................................................................

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.hasFilters() == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcA ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcB ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcC ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcD ) == false )

with expansion:

false == false

-------------------------------------------------------------------------------

Parse test names and tags

empty quoted name

-------------------------------------------------------------------------------

CmdLineTests.cpp:<line number>

...............................................................................

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.hasFilters() == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcA ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcB ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcC ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcD ) == false )

with expansion:

false == false

-------------------------------------------------------------------------------

Parse test names and tags

quoted string followed by tag exclusion

-------------------------------------------------------------------------------

CmdLineTests.cpp:<line number>

...............................................................................

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.hasFilters() == true )

with expansion:

true == true

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcA ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcB ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcC ) == false )

with expansion:

false == false

CmdLineTests.cpp:<line number>:

PASSED:

CHECK( spec.matches( tcD ) == true )

with expansion:

true == true

-------------------------------------------------------------------------------

Parsing a std::pair

-------------------------------------------------------------------------------

TrickyTests.cpp:<line number>

...............................................................................

TrickyTests.cpp:<line number>:

PASSED:

REQUIRE( (std::pair<int, int>( 1, 2 )) == aNicePair )

with expansion:

std::pair( 1, 2 ) == std::pair( 1, 2 )

-------------------------------------------------------------------------------

Pointers can be compared to null

-------------------------------------------------------------------------------

ConditionTests.cpp:<line number>

...............................................................................

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( p == 0 )

with expansion:

NULL == 0

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( p == pNULL )

with expansion:

NULL == NULL

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( p != 0 )

with expansion:

0x<hex digits> != 0

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( cp != 0 )

with expansion:

0x<hex digits> != 0

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( cpc != 0 )

with expansion:

0x<hex digits> != 0

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( returnsNull() == 0 )

with expansion:

{null string} == 0

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( returnsConstNull() == 0 )

with expansion:

{null string} == 0

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( 0 != p )

with expansion:

0 != 0x<hex digits>

-------------------------------------------------------------------------------

Pointers can be converted to strings

-------------------------------------------------------------------------------

MessageTests.cpp:<line number>

...............................................................................

MessageTests.cpp:<line number>:

warning:

actual address of p: 0x<hex digits>

MessageTests.cpp:<line number>:

warning:

toString(p): 0x<hex digits>

-------------------------------------------------------------------------------

Process can be configured on command line

empty args don't cause a crash

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK\_NOTHROW( parser.parseInto( std::vector<std::string>(), config ) )

TestMain.cpp:<line number>:

PASSED:

CHECK( config.processName == "" )

with expansion:

"" == ""

-------------------------------------------------------------------------------

Process can be configured on command line

default - no arguments

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK\_NOTHROW( parseIntoConfig( argv, config ) )

TestMain.cpp:<line number>:

PASSED:

CHECK( config.processName == "test" )

with expansion:

"test" == "test"

TestMain.cpp:<line number>:

PASSED:

CHECK( config.shouldDebugBreak == false )

with expansion:

false == false

TestMain.cpp:<line number>:

PASSED:

CHECK( config.abortAfter == -1 )

with expansion:

-1 == -1

TestMain.cpp:<line number>:

PASSED:

CHECK( config.noThrow == false )

with expansion:

false == false

TestMain.cpp:<line number>:

PASSED:

CHECK( config.reporterNames.empty() )

with expansion:

true

-------------------------------------------------------------------------------

Process can be configured on command line

test lists

1 test

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK\_NOTHROW( parseIntoConfig( argv, config ) )

TestMain.cpp:<line number>:

PASSED:

REQUIRE( cfg.testSpec().matches( fakeTestCase( "notIncluded" ) ) == false )

with expansion:

false == false

TestMain.cpp:<line number>:

PASSED:

REQUIRE( cfg.testSpec().matches( fakeTestCase( "test1" ) ) )

with expansion:

true

-------------------------------------------------------------------------------

Process can be configured on command line

test lists

Specify one test case exclusion using exclude:

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK\_NOTHROW( parseIntoConfig( argv, config ) )

TestMain.cpp:<line number>:

PASSED:

REQUIRE( cfg.testSpec().matches( fakeTestCase( "test1" ) ) == false )

with expansion:

false == false

TestMain.cpp:<line number>:

PASSED:

REQUIRE( cfg.testSpec().matches( fakeTestCase( "alwaysIncluded" ) ) )

with expansion:

true

-------------------------------------------------------------------------------

Process can be configured on command line

test lists

Specify one test case exclusion using ~

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK\_NOTHROW( parseIntoConfig( argv, config ) )

TestMain.cpp:<line number>:

PASSED:

REQUIRE( cfg.testSpec().matches( fakeTestCase( "test1" ) ) == false )

with expansion:

false == false

TestMain.cpp:<line number>:

PASSED:

REQUIRE( cfg.testSpec().matches( fakeTestCase( "alwaysIncluded" ) ) )

with expansion:

true

-------------------------------------------------------------------------------

Process can be configured on command line

reporter

-r/console

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK\_NOTHROW( parseIntoConfig( argv, config ) )

TestMain.cpp:<line number>:

PASSED:

REQUIRE( config.reporterNames[0] == "console" )

with expansion:

"console" == "console"

-------------------------------------------------------------------------------

Process can be configured on command line

reporter

-r/xml

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK\_NOTHROW( parseIntoConfig( argv, config ) )

TestMain.cpp:<line number>:

PASSED:

REQUIRE( config.reporterNames[0] == "xml" )

with expansion:

"xml" == "xml"

-------------------------------------------------------------------------------

Process can be configured on command line

reporter

-r xml and junit

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK\_NOTHROW( parseIntoConfig( argv, config ) )

TestMain.cpp:<line number>:

PASSED:

REQUIRE( config.reporterNames.size() == 2 )

with expansion:

2 == 2

TestMain.cpp:<line number>:

PASSED:

REQUIRE( config.reporterNames[0] == "xml" )

with expansion:

"xml" == "xml"

TestMain.cpp:<line number>:

PASSED:

REQUIRE( config.reporterNames[1] == "junit" )

with expansion:

"junit" == "junit"

-------------------------------------------------------------------------------

Process can be configured on command line

reporter

--reporter/junit

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK\_NOTHROW( parseIntoConfig( argv, config ) )

TestMain.cpp:<line number>:

PASSED:

REQUIRE( config.reporterNames[0] == "junit" )

with expansion:

"junit" == "junit"

-------------------------------------------------------------------------------

Process can be configured on command line

debugger

-b

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK\_NOTHROW( parseIntoConfig( argv, config ) )

TestMain.cpp:<line number>:

PASSED:

REQUIRE( config.shouldDebugBreak == true )

with expansion:

true == true

-------------------------------------------------------------------------------

Process can be configured on command line

debugger

--break

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK\_NOTHROW( parseIntoConfig( argv, config ) )

TestMain.cpp:<line number>:

PASSED:

REQUIRE( config.shouldDebugBreak )

with expansion:

true

-------------------------------------------------------------------------------

Process can be configured on command line

abort

-a aborts after first failure

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK\_NOTHROW( parseIntoConfig( argv, config ) )

TestMain.cpp:<line number>:

PASSED:

REQUIRE( config.abortAfter == 1 )

with expansion:

1 == 1

-------------------------------------------------------------------------------

Process can be configured on command line

abort

-x 2 aborts after two failures

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK\_NOTHROW( parseIntoConfig( argv, config ) )

TestMain.cpp:<line number>:

PASSED:

REQUIRE( config.abortAfter == 2 )

with expansion:

2 == 2

-------------------------------------------------------------------------------

Process can be configured on command line

abort

-x must be greater than zero

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

REQUIRE\_THAT( parseIntoConfigAndReturnError( argv, config ), Contains( "greater than zero" ) )

with expansion:

"Value after -x or --abortAfter must be greater than zero

- while parsing: (-x, --abortx <no. failures>)" contains: "greater than zero"

-------------------------------------------------------------------------------

Process can be configured on command line

abort

-x must be numeric

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

REQUIRE\_THAT( parseIntoConfigAndReturnError( argv, config ), Contains( "-x" ) )

with expansion:

"Unable to convert oops to destination type

- while parsing: (-x, --abortx <no. failures>)" contains: "-x"

-------------------------------------------------------------------------------

Process can be configured on command line

nothrow

-e

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK\_NOTHROW( parseIntoConfig( argv, config ) )

