Team Assembly
Test Cases & Defect Management Version 1.0.0

March 8, 2020

ExC Compiler Test Cases

Test Case ID (testNumber_s#_ type)	**Module**	**Description**	**Steps**	**Prerequisites**	**Test Data**	**Test Number**	**Stage Number**	**Shortname**
001_S1_Valid_Return0	Compiler	Validate an int return function with return 0 and no parameters.	1. Run compiler with .c test data name as input parameter. 2. Verify compiler output.	Elixir environment ready and .c file loaded into test directory. Target assembly code ready to compare.	001_S1_Valid_Return0.c	001	1	Return0
002_S1_Valid_Return7	Compiler	Validate an int return function with return 7 and no parameters.	1. Run compiler with .c test data name as input parameter. 2. Verify compiler output.	Elixir environment ready and .c file loaded into test directory. Target assembly code ready to compare.	002_S1_Valid_Return7.c	002	1	Return7
003_S1_Valid_ReturnMD130	Compiler	Validate an int return function with multi digit return of 130. The function has no input parameters.	1. Run compiler with .c test data name as input parameter. 2. Verify compiler output.	Elixir environment ready and .c file loaded into test directory. Target assembly code ready to compare.	003_S1_Valid_ReturnMD130.c	003	1	ReturnMD130
004_S1_Valid_ReturnBlankSpaces	Compiler	Validate an int return main function with blank spaces and new lines separating each element that would comprise a token.	1. Run compiler with .c test data name as input parameter. 2. Verify compiler output.	Elixir environment ready and .c file loaded into test directory. Target assembly code ready to compare.	004_S1_Valid_ReturnBlankSpaces.c	004	1	ReturnBlankSpaces
005_S1_Valid_ReturnNoLineB	Compiler	Validate an int return main function with no spaces between each element considered as a token.	1. Run compiler with .c test data name as input parameter. 2. Verify compiler output.	Elixir environment ready and .c file loaded into test directory. Target assembly code ready to compare.	005_S1_Valid_ReturnNoLineB.c	005	1	ReturnNoLineB
006_S1_Valid_ReturnSpaceChars	Compiler	Validate an int return main function with different spacing characters such as tab, space or new line between each token.	1. Run compiler with .c test data name as input parameter. 2. Verify compiler output.	Elixir environment ready and .c file loaded into test directory. Target assembly code ready to compare.	006_S1_Valid_ReturnSpaceChars.c	006	1	ReturnSpaceChars
007_S1_Invalid_ReturnNull	Compiler	Validate an int return main function with no return value.	1. Run compiler with .c test data name as input parameter. 2. Verify compiler output. 3. As the test has an invalid input file, no assembly file nor executable should generate.	Elixir environment ready and .c file loaded into test directory. Target assembly code ready to compare.	007_S1_Invalid_ReturnNull.c	007	1	ReturnNull
008_S1_Invalid_ReturnNoFuncName	Compiler	Validate an int return main function with no function name.	1. Run compiler with .c test data name as input parameter. 2. Verify compiler output. 3. As the test has an invalid input file, no assembly file nor executable should generate.	Elixir environment ready and .c file loaded into test directory. Target assembly code ready to compare.	008_S1_Invalid_ReturnNoFuncName.c	008	1	ReturnNoFuncName
009_S1_Invalid_ReturnNoParenth	Compiler	Validate an int return main function with a missing parenthesis.	1. Run compiler with .c test data name as input parameter. 2. Verify compiler output. 3. As the test has an invalid input file, no assembly file nor executable should generate.	Elixir environment ready and .c file loaded into test directory. Target assembly code ready to compare.	009_S1_Invalid_ReturnNoParenth.c	009	1	ReturnNoParenth