TestMain.cpp:<line number>:

PASSED:

REQUIRE( config.noThrow == true )

with expansion:

true == true

-------------------------------------------------------------------------------

Process can be configured on command line

nothrow

--nothrow

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK\_NOTHROW( parseIntoConfig( argv, config ) )

TestMain.cpp:<line number>:

PASSED:

REQUIRE( config.noThrow == true )

with expansion:

true == true

-------------------------------------------------------------------------------

Process can be configured on command line

output filename

-o filename

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK\_NOTHROW( parseIntoConfig( argv, config ) )

TestMain.cpp:<line number>:

PASSED:

REQUIRE( config.outputFilename == "filename.ext" )

with expansion:

"filename.ext" == "filename.ext"

-------------------------------------------------------------------------------

Process can be configured on command line

output filename

--out

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK\_NOTHROW( parseIntoConfig( argv, config ) )

TestMain.cpp:<line number>:

PASSED:

REQUIRE( config.outputFilename == "filename.ext" )

with expansion:

"filename.ext" == "filename.ext"

-------------------------------------------------------------------------------

Process can be configured on command line

combinations

Single character flags can be combined

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK\_NOTHROW( parseIntoConfig( argv, config ) )

TestMain.cpp:<line number>:

PASSED:

CHECK( config.abortAfter == 1 )

with expansion:

1 == 1

TestMain.cpp:<line number>:

PASSED:

CHECK( config.shouldDebugBreak )

with expansion:

true

TestMain.cpp:<line number>:

PASSED:

CHECK( config.noThrow == true )

with expansion:

true == true

-------------------------------------------------------------------------------

Process can be configured on command line

use-colour

without option

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK\_NOTHROW( parseIntoConfig( argv, config ) )

TestMain.cpp:<line number>:

PASSED:

REQUIRE( config.useColour == UseColour::Auto )

with expansion:

0 == 0

-------------------------------------------------------------------------------

Process can be configured on command line

use-colour

auto

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK\_NOTHROW( parseIntoConfig( argv, config ) )

TestMain.cpp:<line number>:

PASSED:

REQUIRE( config.useColour == UseColour::Auto )

with expansion:

0 == 0

-------------------------------------------------------------------------------

Process can be configured on command line

use-colour

yes

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK\_NOTHROW( parseIntoConfig( argv, config ) )

TestMain.cpp:<line number>:

PASSED:

REQUIRE( config.useColour == UseColour::Yes )

with expansion:

1 == 1

-------------------------------------------------------------------------------

Process can be configured on command line

use-colour

no

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK\_NOTHROW( parseIntoConfig( argv, config ) )

TestMain.cpp:<line number>:

PASSED:

REQUIRE( config.useColour == UseColour::No )

with expansion:

2 == 2

-------------------------------------------------------------------------------

Process can be configured on command line

use-colour

error

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

REQUIRE\_THROWS\_WITH( parseIntoConfig( argv, config ), Contains( "colour mode must be one of" ) )

-------------------------------------------------------------------------------

Reconstruction should be based on stringification: #914

-------------------------------------------------------------------------------

DecompositionTests.cpp:<line number>

...............................................................................

DecompositionTests.cpp:<line number>: FAILED:

CHECK( truthy(false) )

with expansion:

Hey, its truthy!

-------------------------------------------------------------------------------

SCOPED\_INFO is reset for each loop

-------------------------------------------------------------------------------

MessageTests.cpp:<line number>

...............................................................................

MessageTests.cpp:<line number>:

PASSED:

REQUIRE( i < 10 )

with expansion:

0 < 10

with messages:

current counter 0

i := 0

MessageTests.cpp:<line number>:

PASSED:

REQUIRE( i < 10 )

with expansion:

1 < 10

with messages:

current counter 1

i := 1

MessageTests.cpp:<line number>:

PASSED:

REQUIRE( i < 10 )

with expansion:

2 < 10

with messages:

current counter 2

i := 2

MessageTests.cpp:<line number>:

PASSED:

REQUIRE( i < 10 )

with expansion:

3 < 10

with messages:

current counter 3

i := 3

MessageTests.cpp:<line number>:

PASSED:

REQUIRE( i < 10 )

with expansion:

4 < 10

with messages:

current counter 4

i := 4

MessageTests.cpp:<line number>:

PASSED:

REQUIRE( i < 10 )

with expansion:

5 < 10

with messages:

current counter 5

i := 5

MessageTests.cpp:<line number>:

PASSED:

REQUIRE( i < 10 )

with expansion:

6 < 10

with messages:

current counter 6

i := 6

MessageTests.cpp:<line number>:

PASSED:

REQUIRE( i < 10 )

with expansion:

7 < 10

with messages:

current counter 7

i := 7

MessageTests.cpp:<line number>:

PASSED:

REQUIRE( i < 10 )

with expansion:

8 < 10

with messages:

current counter 8

i := 8

MessageTests.cpp:<line number>:

PASSED:

REQUIRE( i < 10 )

with expansion:

9 < 10

with messages:

current counter 9

i := 9

MessageTests.cpp:<line number>: FAILED:

REQUIRE( i < 10 )

with expansion:

10 < 10

with messages:

current counter 10

i := 10

-------------------------------------------------------------------------------

SUCCEED counts as a test pass

-------------------------------------------------------------------------------

MessageTests.cpp:<line number>

...............................................................................

MessageTests.cpp:<line number>:

PASSED:

with message:

this is a success

-------------------------------------------------------------------------------

SUCCESS does not require an argument

-------------------------------------------------------------------------------

MessageTests.cpp:<line number>

...............................................................................

MessageTests.cpp:<line number>:

PASSED:

-------------------------------------------------------------------------------

Scenario: BDD tests requiring Fixtures to provide commonly-accessed data or

methods

Given: No operations precede me

-------------------------------------------------------------------------------

BDDTests.cpp:<line number>

...............................................................................

BDDTests.cpp:<line number>:

PASSED:

REQUIRE( before == 0 )

with expansion:

0 == 0

-------------------------------------------------------------------------------

Scenario: BDD tests requiring Fixtures to provide commonly-accessed data or

methods

Given: No operations precede me

When: We get the count

Then: Subsequently values are higher

-------------------------------------------------------------------------------

BDDTests.cpp:<line number>

...............................................................................

BDDTests.cpp:<line number>:

PASSED:

REQUIRE( after > before )

with expansion:

1 > 0

-------------------------------------------------------------------------------

Scenario: Do that thing with the thing

Given: This stuff exists

When: I do this

Then: it should do this

-------------------------------------------------------------------------------

BDDTests.cpp:<line number>

...............................................................................

BDDTests.cpp:<line number>:

PASSED:

REQUIRE( itDoesThis() )

with expansion:

true

-------------------------------------------------------------------------------

Scenario: Do that thing with the thing

Given: This stuff exists

When: I do this

Then: it should do this

And: do that

-------------------------------------------------------------------------------

BDDTests.cpp:<line number>

...............................................................................

BDDTests.cpp:<line number>:

PASSED:

REQUIRE( itDoesThat() )

with expansion:

true

-------------------------------------------------------------------------------

Scenario: This is a really long scenario name to see how the list command deals

with wrapping

Given: A section name that is so long that it cannot fit in a single

console width

When: The test headers are printed as part of the normal running of the

scenario

Then: The, deliberately very long and overly verbose (you see what I did

there?) section names must wrap, along with an indent

-------------------------------------------------------------------------------

BDDTests.cpp:<line number>

...............................................................................

BDDTests.cpp:<line number>:

PASSED:

with message:

boo!

-------------------------------------------------------------------------------

Scenario: Vector resizing affects size and capacity

Given: an empty vector

-------------------------------------------------------------------------------

BDDTests.cpp:<line number>

...............................................................................

BDDTests.cpp:<line number>:

PASSED:

REQUIRE( v.size() == 0 )

with expansion:

0 == 0

-------------------------------------------------------------------------------

Scenario: Vector resizing affects size and capacity

Given: an empty vector

When: it is made larger

Then: the size and capacity go up

-------------------------------------------------------------------------------

BDDTests.cpp:<line number>

...............................................................................

BDDTests.cpp:<line number>:

PASSED:

REQUIRE( v.size() == 10 )

with expansion:

10 == 10

BDDTests.cpp:<line number>:

PASSED:

REQUIRE( v.capacity() >= 10 )

with expansion:

10 >= 10

-------------------------------------------------------------------------------

Scenario: Vector resizing affects size and capacity

Given: an empty vector

When: it is made larger

Then: the size and capacity go up

And when: it is made smaller again

Then: the size goes down but the capacity stays the same

-------------------------------------------------------------------------------

BDDTests.cpp:<line number>

...............................................................................

BDDTests.cpp:<line number>:

PASSED:

REQUIRE( v.size() == 5 )

with expansion:

5 == 5

BDDTests.cpp:<line number>:

PASSED:

REQUIRE( v.capacity() >= 10 )

with expansion:

10 >= 10

-------------------------------------------------------------------------------

Scenario: Vector resizing affects size and capacity

Given: an empty vector

-------------------------------------------------------------------------------

BDDTests.cpp:<line number>

...............................................................................

BDDTests.cpp:<line number>:

PASSED:

REQUIRE( v.size() == 0 )

with expansion:

0 == 0

-------------------------------------------------------------------------------

Scenario: Vector resizing affects size and capacity

Given: an empty vector

When: we reserve more space

Then: The capacity is increased but the size remains the same

-------------------------------------------------------------------------------

BDDTests.cpp:<line number>

...............................................................................

BDDTests.cpp:<line number>:

PASSED:

REQUIRE( v.capacity() >= 10 )

with expansion:

10 >= 10

BDDTests.cpp:<line number>:

PASSED:

REQUIRE( v.size() == 0 )

with expansion:

0 == 0

A string sent directly to stdout

A string sent directly to stderr

-------------------------------------------------------------------------------

Some simple comparisons between doubles

-------------------------------------------------------------------------------

ApproxTests.cpp:<line number>

...............................................................................

ApproxTests.cpp:<line number>:

PASSED:

REQUIRE( d == Approx( 1.23 ) )

with expansion:

1.23 == Approx( 1.23 )

ApproxTests.cpp:<line number>:

PASSED:

REQUIRE( d != Approx( 1.22 ) )

with expansion:

1.23 != Approx( 1.22 )

ApproxTests.cpp:<line number>:

PASSED:

REQUIRE( d != Approx( 1.24 ) )

with expansion:

1.23 != Approx( 1.24 )

ApproxTests.cpp:<line number>:

PASSED:

REQUIRE( Approx( d ) == 1.23 )

with expansion:

Approx( 1.23 ) == 1.23

ApproxTests.cpp:<line number>:

PASSED:

REQUIRE( Approx( d ) != 1.22 )

with expansion:

Approx( 1.23 ) != 1.22

ApproxTests.cpp:<line number>:

PASSED:

REQUIRE( Approx( d ) != 1.24 )

with expansion:

Approx( 1.23 ) != 1.24

ApproxTests.cpp:<line number>:

PASSED:

REQUIRE( 0 == Approx(0) )

with expansion:

0 == Approx( 0.0 )

Write to std::cerr

-------------------------------------------------------------------------------

Standard error is reported and redirected

std::cerr

-------------------------------------------------------------------------------

MessageTests.cpp:<line number>

...............................................................................

No assertions in section 'std::cerr'

Write to std::clog

-------------------------------------------------------------------------------

Standard error is reported and redirected

std::clog

-------------------------------------------------------------------------------

MessageTests.cpp:<line number>

...............................................................................

No assertions in section 'std::clog'

Interleaved writes to error streams

-------------------------------------------------------------------------------

Standard error is reported and redirected

Interleaved writes to cerr and clog

-------------------------------------------------------------------------------

MessageTests.cpp:<line number>

...............................................................................

No assertions in section 'Interleaved writes to cerr and clog'

Message from section one

-------------------------------------------------------------------------------

Standard output from all sections is reported

one

-------------------------------------------------------------------------------

MessageTests.cpp:<line number>

...............................................................................

No assertions in section 'one'

Message from section two

-------------------------------------------------------------------------------

Standard output from all sections is reported

two

-------------------------------------------------------------------------------

MessageTests.cpp:<line number>

...............................................................................

No assertions in section 'two'

-------------------------------------------------------------------------------

StartsWith string matcher

-------------------------------------------------------------------------------

MatchersTests.cpp:<line number>

...............................................................................

MatchersTests.cpp:<line number>: FAILED:

CHECK\_THAT( testStringForMatching(), StartsWith( "string" ) )

with expansion:

"this string contains 'abc' as a substring" starts with: "string"

-------------------------------------------------------------------------------

String matchers

-------------------------------------------------------------------------------

MatchersTests.cpp:<line number>

...............................................................................

MatchersTests.cpp:<line number>:

PASSED:

REQUIRE\_THAT( testStringForMatching(), Contains( "string" ) )

with expansion:

"this string contains 'abc' as a substring" contains: "string"

MatchersTests.cpp:<line number>:

PASSED:

CHECK\_THAT( testStringForMatching(), Contains( "abc" ) )

with expansion:

"this string contains 'abc' as a substring" contains: "abc"

MatchersTests.cpp:<line number>:

PASSED:

CHECK\_THAT( testStringForMatching(), StartsWith( "this" ) )

with expansion:

"this string contains 'abc' as a substring" starts with: "this"

MatchersTests.cpp:<line number>:

PASSED:

CHECK\_THAT( testStringForMatching(), EndsWith( "substring" ) )

with expansion:

"this string contains 'abc' as a substring" ends with: "substring"

hello

hello

-------------------------------------------------------------------------------

Tabs and newlines show in output

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>: FAILED:

CHECK( s1 == s2 )

with expansion:

"if ($b == 10) {

$a = 20;

}"

==

"if ($b == 10) {

$a = 20;

}

"

-------------------------------------------------------------------------------

Tag alias can be registered against tag patterns

The same tag alias can only be registered once

-------------------------------------------------------------------------------

TagAliasTests.cpp:<line number>

...............................................................................

TagAliasTests.cpp:<line number>:

PASSED:

CHECK\_THAT( what, Contains( "[@zzz]" ) )

with expansion:

"error: tag alias, "[@zzz]" already registered.

First seen at file:2

Redefined at file:10

" contains: "[@zzz]"

TagAliasTests.cpp:<line number>:

PASSED:

CHECK\_THAT( what, Contains( "file" ) )

with expansion:

"error: tag alias, "[@zzz]" already registered.

First seen at file:2

Redefined at file:10

" contains: "file"

TagAliasTests.cpp:<line number>:

PASSED:

CHECK\_THAT( what, Contains( "2" ) )

with expansion:

"error: tag alias, "[@zzz]" already registered.

First seen at file:2

Redefined at file:10

" contains: "2"

TagAliasTests.cpp:<line number>:

PASSED:

CHECK\_THAT( what, Contains( "10" ) )

with expansion:

"error: tag alias, "[@zzz]" already registered.

First seen at file:2

Redefined at file:10

" contains: "10"

-------------------------------------------------------------------------------

Tag alias can be registered against tag patterns

Tag aliases must be of the form [@name]

-------------------------------------------------------------------------------

TagAliasTests.cpp:<line number>

...............................................................................

TagAliasTests.cpp:<line number>:

PASSED:

CHECK\_THROWS( registry.add( "[no ampersat]", "", Catch::SourceLineInfo( "file", 3 ) ) )

TagAliasTests.cpp:<line number>:

PASSED:

CHECK\_THROWS( registry.add( "[the @ is not at the start]", "", Catch::SourceLineInfo( "file", 3 ) ) )

TagAliasTests.cpp:<line number>:

PASSED:

CHECK\_THROWS( registry.add( "@no square bracket at start]", "", Catch::SourceLineInfo( "file", 3 ) ) )

TagAliasTests.cpp:<line number>:

PASSED:

CHECK\_THROWS( registry.add( "[@no square bracket at end", "", Catch::SourceLineInfo( "file", 3 ) ) )

-------------------------------------------------------------------------------

Test case with one argument

-------------------------------------------------------------------------------

VariadicMacrosTests.cpp:<line number>

...............................................................................

VariadicMacrosTests.cpp:<line number>:

PASSED:

with message:

no assertions

-------------------------------------------------------------------------------

Test enum bit values

-------------------------------------------------------------------------------

TrickyTests.cpp:<line number>

...............................................................................