Test Case ID (testNumber_s#_ type)	Module	Description	Steps	Prerequisites	Test Data	Test Number	Stage Number	Shortname
010_S1_Invalid_ReturnNoBrack	Compiler	Validate an int return main function with a missing bracket.	1. Run compiler with .c test data name as input parameter. 2. Verify compiler output. 3. As the test has an invalid input file, no assembly file nor executable should generate.	Elixir environment ready and .c file loaded into test directory. Target assembly code ready to compare.	010_S1_Invalid_ReturnNoBrack.c	010	1	ReturnNoBrack
011_S1_Invalid_ReturnNoSpaces	Compiler	Validate an int return main function with no space between the function type and name.	1. Run compiler with .c test data name as input parameter. 2. Verify compiler output. 3. As the test has an invalid input file, no assembly file nor executable should generate.	Elixir environment ready and .c file loaded into test directory. Target assembly code ready to compare.	011_S1_Invalid_ReturnNoSpaces.c	011	1	ReturnNoSpaces
012_S1_Invalid_ReturnComma	Compiler	Validate an int return main function with a comma instead of semicolon after return statement.	1. Run compiler with .c test data name as input parameter. 2. Verify compiler output. 3. As the test has an invalid input file, no assembly file nor executable should generate.	Elixir environment ready and .c file loaded into test directory. Target assembly code ready to compare.	012_S1_Invalid_ReturnComma.c	012	1	ReturnComma
013_S1_Invalid_ReturnCaps	Compiler	Validate an int return main function with different caps format for statements on the function type and return statement.	1. Run compiler with .c test data name as input parameter. 2. Verify compiler output. 3. As the test has an invalid input file, no assembly file nor executable should generate.	Elixir environment ready and .c file loaded into test directory. Target assembly code ready to compare.	013_S1_Invalid_ReturnCaps.c	013	1	ReturnCaps
014_S1_Valid_ReturnPrecZero	Compiler	Validate an int return main function with a return value preceded by zeros.	1. Run compiler with .c test data name as input parameter. 2. Verify compiler output.	Elixir environment ready and .c file loaded into test directory. Target assembly code ready to compare.	014_S1_Valid_ReturnPrecZero.c	014	1	ReturnPrecZero
015_S2_Valid_Negative	Compiler	Validate an int return main function with a negated int value of any decimal number.	1. Run compiler with .c test data name as input parameter. 2. Verify compiler output against valid assembly code for the .c input.	Elixir environment ready and .c file loaded into test directory. Target assembly code ready to compare.	015_S2_Valid_Negative.c	015	2	Negative
016_S2_Valid_Bitwise	Compiler	Validate the compilation of the bitwise (~) operator with a decimal number.	1. Run compiler with .c test data name as input parameter. 2. Verify compiler output against valid assembly code for the .c input.	Elixir environment ready and .c file loaded into test directory. Target assembly code ready to compare.	016_S2_Valid_Bitwise.c	016	2	Bitwise
017_S2_Valid_Bitwise_0	Compiler	Validate the compilation of the bitwise (~) operator on the number zero.	1. Run compiler with .c test data name as input parameter. 2. Verify compiler output against valid assembly code for the .c input.	Elixir environment ready and .c file loaded into test directory. Target assembly code ready to compare.	017_S2_Valid_Bitwise_0.c	017	2	Bitwise_0
018_S2_Valid_Not_7	Compiler	Validate the compilation of the logical NOT operator applied to the number seven.	1. Run compiler with .c test data name as input parameter. 2. Verify compiler output against valid assembly code for the .c input.	Elixir environment ready and .c file loaded into test directory. Target assembly code ready to compare.	018_S2_Valid_Not_7.c	018	2	Not_7