TrickyTests.cpp:<line number>:

PASSED:

REQUIRE( 0x<hex digits> == bit30and31 )

with expansion:

3221225472 (0x<hex digits>) == 3221225472

-------------------------------------------------------------------------------

Text can be formatted using the Text class

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK( Text( "hi there" ).toString() == "hi there" )

with expansion:

"hi there" == "hi there"

TestMain.cpp:<line number>:

PASSED:

CHECK( Text( "hi there", narrow ).toString() == "hi\nthere" )

with expansion:

"hi

there"

==

"hi

there"

-------------------------------------------------------------------------------

The NO\_FAIL macro reports a failure but does not fail the test

-------------------------------------------------------------------------------

MessageTests.cpp:<line number>

...............................................................................

MessageTests.cpp:<line number>:

FAILED - but was ok:

CHECK\_NOFAIL( 1 == 2 )

-------------------------------------------------------------------------------

This test 'should' fail but doesn't

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

with message:

oops!

-------------------------------------------------------------------------------

Tracker

-------------------------------------------------------------------------------

PartTrackerTests.cpp:<line number>

...............................................................................

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase.isOpen() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s1.isOpen() )

with expansion:

true

-------------------------------------------------------------------------------

Tracker

successfully close one section

-------------------------------------------------------------------------------

PartTrackerTests.cpp:<line number>

...............................................................................

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s1.isSuccessfullyCompleted() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase.isComplete() == false )

with expansion:

false == false

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( ctx.completedCycle() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase.isSuccessfullyCompleted() )

with expansion:

true

-------------------------------------------------------------------------------

Tracker

-------------------------------------------------------------------------------

PartTrackerTests.cpp:<line number>

...............................................................................

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase.isOpen() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s1.isOpen() )

with expansion:

true

-------------------------------------------------------------------------------

Tracker

fail one section

-------------------------------------------------------------------------------

PartTrackerTests.cpp:<line number>

...............................................................................

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s1.isComplete() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s1.isSuccessfullyCompleted() == false )

with expansion:

false == false

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase.isComplete() == false )

with expansion:

false == false

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( ctx.completedCycle() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase.isSuccessfullyCompleted() == false )

with expansion:

false == false

-------------------------------------------------------------------------------

Tracker

fail one section

re-enter after failed section

-------------------------------------------------------------------------------

PartTrackerTests.cpp:<line number>

...............................................................................

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase2.isOpen() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s1b.isOpen() == false )

with expansion:

false == false

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( ctx.completedCycle() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase.isComplete() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase.isSuccessfullyCompleted() )

with expansion:

true

-------------------------------------------------------------------------------

Tracker

-------------------------------------------------------------------------------

PartTrackerTests.cpp:<line number>

...............................................................................

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase.isOpen() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s1.isOpen() )

with expansion:

true

-------------------------------------------------------------------------------

Tracker

fail one section

-------------------------------------------------------------------------------

PartTrackerTests.cpp:<line number>

...............................................................................

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s1.isComplete() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s1.isSuccessfullyCompleted() == false )

with expansion:

false == false

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase.isComplete() == false )

with expansion:

false == false

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( ctx.completedCycle() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase.isSuccessfullyCompleted() == false )

with expansion:

false == false

-------------------------------------------------------------------------------

Tracker

fail one section

re-enter after failed section and find next section

-------------------------------------------------------------------------------

PartTrackerTests.cpp:<line number>

...............................................................................

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase2.isOpen() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s1b.isOpen() == false )

with expansion:

false == false

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s2.isOpen() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( ctx.completedCycle() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase.isComplete() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase.isSuccessfullyCompleted() )

with expansion:

true

-------------------------------------------------------------------------------

Tracker

-------------------------------------------------------------------------------

PartTrackerTests.cpp:<line number>

...............................................................................

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase.isOpen() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s1.isOpen() )

with expansion:

true

-------------------------------------------------------------------------------

Tracker

successfully close one section, then find another

-------------------------------------------------------------------------------

PartTrackerTests.cpp:<line number>

...............................................................................

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s2.isOpen() == false )

with expansion:

false == false

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase.isComplete() == false )

with expansion:

false == false

-------------------------------------------------------------------------------

Tracker

successfully close one section, then find another

Re-enter - skips S1 and enters S2

-------------------------------------------------------------------------------

PartTrackerTests.cpp:<line number>

...............................................................................

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase2.isOpen() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s1b.isOpen() == false )

with expansion:

false == false

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s2b.isOpen() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( ctx.completedCycle() == false )

with expansion:

false == false

-------------------------------------------------------------------------------

Tracker

successfully close one section, then find another

Re-enter - skips S1 and enters S2

Successfully close S2

-------------------------------------------------------------------------------

PartTrackerTests.cpp:<line number>

...............................................................................

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( ctx.completedCycle() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s2b.isSuccessfullyCompleted() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase2.isComplete() == false )

with expansion:

false == false

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase2.isSuccessfullyCompleted() )

with expansion:

true

-------------------------------------------------------------------------------

Tracker

-------------------------------------------------------------------------------

PartTrackerTests.cpp:<line number>

...............................................................................

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase.isOpen() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s1.isOpen() )

with expansion:

true

-------------------------------------------------------------------------------

Tracker

successfully close one section, then find another

-------------------------------------------------------------------------------

PartTrackerTests.cpp:<line number>

...............................................................................

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s2.isOpen() == false )

with expansion:

false == false

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase.isComplete() == false )

with expansion:

false == false

-------------------------------------------------------------------------------

Tracker

successfully close one section, then find another

Re-enter - skips S1 and enters S2

-------------------------------------------------------------------------------

PartTrackerTests.cpp:<line number>

...............................................................................

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase2.isOpen() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s1b.isOpen() == false )

with expansion:

false == false

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s2b.isOpen() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( ctx.completedCycle() == false )

with expansion:

false == false

-------------------------------------------------------------------------------

Tracker

successfully close one section, then find another

Re-enter - skips S1 and enters S2

fail S2

-------------------------------------------------------------------------------

PartTrackerTests.cpp:<line number>

...............................................................................

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( ctx.completedCycle() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s2b.isComplete() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s2b.isSuccessfullyCompleted() == false )

with expansion:

false == false

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase2.isSuccessfullyCompleted() == false )

with expansion:

false == false

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase3.isOpen() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s1c.isOpen() == false )

with expansion:

false == false

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s2c.isOpen() == false )

with expansion:

false == false

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase3.isSuccessfullyCompleted() )

with expansion:

true

-------------------------------------------------------------------------------

Tracker

-------------------------------------------------------------------------------

PartTrackerTests.cpp:<line number>

...............................................................................

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase.isOpen() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s1.isOpen() )

with expansion:

true

-------------------------------------------------------------------------------

Tracker

open a nested section

-------------------------------------------------------------------------------

PartTrackerTests.cpp:<line number>

...............................................................................

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s2.isOpen() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s2.isComplete() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s1.isComplete() == false )

with expansion:

false == false

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s1.isComplete() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase.isComplete() == false )

with expansion:

false == false

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase.isComplete() )

with expansion:

true

-------------------------------------------------------------------------------

Tracker

-------------------------------------------------------------------------------

PartTrackerTests.cpp:<line number>

...............................................................................

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase.isOpen() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s1.isOpen() )

with expansion:

true

-------------------------------------------------------------------------------

Tracker

start a generator

-------------------------------------------------------------------------------

PartTrackerTests.cpp:<line number>

...............................................................................

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( g1.isOpen() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( g1.index() == 0 )

with expansion:

0 == 0

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( g1.isComplete() == false )

with expansion:

false == false

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s1.isComplete() == false )

with expansion:

false == false

-------------------------------------------------------------------------------

Tracker

start a generator

close outer section

-------------------------------------------------------------------------------

PartTrackerTests.cpp:<line number>

...............................................................................

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s1.isComplete() == false )

with expansion:

false == false

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase.isSuccessfullyCompleted() == false )

with expansion:

false == false

-------------------------------------------------------------------------------

Tracker

start a generator

close outer section

Re-enter for second generation

-------------------------------------------------------------------------------

PartTrackerTests.cpp:<line number>

...............................................................................

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase2.isOpen() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s1b.isOpen() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( g1b.isOpen() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( g1b.index() == 1 )

with expansion:

1 == 1

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s1.isComplete() == false )

with expansion:

false == false

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s1b.isComplete() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( g1b.isComplete() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase2.isComplete() )

with expansion:

true

-------------------------------------------------------------------------------

Tracker

-------------------------------------------------------------------------------

PartTrackerTests.cpp:<line number>

...............................................................................

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase.isOpen() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s1.isOpen() )

with expansion:

true

-------------------------------------------------------------------------------

Tracker

start a generator

-------------------------------------------------------------------------------

PartTrackerTests.cpp:<line number>

...............................................................................

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( g1.isOpen() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( g1.index() == 0 )

with expansion:

0 == 0

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( g1.isComplete() == false )

with expansion:

false == false

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s1.isComplete() == false )

with expansion:

false == false

-------------------------------------------------------------------------------

Tracker

start a generator

Start a new inner section

-------------------------------------------------------------------------------

PartTrackerTests.cpp:<line number>

...............................................................................

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s2.isOpen() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s2.isComplete() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s1.isComplete() == false )

with expansion:

false == false

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase.isComplete() == false )

with expansion:

false == false

-------------------------------------------------------------------------------

Tracker

start a generator

Start a new inner section

Re-enter for second generation

-------------------------------------------------------------------------------

PartTrackerTests.cpp:<line number>

...............................................................................

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase2.isOpen() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s1b.isOpen() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( g1b.isOpen() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( g1b.index() == 1 )

with expansion:

1 == 1

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s2b.isOpen() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s2b.isComplete() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( g1b.isComplete() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s1b.isComplete() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase2.isComplete() )

with expansion:

true

-------------------------------------------------------------------------------

Tracker

-------------------------------------------------------------------------------

PartTrackerTests.cpp:<line number>

...............................................................................

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase.isOpen() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s1.isOpen() )

with expansion:

true

-------------------------------------------------------------------------------

Tracker

start a generator

-------------------------------------------------------------------------------

PartTrackerTests.cpp:<line number>

...............................................................................

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( g1.isOpen() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( g1.index() == 0 )

with expansion:

0 == 0

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( g1.isComplete() == false )

with expansion:

false == false

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s1.isComplete() == false )

with expansion:

false == false

-------------------------------------------------------------------------------

Tracker

start a generator

Fail an inner section

-------------------------------------------------------------------------------

PartTrackerTests.cpp:<line number>

...............................................................................

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s2.isOpen() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s2.isComplete() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s2.isSuccessfullyCompleted() == false )

with expansion:

false == false

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s1.isComplete() == false )

with expansion:

false == false

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase.isComplete() == false )

with expansion:

false == false

-------------------------------------------------------------------------------

Tracker

start a generator

Fail an inner section

Re-enter for second generation

-------------------------------------------------------------------------------

PartTrackerTests.cpp:<line number>

...............................................................................

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase2.isOpen() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s1b.isOpen() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( g1b.isOpen() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( g1b.index() == 0 )

with expansion:

0 == 0

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s2b.isOpen() == false )

with expansion:

false == false

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( g1b.isComplete() == false )

with expansion:

false == false

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s1b.isComplete() == false )

with expansion:

false == false

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase2.isComplete() == false )

with expansion:

false == false

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase3.isOpen() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s1c.isOpen() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( g1c.isOpen() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( g1c.index() == 1 )

with expansion:

1 == 1

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s2c.isOpen() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s2c.isComplete() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( g1c.isComplete() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( s1c.isComplete() )

with expansion:

true

PartTrackerTests.cpp:<line number>:

PASSED:

REQUIRE( testCase3.isComplete() )

with expansion:

true

-------------------------------------------------------------------------------

Unexpected exceptions can be translated

-------------------------------------------------------------------------------

ExceptionTests.cpp:<line number>

...............................................................................

ExceptionTests.cpp:<line number>: FAILED:

due to unexpected exception with message:

3.14

-------------------------------------------------------------------------------

Use a custom approx

-------------------------------------------------------------------------------

ApproxTests.cpp:<line number>

...............................................................................

ApproxTests.cpp:<line number>:

PASSED:

REQUIRE( d == approx( 1.23 ) )

with expansion:

1.23 == Approx( 1.23 )

ApproxTests.cpp:<line number>:

PASSED:

REQUIRE( d == approx( 1.22 ) )

with expansion:

1.23 == Approx( 1.22 )

ApproxTests.cpp:<line number>:

PASSED:

REQUIRE( d == approx( 1.24 ) )

with expansion:

1.23 == Approx( 1.24 )

ApproxTests.cpp:<line number>:

PASSED:

REQUIRE( d != approx( 1.25 ) )

with expansion:

1.23 != Approx( 1.25 )

ApproxTests.cpp:<line number>:

PASSED:

REQUIRE( approx( d ) == 1.23 )

with expansion:

Approx( 1.23 ) == 1.23

ApproxTests.cpp:<line number>:

PASSED:

REQUIRE( approx( d ) == 1.22 )

with expansion:

Approx( 1.23 ) == 1.22

ApproxTests.cpp:<line number>:

PASSED:

REQUIRE( approx( d ) == 1.24 )

with expansion:

Approx( 1.23 ) == 1.24

ApproxTests.cpp:<line number>:

PASSED:

REQUIRE( approx( d ) != 1.25 )

with expansion:

Approx( 1.23 ) != 1.25

-------------------------------------------------------------------------------

Variadic macros

Section with one argument

-------------------------------------------------------------------------------

VariadicMacrosTests.cpp:<line number>

...............................................................................

VariadicMacrosTests.cpp:<line number>:

PASSED:

with message:

no assertions

-------------------------------------------------------------------------------

Vector matchers

Contains (element)

-------------------------------------------------------------------------------

MatchersTests.cpp:<line number>

...............................................................................

MatchersTests.cpp:<line number>:

PASSED:

CHECK\_THAT( v, VectorContains( 1 ) )

with expansion:

{ 1, 2, 3 } Contains: 1

MatchersTests.cpp:<line number>:

PASSED:

CHECK\_THAT( v, VectorContains( 2 ) )

with expansion:

{ 1, 2, 3 } Contains: 2

-------------------------------------------------------------------------------

Vector matchers

Contains (vector)

-------------------------------------------------------------------------------

MatchersTests.cpp:<line number>

...............................................................................

MatchersTests.cpp:<line number>:

PASSED:

CHECK\_THAT( v, Contains( v2 ) )

with expansion:

{ 1, 2, 3 } Contains: { 1, 2 }

MatchersTests.cpp:<line number>:

PASSED:

CHECK\_THAT( v, Contains( v2 ) )

with expansion:

{ 1, 2, 3 } Contains: { 1, 2, 3 }

MatchersTests.cpp:<line number>:

PASSED:

CHECK\_THAT( v, Contains( empty) )

with expansion:

{ 1, 2, 3 } Contains: { }

MatchersTests.cpp:<line number>:

PASSED:

CHECK\_THAT( empty, Contains( empty) )

with expansion:

{ } Contains: { }

-------------------------------------------------------------------------------

Vector matchers

Equals

-------------------------------------------------------------------------------

MatchersTests.cpp:<line number>

...............................................................................

MatchersTests.cpp:<line number>:

PASSED:

CHECK\_THAT( v, Equals( v ) )

with expansion:

{ 1, 2, 3 } Equals: { 1, 2, 3 }

MatchersTests.cpp:<line number>:

PASSED:

CHECK\_THAT( empty, Equals( empty ) )

with expansion:

{ } Equals: { }

MatchersTests.cpp:<line number>:

PASSED:

CHECK\_THAT( v, Equals( v2 ) )

with expansion:

{ 1, 2, 3 } Equals: { 1, 2, 3 }

-------------------------------------------------------------------------------

Vector matchers that fail

Contains (element)

-------------------------------------------------------------------------------

MatchersTests.cpp:<line number>

...............................................................................

MatchersTests.cpp:<line number>: FAILED:

CHECK\_THAT( v, VectorContains( -1 ) )

with expansion:

{ 1, 2, 3 } Contains: -1

MatchersTests.cpp:<line number>: FAILED:

CHECK\_THAT( empty, VectorContains( 1 ) )

with expansion:

{ } Contains: 1

-------------------------------------------------------------------------------

Vector matchers that fail

Contains (vector)

-------------------------------------------------------------------------------

MatchersTests.cpp:<line number>

...............................................................................