Test Case ID (testNumber_s#_ type)	Module	Description	Steps	Prerequisites	Test Data	Test Number	Stage Number	Shortname
019_S2_Valid_Not_0	Compiler	Validate the compilation of the logical NOT operator on the number zero.	1. Run compiler with .c test data name as input parameter. 2. Verify compiler output against valid assembly code for the .c input.	Elixir environment ready and .c file loaded into test directory. Target assembly code ready to compare.	019_S2_Valid_Not_0.c	019	2	Not_0
020_S2_Valid_Multiple_Ops_1	Compiler	Validate the compilation of the negative and bitwise operator used on the number 7.	1. Run compiler with .c test data name as input parameter. 2. Verify compiler output against valid assembly code for the .c input.	Elixir environment ready and .c file loaded into test directory. Target assembly code ready to compare.	020_S2_Valid_Multiple_Ops_1.c	020	2	Multiple_Ops_1
021_S2_Valid_Multiple_Ops_2	Compiler	Validate the compilation of the NOT operator and negative operator used on the number 4.	1. Run compiler with .c test data name as input parameter. 2. Verify compiler output against valid assembly code for the .c input.	Elixir environment ready and .c file loaded into test directory. Target assembly code ready to compare.	021_S2_Valid_Multiple_Ops_2.c	021	2	Multiple_Ops_2
022_S2_Valid_Multiple_Ops_3	Compiler	Validate the compilation of the NOT operator and bitwise operator used on the number 0.	1. Run compiler with .c test data name as input parameter. 2. Verify compiler output against valid assembly code for the .c input.	Elixir environment ready and .c file loaded into test directory. Target assembly code ready to compare.	022_S2_Valid_Multiple_Ops_3.c	022	2	Multiple_Ops_3
023_S2_Invalid_Wrong_Order_Negative	Compiler	Refute the compilation of a main function using the negative operator on an incorrect order <- first number 7 and then the operator.	1. Run compiler with .c test data name as input parameter. 2. Verify that the compiler shows an error on run console. 3. No output assembly file should generate.	Elixir environment ready and .c file loaded into test directory.	023_S2_Invalid_Wrong_Order_Negative.c	023	2	Wrong_Order_Negative
024_S2_Invalid_Correct_Neg_Wrong_Bitwise_Order	Compiler	Refute the compilation of a main function using the negative operator on the correct order with the bitwise operator after the number <- first negative operator, then number 5 and then the bitwise operator.	1. Run compiler with .c test data name as input parameter. 2. Verify that the compiler shows an error on run console. 3. No output assembly file should generate.	Elixir environment ready and .c file loaded into test directory.	024_S2_Invalid_Correct_Neg_Wrong_Bitwise_Order.c	024	2	Correct_Neg_Wrong_Bitwise_Order
025_S2_Invalid_Bitwise_No_Semicolon	Compiler	Refute the compilation of a main function using the bitwise operator on the number zero with a missing semicolon to end statement.	1. Run compiler with .c test data name as input parameter. 2. Verify that the compiler shows an error on run console. 3. No output assembly file should generate.	Elixir environment ready and .c file loaded into test directory.	025_S2_Invalid_Bitwise_No_Semicolon.c	025	2	Bitwise_No_Semicolon
026_S2_Invalid_Not_Missing_Const	Compiler	Refute the compilation of a main function that has a missing constant on the return statement.	1. Run compiler with .c test data name as input parameter. 2. Verify that the compiler shows an error on run console. 3. No output assembly file should generate.	Elixir environment ready and .c file loaded into test directory.	026_S2_Invalid_Not_Missing_Const.c	026	2	Not_Missing_Const

Test Case ID (testNumber_s#_ type)	Module	Description	Steps	Prerequisites	Test Data	Test Number	Stage Number	Shortname
027_S2_Invalid_Not_Bitwise_Const	Compiler	Refute the compilation of a main function that has a missing constant on a return statement that has a NOT and bitwise operators.	1. Run compiler with .c test data name as input parameter. 2. Verify that the compiler shows an error on run console. 3. No output assembly file should generate.	Elixir environment ready and .c file loaded into test directory.	027_S2_Invalid_Not_Bitwise_Const.c	027	2	Not_Bitwise_Const
028_S3_Valid_Add	Compiler	Validate the compilation of the add operator of two integers on a main function with int return.	1. Run compiler with .c test data name as input parameter. 2. Verify that the compiler output corresponds to valid binary code generated on gcc or clang compilers.	Elixir environment ready and .c file loaded into test directory.	028_S3_Valid_Add.c	028	3	Add
029_S3_Valid_SubstractPositive	Compiler	Validate the compilation of the subtract operator of two positive integers on a main function with int return.	1. Run compiler with .c test data name as input parameter. 2. Verify that the compiler output corresponds to valid binary code generated on gcc or clang compilers.	Elixir environment ready and .c file loaded into test directory.	029_S3_Valid_SubstractPositive.c	029	3	SubstractPositive
030_S3_Valid_SubstractNegative	Compiler	Validate the compilation of the subtract operator of a positive and a negative integer on a main function with int return.	1. Run compiler with .c test data name as input parameter. 2. Verify that the compiler output corresponds to valid binary code generated on gcc or clang compilers.	Elixir environment ready and .c file loaded into test directory.	030_S3_Valid_SubstractNegative.c	030	3	SubstractNegative
031_S3_Valid_DivPositive	Compiler	Validate the compilation of the div operator of two positive integers on a main function with int return.	1. Run compiler with .c test data name as input parameter. 2. Verify that the compiler output corresponds to valid binary code generated on gcc or clang compilers.	Elixir environment ready and .c file loaded into test directory.	031_S3_Valid_DivPositive.c	031	3	DivPositive
032_S3_Valid_DivNegative	Compiler	Validate the compilation of the div operator of two negative integers on a main function with int return.	1. Run compiler with .c test data name as input parameter. 2. Verify that the compiler output corresponds to valid binary code generated on gcc or clang compilers.	Elixir environment ready and .c file loaded into test directory.	032_S3_Valid_DivNegative.c	032	3	DivNegative
033_S3_Valid_MultPositive	Compiler	Validate the compilation of the multiplication (*) operator of two integers on a main function with int return.	1. Run compiler with .c test data name as input parameter. 2. Verify that the compiler output corresponds to valid binary code generated on gcc or clang compilers.	Elixir environment ready and .c file loaded into test directory.	033_S3_Valid_MultPositive.c	033	3	MultPositive
034_S3_Valid_MultNeg	Compiler	Validate the compilation of the multiplication (*) operator of two integers, one positive and one negative, on a main function with int return.	1. Run compiler with .c test data name as input parameter. 2. Verify that the compiler output corresponds to valid binary code generated on gcc or clang compilers.	Elixir environment ready and .c file loaded into test directory.	034_S3_Valid_MultNeg.c	034	3	MultNeg