MatchersTests.cpp:<line number>: FAILED:

CHECK\_THAT( empty, Contains( v) )

with expansion:

{ } Contains: { 1, 2, 3 }

MatchersTests.cpp:<line number>: FAILED:

CHECK\_THAT( v, Contains( v2 ) )

with expansion:

{ 1, 2, 3 } Contains: { 1, 2, 4 }

-------------------------------------------------------------------------------

Vector matchers that fail

Equals

-------------------------------------------------------------------------------

MatchersTests.cpp:<line number>

...............................................................................

MatchersTests.cpp:<line number>: FAILED:

CHECK\_THAT( v, Equals( v2 ) )

with expansion:

{ 1, 2, 3 } Equals: { 1, 2 }

MatchersTests.cpp:<line number>: FAILED:

CHECK\_THAT( v2, Equals( v ) )

with expansion:

{ 1, 2 } Equals: { 1, 2, 3 }

MatchersTests.cpp:<line number>: FAILED:

CHECK\_THAT( empty, Equals( v ) )

with expansion:

{ } Equals: { 1, 2, 3 }

MatchersTests.cpp:<line number>: FAILED:

CHECK\_THAT( v, Equals( empty ) )

with expansion:

{ 1, 2, 3 } Equals: { }

-------------------------------------------------------------------------------

When checked exceptions are thrown they can be expected or unexpected

-------------------------------------------------------------------------------

ExceptionTests.cpp:<line number>

...............................................................................

ExceptionTests.cpp:<line number>:

PASSED:

REQUIRE\_THROWS\_AS( thisThrows(), std::domain\_error )

ExceptionTests.cpp:<line number>:

PASSED:

REQUIRE\_NOTHROW( thisDoesntThrow() )

ExceptionTests.cpp:<line number>:

PASSED:

REQUIRE\_THROWS( thisThrows() )

-------------------------------------------------------------------------------

When unchecked exceptions are thrown directly they are always failures

-------------------------------------------------------------------------------

ExceptionTests.cpp:<line number>

...............................................................................

ExceptionTests.cpp:<line number>: FAILED:

due to unexpected exception with message:

unexpected exception

-------------------------------------------------------------------------------

When unchecked exceptions are thrown during a CHECK the test should continue

-------------------------------------------------------------------------------

ExceptionTests.cpp:<line number>

...............................................................................

ExceptionTests.cpp:<line number>: FAILED:

CHECK( thisThrows() == 0 )

due to unexpected exception with message:

expected exception

-------------------------------------------------------------------------------

When unchecked exceptions are thrown during a REQUIRE the test should abort

fail

-------------------------------------------------------------------------------

ExceptionTests.cpp:<line number>

...............................................................................

ExceptionTests.cpp:<line number>: FAILED:

REQUIRE( thisThrows() == 0 )

due to unexpected exception with message:

expected exception

-------------------------------------------------------------------------------

When unchecked exceptions are thrown from functions they are always failures

-------------------------------------------------------------------------------

ExceptionTests.cpp:<line number>

...............................................................................

ExceptionTests.cpp:<line number>: FAILED:

CHECK( thisThrows() == 0 )

due to unexpected exception with message:

expected exception

-------------------------------------------------------------------------------

When unchecked exceptions are thrown from sections they are always failures

section name

-------------------------------------------------------------------------------

ExceptionTests.cpp:<line number>

...............................................................................

ExceptionTests.cpp:<line number>: FAILED:

due to unexpected exception with message:

unexpected exception

-------------------------------------------------------------------------------

Where the LHS is not a simple value

-------------------------------------------------------------------------------

TrickyTests.cpp:<line number>

...............................................................................

TrickyTests.cpp:<line number>:

warning:

Uncomment the code in this test to check that it gives a sensible compiler

error

-------------------------------------------------------------------------------

Where there is more to the expression after the RHS

-------------------------------------------------------------------------------

TrickyTests.cpp:<line number>

...............................................................................

TrickyTests.cpp:<line number>:

warning:

Uncomment the code in this test to check that it gives a sensible compiler

error

-------------------------------------------------------------------------------

X/level/0/a

-------------------------------------------------------------------------------

TrickyTests.cpp:<line number>

...............................................................................

TrickyTests.cpp:<line number>:

PASSED:

-------------------------------------------------------------------------------

X/level/0/b

-------------------------------------------------------------------------------

TrickyTests.cpp:<line number>

...............................................................................

TrickyTests.cpp:<line number>:

PASSED:

-------------------------------------------------------------------------------

X/level/1/a

-------------------------------------------------------------------------------

TrickyTests.cpp:<line number>

...............................................................................

TrickyTests.cpp:<line number>:

PASSED:

-------------------------------------------------------------------------------

X/level/1/b

-------------------------------------------------------------------------------

TrickyTests.cpp:<line number>

...............................................................................

TrickyTests.cpp:<line number>:

PASSED:

-------------------------------------------------------------------------------

XmlEncode

normal string

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( encode( "normal string" ) == "normal string" )

with expansion:

"normal string" == "normal string"

-------------------------------------------------------------------------------

XmlEncode

empty string

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( encode( "" ) == "" )

with expansion:

"" == ""

-------------------------------------------------------------------------------

XmlEncode

string with ampersand

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( encode( "smith & jones" ) == "smith &amp; jones" )

with expansion:

"smith &amp; jones" == "smith &amp; jones"

-------------------------------------------------------------------------------

XmlEncode

string with less-than

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( encode( "smith < jones" ) == "smith &lt; jones" )

with expansion:

"smith &lt; jones" == "smith &lt; jones"

-------------------------------------------------------------------------------

XmlEncode

string with greater-than

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( encode( "smith > jones" ) == "smith > jones" )

with expansion:

"smith > jones" == "smith > jones"

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( encode( "smith ]]> jones" ) == "smith ]]&gt; jones" )

with expansion:

"smith ]]&gt; jones"

==

"smith ]]&gt; jones"

-------------------------------------------------------------------------------

XmlEncode

string with quotes

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( encode( stringWithQuotes ) == stringWithQuotes )

with expansion:

"don't "quote" me on that"

==

"don't "quote" me on that"

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( encode( stringWithQuotes, Catch::XmlEncode::ForAttributes ) == "don't &quot;quote&quot; me on that" )

with expansion:

"don't &quot;quote&quot; me on that"

==

"don't &quot;quote&quot; me on that"

-------------------------------------------------------------------------------

XmlEncode

string with control char (1)

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( encode( "[\x01]" ) == "[\\x01]" )

with expansion:

"[\x01]" == "[\x01]"

-------------------------------------------------------------------------------

XmlEncode

string with control char (x7F)

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( encode( "[\x7F]" ) == "[\\x7F]" )

with expansion:

"[\x7F]" == "[\x7F]"

-------------------------------------------------------------------------------

atomic if

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( x == 0 )

with expansion:

0 == 0

-------------------------------------------------------------------------------

boolean member

-------------------------------------------------------------------------------

TrickyTests.cpp:<line number>

...............................................................................

TrickyTests.cpp:<line number>:

PASSED:

REQUIRE( obj.prop != 0 )

with expansion:

0x<hex digits> != 0

-------------------------------------------------------------------------------

checkedElse

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

CHECKED\_ELSE( flag )

with expansion:

true

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( testCheckedElse( true ) )

with expansion:

true

-------------------------------------------------------------------------------

checkedElse, failing

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>: FAILED:

CHECKED\_ELSE( flag )

with expansion:

false

MiscTests.cpp:<line number>: FAILED:

REQUIRE( testCheckedElse( false ) )

with expansion:

false

-------------------------------------------------------------------------------

checkedIf

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

CHECKED\_IF( flag )

with expansion:

true

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( testCheckedIf( true ) )

with expansion:

true

-------------------------------------------------------------------------------

checkedIf, failing

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>: FAILED:

CHECKED\_IF( flag )

with expansion:

false

MiscTests.cpp:<line number>: FAILED:

REQUIRE( testCheckedIf( false ) )

with expansion:

false

-------------------------------------------------------------------------------

comparisons between const int variables

-------------------------------------------------------------------------------

ConditionTests.cpp:<line number>

...............................................................................