Test Case ID (testNumber_s#_ type)	Module	Description	Steps	Prerequisites	Test Data	Test Number	Stage Number	Shortname
035_S3_Valid_Parenthesis	Compiler	Validate that the use of parenthesis maintains the precedence of the operations.	1. Run compiler with .c test data name as input parameter. 2. Verify that the compiler output corresponds to valid binary code generated on gcc or clang compilers.	Elixir environment ready and .c file loaded into test directory.	035_S3_Valid_Parenthesis.c	035	3	Parenthesis
036_S3_Valid_SimpleParenthesis	Compiler	Validate that the use of parenthesis maintains the precedence of the operations.	1. Run compiler with .c test data name as input parameter. 2. Verify that the compiler output corresponds to valid binary code generated on gcc or clang compilers.	Elixir environment ready and .c file loaded into test directory.	036_S3_Valid_SimpleParenthesis.c	036	3	SimpleParenthesis
037_S3_Valid_Precedence	Compiler	Validate that precedence is correctly followed when using operators with no parenthesis. Program is a main function with an int return.	1. Run compiler with .c test data name as input parameter. 2. Verify that the compiler output corresponds to valid binary code generated on gcc or clang compilers.	Elixir environment ready and .c file loaded into test directory.	037_S3_Valid_Precedence.c	037	3	Precedence
038_S3_Valid_Bitwise_NoParenthesis	Compiler	Validate that precedence is correctly followed when using operators with no parenthesis when using the bitwise operator with a subtract operation. Program is a main function with an int return.	1. Run compiler with .c test data name as input parameter. 2. Verify that the compiler output corresponds to valid binary code generated on gcc or clang compilers.	Elixir environment ready and .c file loaded into test directory.	038_S3_Valid_Bitwise_NoParenthesis.c	038	3	Bitwise_NoParenthesis
039_S3_Valid_BItwise_Parenthesis	Compiler	Validate that precedence is correctly followed when using operators with a parenthesis when using the bitwise operator with a subtract operation. Program is a main function with an int return.	1. Run compiler with .c test data name as input parameter. 2. Verify that the compiler output corresponds to valid binary code generated on gcc or clang compilers.	Elixir environment ready and .c file loaded into test directory.	039_S3_Valid_BItwise_Parenthesis.c	039	3	BItwise_Parenthesis
040_S3_Valid_Multiple_Parenthesis	Compiler	Validate the use of multiple parenthesis with a variety of operators.	1. Run compiler with .c test data name as input parameter. 2. Verify that the compiler output corresponds to valid binary code generated on gcc or clang compilers.	Elixir environment ready and .c file loaded into test directory.	040_S3_Valid_Multiple_Parenthesis.c	040	3	Multiple_Parenthesis
041_S3_Invalid_Div_Missing_Operator	Compiler	Refute the compilation of a main function using the the div operator with a missing element.	1. Run compiler with .c test data name as input parameter. 2. Verify that the compiler shows an error on run console. 3. No output assembly file should generate.	Elixir environment ready and .c file loaded into test directory.	041_S3_Invalid_Div_Missing_Operator.c	041	3	Div_Missing_Operator
042_S3_Invalid_Sum_Missing_Operator	Compiler	Refute the compilation of a main function using the sum operator with a missing operator.	1. Run compiler with .c test data name as input parameter. 2. Verify that the compiler shows an error on run console. 3. No output assembly file should generate.	Elixir environment ready and .c file loaded into test directory.	042_S3_Invalid_Sum_Missing_Operator.c	042	3	Sum_Missing_Operator