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( unsigned\_char\_var == 1 )

with expansion:

1 == 1

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( unsigned\_short\_var == 1 )

with expansion:

1 == 1

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( unsigned\_int\_var == 1 )

with expansion:

1 == 1

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( unsigned\_long\_var == 1 )

with expansion:

1 == 1

-------------------------------------------------------------------------------

comparisons between int variables

-------------------------------------------------------------------------------

ConditionTests.cpp:<line number>

...............................................................................

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( long\_var == unsigned\_char\_var )

with expansion:

1 == 1

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( long\_var == unsigned\_short\_var )

with expansion:

1 == 1

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( long\_var == unsigned\_int\_var )

with expansion:

1 == 1

ConditionTests.cpp:<line number>:

PASSED:

REQUIRE( long\_var == unsigned\_long\_var )

with expansion:

1 == 1

-------------------------------------------------------------------------------

even more nested SECTION tests

c

d (leaf)

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

-------------------------------------------------------------------------------

even more nested SECTION tests

c

e (leaf)

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

-------------------------------------------------------------------------------

even more nested SECTION tests

f (leaf)

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

spanner-------------------------------------------------------------------------------

just failure

-------------------------------------------------------------------------------

MessageTests.cpp:<line number>

...............................................................................

MessageTests.cpp:<line number>: FAILED:

explicitly with message:

Previous info should not be seen

-------------------------------------------------------------------------------

looped SECTION tests

s1

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>: FAILED:

CHECK( b > a )

with expansion:

0 > 1

-------------------------------------------------------------------------------

looped tests

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>: FAILED:

CHECK( ( fib[i] % 2 ) == 0 )

with expansion:

1 == 0

with message:

Testing if fib[0] (1) is even

MiscTests.cpp:<line number>: FAILED:

CHECK( ( fib[i] % 2 ) == 0 )

with expansion:

1 == 0

with message:

Testing if fib[1] (1) is even

MiscTests.cpp:<line number>:

PASSED:

CHECK( ( fib[i] % 2 ) == 0 )

with expansion:

0 == 0

with message:

Testing if fib[2] (2) is even

MiscTests.cpp:<line number>: FAILED:

CHECK( ( fib[i] % 2 ) == 0 )

with expansion:

1 == 0

with message:

Testing if fib[3] (3) is even

MiscTests.cpp:<line number>: FAILED:

CHECK( ( fib[i] % 2 ) == 0 )

with expansion:

1 == 0

with message:

Testing if fib[4] (5) is even

MiscTests.cpp:<line number>:

PASSED:

CHECK( ( fib[i] % 2 ) == 0 )

with expansion:

0 == 0

with message:

Testing if fib[5] (8) is even

MiscTests.cpp:<line number>: FAILED:

CHECK( ( fib[i] % 2 ) == 0 )

with expansion:

1 == 0

with message:

Testing if fib[6] (13) is even

MiscTests.cpp:<line number>: FAILED:

CHECK( ( fib[i] % 2 ) == 0 )

with expansion:

1 == 0

with message:

Testing if fib[7] (21) is even

-------------------------------------------------------------------------------

more nested SECTION tests

s1

s2

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>: FAILED:

REQUIRE( a == b )

with expansion:

1 == 2

-------------------------------------------------------------------------------

more nested SECTION tests

s1

s3

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( a != b )

with expansion:

1 != 2

-------------------------------------------------------------------------------

more nested SECTION tests

s1

s4

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( a < b )

with expansion:

1 < 2

-------------------------------------------------------------------------------

nested SECTION tests

s1

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( a != b )

with expansion:

1 != 2

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( b != a )

with expansion:

2 != 1

-------------------------------------------------------------------------------

nested SECTION tests

s1

s2

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( a != b )

with expansion:

1 != 2

-------------------------------------------------------------------------------

non streamable - with conv. op

-------------------------------------------------------------------------------

TrickyTests.cpp:<line number>

...............................................................................

TrickyTests.cpp:<line number>:

PASSED:

REQUIRE( s == "7" )

with expansion:

"7" == "7"

-------------------------------------------------------------------------------

not allowed

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

-------------------------------------------------------------------------------

null strings

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( makeString( false ) != static\_cast<char\*>(0) )

with expansion:

"valid string" != {null string}

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( makeString( true ) == static\_cast<char\*>(0) )

with expansion:

{null string} == {null string}

-------------------------------------------------------------------------------

pair<pair<int,const char \*,pair<std::string,int> > -> toString

-------------------------------------------------------------------------------

ToStringPair.cpp:<line number>

...............................................................................

ToStringPair.cpp:<line number>:

PASSED:

REQUIRE( Catch::toString( pair ) == "{ { 42, \"Arthur\" }, { \"Ford\", 24 } }" )

with expansion:

"{ { 42, "Arthur" }, { "Ford", 24 } }"

==

"{ { 42, "Arthur" }, { "Ford", 24 } }"

-------------------------------------------------------------------------------

pointer to class

-------------------------------------------------------------------------------

TrickyTests.cpp:<line number>

...............................................................................

TrickyTests.cpp:<line number>:

PASSED:

REQUIRE( p == 0 )

with expansion:

NULL == 0

-------------------------------------------------------------------------------

random SECTION tests

s1

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( a != b )

with expansion:

1 != 2

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( b != a )

with expansion:

2 != 1

-------------------------------------------------------------------------------

random SECTION tests

s2

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( a != b )

with expansion:

1 != 2

-------------------------------------------------------------------------------

replaceInPlace

replace single char

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK( replaceInPlace( letters, "b", "z" ) )

with expansion:

true

TestMain.cpp:<line number>:

PASSED:

CHECK( letters == "azcdefcg" )

with expansion:

"azcdefcg" == "azcdefcg"

-------------------------------------------------------------------------------

replaceInPlace

replace two chars

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK( replaceInPlace( letters, "c", "z" ) )

with expansion:

true

TestMain.cpp:<line number>:

PASSED:

CHECK( letters == "abzdefzg" )

with expansion:

"abzdefzg" == "abzdefzg"

-------------------------------------------------------------------------------

replaceInPlace

replace first char

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK( replaceInPlace( letters, "a", "z" ) )

with expansion:

true

TestMain.cpp:<line number>:

PASSED:

CHECK( letters == "zbcdefcg" )

with expansion:

"zbcdefcg" == "zbcdefcg"

-------------------------------------------------------------------------------

replaceInPlace

replace last char

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK( replaceInPlace( letters, "g", "z" ) )

with expansion:

true

TestMain.cpp:<line number>:

PASSED:

CHECK( letters == "abcdefcz" )

with expansion:

"abcdefcz" == "abcdefcz"

-------------------------------------------------------------------------------

replaceInPlace

replace all chars

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK( replaceInPlace( letters, letters, "replaced" ) )

with expansion:

true

TestMain.cpp:<line number>:

PASSED:

CHECK( letters == "replaced" )

with expansion:

"replaced" == "replaced"

-------------------------------------------------------------------------------

replaceInPlace

replace no chars

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK\_FALSE( replaceInPlace( letters, "x", "z" ) )

with expansion:

!false

TestMain.cpp:<line number>:

PASSED:

CHECK( letters == letters )

with expansion:

"abcdefcg" == "abcdefcg"

-------------------------------------------------------------------------------

replaceInPlace

escape '

-------------------------------------------------------------------------------

TestMain.cpp:<line number>

...............................................................................

TestMain.cpp:<line number>:

PASSED:

CHECK( replaceInPlace( s, "'", "|'" ) )

with expansion:

true

TestMain.cpp:<line number>:

PASSED:

CHECK( s == "didn|'t" )

with expansion:

"didn|'t" == "didn|'t"

-------------------------------------------------------------------------------

send a single char to INFO

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>: FAILED:

REQUIRE( false )

with message:

3

-------------------------------------------------------------------------------

sends information to INFO

-------------------------------------------------------------------------------

MessageTests.cpp:<line number>

...............................................................................

MessageTests.cpp:<line number>: FAILED:

REQUIRE( false )

with messages:

hi

i := 7

-------------------------------------------------------------------------------

std::pair<int,const std::string> -> toString

-------------------------------------------------------------------------------

ToStringPair.cpp:<line number>

...............................................................................

ToStringPair.cpp:<line number>:

PASSED:

REQUIRE( Catch::toString(value) == "{ 34, \"xyzzy\" }" )

with expansion:

"{ 34, "xyzzy" }" == "{ 34, "xyzzy" }"

-------------------------------------------------------------------------------

std::pair<int,std::string> -> toString

-------------------------------------------------------------------------------

ToStringPair.cpp:<line number>

...............................................................................