Test Case ID (testNumber_s#_ type)	Module	Description	Steps	Prerequisites	Test Data	Test Number	Stage Number	Shortname
043_S3_Invalid_Parenthesis_Middle_Operator	Compiler	Refute the compilation of a main function missing an operator between close parenthesis and another element.	1. Run compiler with .c test data name as input parameter. 2. Verify that the compiler shows an error on run console. 3. No output assembly file should generate.	Elixir environment ready and .c file loaded into test directory.	043_S3_Invalid_Parenthesis_Middle_Operator.c	043	3	Parenthesis_Middle_Operator
044_S3_Invalid_Neg_Missing_Operator	Compiler	Refute the compilation of a main function using the negative operator with a missing element.	1. Run compiler with .c test data name as input parameter. 2. Verify that the compiler shows an error on run console. 3. No output assembly file should generate.	Elixir environment ready and .c file loaded into test directory.	044_S3_Invalid_Neg_Missing_Operator.c	044	3	Neg_Missing_Operator
045_S3_Invalid_Missing_Parenthesis	Compiler	Refute the compilation of a main function with a missing close parenthesis.	1. Run compiler with .c test data name as input parameter. 2. Verify that the compiler shows an error on run console. 3. No output assembly file should generate.	Elixir environment ready and .c file loaded into test directory.	045_S3_Invalid_Missing_Parenthesis.c	045	3	Missing_Parenthesis

Test Case ID (testNumber_s#_ type)	**Expected Result**	**Tester**
001_S1_Valid_Return0	Valid	bondi7
002_S1_Valid_Return7	Valid	bondi7
003_S1_Valid_ReturnMD130	Valid	bondi7
004_S1_Valid_ReturnBlankSpaces	Valid	bondi7
005_S1_Valid_ReturnNoLineB	Valid	bondi7
006_S1_Valid_ReturnSpaceChars	Valid	bondi7
007_S1_Invalid_ReturnNull	Invalid	bondi7
008_S1_Invalid_ReturnNoFuncName	Invalid	bondi7
009_S1_Invalid_ReturnNoParenth	Invalid	bondi7

Test Case ID (testNumber_s#_ type)	Expected Result	Tester
010_S1_Invalid_ReturnNoBrack	Invalid	bondi7
011_S1_Invalid_ReturnNoSpaces	Invalid	bondi7
012_S1_Invalid_ReturnComma	Invalid	bondi7
013_S1_Invalid_ReturnCaps	Invalid	bondi7
014_S1_Valid_ReturnPrecZero	Valid	bondi7
015_S2_Valid_Negative	Valid	bondi7
016_S2_Valid_Bitwise	Valid	bondi7
017_S2_Valid_Bitwise_0	Valid	bondi7
018_S2_Valid_Not_7	Valid	bondi7

Test Case ID (testNumber_s#_ type)	Expected Result	Tester
019_S2_Valid_Not_0	Valid	bondi7
020_S2_Valid_Multiple_Ops_1	Valid	bondi7
021_S2_Valid_Multiple_Ops_2	Valid	bondi7
022_S2_Valid_Multiple_Ops_3	Valid	bondi7
023_S2_Invalid_Wrong_Order_Negative	Invalid	bondi7
024_S2_Invalid_Correct_Neg_Wrong_Bitwise_Order	Invalid	bondi7
025_S2_Invalid_Bitwise_No_Semicolon	Invalid	bondi7
026_S2_Invalid_Not_Missing_Const	Invalid	bondi7

Test Case ID (testNumber_s#_ type)	Expected Result	Tester
027_S2_Invalid_Not_Bitwise_Const	Invalid	bondi7
028_S3_Valid_Add	Valid	bondi7
029_S3_Valid_SubstractPositive	Valid	bondi7
030_S3_Valid_SubstractNegative	Valid	bondi7
031_S3_Valid_DivPositive	Valid	bondi7
032_S3_Valid_DivNegative	Valid	bondi7
033_S3_Valid_MultPositive	Valid	bondi7
034_S3_Valid_MultNeg	Valid	bondi7

Test Case ID (testNumber_s#_ type)	Expected Result	Tester
035_S3_Valid_Parenthesis	Valid	bondi7
036_S3_Valid_SimpleParenthesis	Valid	bondi7
037_S3_Valid_Precedence	Valid	bondi7
038_S3_Valid_Bitwise_NoParenthesis	Valid	bondi7
039_S3_Valid_BItwise_Parenthesis	Valid	bondi7
040_S3_Valid_Multiple_Parenthesis	Valid	bondi7
041_S3_Invalid_Div_Missing_Operator	Invalid	bondi7
042_S3_Invalid_Sum_Missing_Operator	Invalid	bondi7

Test Case ID (testNumber_s#_ type)	Expected Result	Tester
043_S3_Invalid_Parenthesis_Middle_Operator	Invalid	bondi7
044_S3_Invalid_Neg_Missing_Operator	Invalid	bondi7
045_S3_Invalid_Missing_Parenthesis	Invalid	bondi7

Tabla 1