ToStringPair.cpp:<line number>:

PASSED:

REQUIRE( Catch::toString( value ) == "{ 34, \"xyzzy\" }" )

with expansion:

"{ 34, "xyzzy" }" == "{ 34, "xyzzy" }"

-------------------------------------------------------------------------------

std::vector<std::pair<std::string,int> > -> toString

-------------------------------------------------------------------------------

ToStringPair.cpp:<line number>

...............................................................................

ToStringPair.cpp:<line number>:

PASSED:

REQUIRE( Catch::toString( pr ) == "{ { \"green\", 55 } }" )

with expansion:

"{ { "green", 55 } }"

==

"{ { "green", 55 } }"

-------------------------------------------------------------------------------

string literals of different sizes can be compared

-------------------------------------------------------------------------------

TrickyTests.cpp:<line number>

...............................................................................

TrickyTests.cpp:<line number>: FAILED:

REQUIRE( std::string( "first" ) == "second" )

with expansion:

"first" == "second"

-------------------------------------------------------------------------------

toString on const wchar\_t const pointer returns the string contents

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

CHECK( result == "\"wide load\"" )

with expansion:

""wide load"" == ""wide load""

-------------------------------------------------------------------------------

toString on const wchar\_t pointer returns the string contents

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

CHECK( result == "\"wide load\"" )

with expansion:

""wide load"" == ""wide load""

-------------------------------------------------------------------------------

toString on wchar\_t const pointer returns the string contents

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

CHECK( result == "\"wide load\"" )

with expansion:

""wide load"" == ""wide load""

-------------------------------------------------------------------------------

toString on wchar\_t returns the string contents

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

CHECK( result == "\"wide load\"" )

with expansion:

""wide load"" == ""wide load""

-------------------------------------------------------------------------------

toString( has\_maker )

-------------------------------------------------------------------------------

ToStringWhich.cpp:<line number>

...............................................................................

ToStringWhich.cpp:<line number>:

PASSED:

REQUIRE( Catch::toString( item ) == "StringMaker<has\_maker>" )

with expansion:

"StringMaker<has\_maker>"

==

"StringMaker<has\_maker>"

-------------------------------------------------------------------------------

toString( has\_maker\_and\_toString )

-------------------------------------------------------------------------------

ToStringWhich.cpp:<line number>

...............................................................................

ToStringWhich.cpp:<line number>:

PASSED:

REQUIRE( Catch::toString( item ) == "toString( has\_maker\_and\_toString )" )

with expansion:

"toString( has\_maker\_and\_toString )"

==

"toString( has\_maker\_and\_toString )"

-------------------------------------------------------------------------------

toString( has\_toString )

-------------------------------------------------------------------------------

ToStringWhich.cpp:<line number>

...............................................................................

ToStringWhich.cpp:<line number>:

PASSED:

REQUIRE( Catch::toString( item ) == "toString( has\_toString )" )

with expansion:

"toString( has\_toString )"

==

"toString( has\_toString )"

-------------------------------------------------------------------------------

toString( vectors<has\_maker )

-------------------------------------------------------------------------------

ToStringWhich.cpp:<line number>

...............................................................................

ToStringWhich.cpp:<line number>:

PASSED:

REQUIRE( Catch::toString( v ) == "{ StringMaker<has\_maker> }" )

with expansion:

"{ StringMaker<has\_maker> }"

==

"{ StringMaker<has\_maker> }"

-------------------------------------------------------------------------------

toString(enum w/operator<<)

-------------------------------------------------------------------------------

EnumToString.cpp:<line number>

...............................................................................

EnumToString.cpp:<line number>:

PASSED:

CHECK( Catch::toString(e0) == "E2{0}" )

with expansion:

"E2{0}" == "E2{0}"

EnumToString.cpp:<line number>:

PASSED:

CHECK( Catch::toString(e1) == "E2{1}" )

with expansion:

"E2{1}" == "E2{1}"

-------------------------------------------------------------------------------

toString(enum)

-------------------------------------------------------------------------------

EnumToString.cpp:<line number>

...............................................................................

EnumToString.cpp:<line number>:

PASSED:

CHECK( Catch::toString(e0) == "0" )

with expansion:

"0" == "0"

EnumToString.cpp:<line number>:

PASSED:

CHECK( Catch::toString(e1) == "1" )

with expansion:

"1" == "1"

-------------------------------------------------------------------------------

vector<int> -> toString

-------------------------------------------------------------------------------

ToStringVector.cpp:<line number>

...............................................................................

ToStringVector.cpp:<line number>:

PASSED:

REQUIRE( Catch::toString(vv) == "{ }" )

with expansion:

"{ }" == "{ }"

ToStringVector.cpp:<line number>:

PASSED:

REQUIRE( Catch::toString(vv) == "{ 42 }" )

with expansion:

"{ 42 }" == "{ 42 }"

ToStringVector.cpp:<line number>:

PASSED:

REQUIRE( Catch::toString(vv) == "{ 42, 250 }" )

with expansion:

"{ 42, 250 }" == "{ 42, 250 }"

-------------------------------------------------------------------------------

vector<string> -> toString

-------------------------------------------------------------------------------

ToStringVector.cpp:<line number>

...............................................................................

ToStringVector.cpp:<line number>:

PASSED:

REQUIRE( Catch::toString(vv) == "{ }" )

with expansion:

"{ }" == "{ }"

ToStringVector.cpp:<line number>:

PASSED:

REQUIRE( Catch::toString(vv) == "{ \"hello\" }" )

with expansion:

"{ "hello" }" == "{ "hello" }"

ToStringVector.cpp:<line number>:

PASSED:

REQUIRE( Catch::toString(vv) == "{ \"hello\", \"world\" }" )

with expansion:

"{ "hello", "world" }"

==

"{ "hello", "world" }"

-------------------------------------------------------------------------------

vectors can be sized and resized

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( v.size() == 5 )

with expansion:

5 == 5

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( v.capacity() >= 5 )

with expansion:

5 >= 5

-------------------------------------------------------------------------------

vectors can be sized and resized

resizing bigger changes size and capacity

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( v.size() == 10 )

with expansion:

10 == 10

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( v.capacity() >= 10 )

with expansion:

10 >= 10

-------------------------------------------------------------------------------

vectors can be sized and resized

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( v.size() == 5 )

with expansion:

5 == 5

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( v.capacity() >= 5 )

with expansion:

5 >= 5

-------------------------------------------------------------------------------

vectors can be sized and resized

resizing smaller changes size but not capacity

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( v.size() == 0 )

with expansion:

0 == 0

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( v.capacity() >= 5 )

with expansion:

5 >= 5

-------------------------------------------------------------------------------

vectors can be sized and resized

resizing smaller changes size but not capacity

We can use the 'swap trick' to reset the capacity

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( v.capacity() == 0 )

with expansion:

0 == 0

-------------------------------------------------------------------------------

vectors can be sized and resized

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( v.size() == 5 )

with expansion:

5 == 5

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( v.capacity() >= 5 )

with expansion:

5 >= 5

-------------------------------------------------------------------------------

vectors can be sized and resized

reserving bigger changes capacity but not size

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( v.size() == 5 )

with expansion:

5 == 5

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( v.capacity() >= 10 )

with expansion:

10 >= 10

-------------------------------------------------------------------------------

vectors can be sized and resized

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( v.size() == 5 )

with expansion:

5 == 5

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( v.capacity() >= 5 )

with expansion:

5 >= 5

-------------------------------------------------------------------------------

vectors can be sized and resized

reserving smaller does not change size or capacity

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( v.size() == 5 )

with expansion:

5 == 5

MiscTests.cpp:<line number>:

PASSED:

REQUIRE( v.capacity() >= 5 )

with expansion:

5 >= 5

-------------------------------------------------------------------------------

xmlentitycheck

embedded xml

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

-------------------------------------------------------------------------------

xmlentitycheck

encoded chars

-------------------------------------------------------------------------------

MiscTests.cpp:<line number>

...............................................................................

MiscTests.cpp:<line number>:

PASSED:

===============================================================================

test cases: 171 | 120 passed | 47 failed | 4 failed as expected

assertions: 985 | 871 passed | 93 failed | 21 failed as expected