**PlusEqualAndMinusEqual1**

* Responsible for testing -= and += with types integers, floats, strings (integers and floats format). Each test section is applied to one input with alternating -= and += to arrive at a satisfied answer.

**PlusEqualAndMinus2**

* Does additional testing with variables of type Float and Int as well as arrays. Demonstrates coercion of string types to either Int or Float and performs -= += operations on them. It applies these to arrays as well.

**BuiltInFunctionsTests**

* Responsible for testing all built in functions including LENGTH, SPACES, ELEM, MAXELEM, IN, and NOTIN.
  + First section of tests are String functions LENGTH and SPACES. They test types String, Int, Bool, and Float. All can be coerced to strings and operated on.
  + Second section of tests are array functions ELEM and MAXELEM. They test String, Float, and Integer arrays. Arrays are a mix of fixed and unbounded types.
  + Third section runs tests on program 4 functions test file provided by Clark.
  + Fourth section of tests are dedicated to IN and NOTIN functions. It test types String, Float, Int, and Bool. It runs these tests against Fixed and Unbounded arrays of all types. It runs these tests against in place list that are defined in the print statements.

**DateTests**

* Responsible for all date functions including dateAdj, dateDiff, and dateAge. These accept string and date types.
  + dateAdj
    - Test that dates can be adjusted forward
    - Test that dates can be adjusted backward
    - Test that dates that are adjusted by one year consider leap years and adjust appropriately.
    - Test that dates can be adjusted by a primitive number and a variable number that is either an integer, string, or float.
  + dateAge
    - Test that dateAge can accept a primitive or a variable.
  + dateDiff
    - Test that dateDiff can accept a primitive, a variable, or a function.
    - Test that all functions can be combined in some way.
    - Test that the output of the functions can be coerced into something like a different valid date entirely.

**LoopTests**

* Responsible for testing all kinds of loops in the language including while and for. It additional test condition statements including if and else. This includes the testing of logical operators as well.
  + First section is testing a while loop that only exits on a Bool flag. It incorporates conditions.
  + Second section is testing the sorting of an array of integers. It features a double for loop with a conditional for element index comparisons.
  + Third section is testing the printing of all elements in a fixed array
  + Fourth section is testing the printing of all elements in an unbound array
  + Fifth section is testing a while loop that loops through a fixed element array printing each element out and then printing an indicator when it reaches half way to the maximum value of the fixed array.
  + Sixth section is taking an array of strings, reversing it, and then storing it in another string.
  + Seventh section is taking an array of strings, reversing every element in the array, reversing the entire array with the reversed elements, and storing the result in another string.
  + Section 8 finds all pairs of elements in a fixed array whose sum is equal to a given number
  + Section 9 finds roman numeral equivalents of a decimal number.

**TokenizingFor**

* Responsible for testing the tokenizing of a string inside of a for loop.
  + Section 1 is tokenizing a sentence word by word using spaces as a delimiter.
  + Section 2 is tokenizing a URL by the delimiter ‘%20’ which is a URL space.
  + Section 3 is not tokenizing anything because it does not have a delimiter that matches the string it is operating on.
  + Section 4 is demonstrating that changing the string mid loop does not affect the original loop’s iteration.
  + Section 5 is demonstrating the changing the delimiter mid loop does not after the original loop’s iteration.
  + Section 6 is demonstrating that having a delimiter that is empty tokenizes a string character by character.
  + Section 7 is demonstrating that having an empty delimiter for an empty string will not even attempt to enter the loop.
  + Section 8 is demonstrating tokenizing a string out of a string that is being tokenized to get at a specific format of data using two loops.

**P5FixedArrayTest01.txt**

* Test the initializing and looping of fixed arrays. Mainly for Float and Int arrays.

**P5FixedArrayTest02.txt**

* Test the initializing and looping of fixed arrays. Mainly for String arrays.

**P5UnboundedArrayTest.txt**

* Test Scalar assignment to unbound array.
* Test incrementing through an unbounded array by a certain increment value.
* Test unbound array to fixed array assignment.
* Test fixed array to unbound array assignment.

**P5Coercion\_valid.txt**

* Test all of the coercion rules that can be found in Section 4.1.

**BuiltinForLoop**

* Section 1 is testing that basic counting is happening.
* Section 2 is testing that counting with an increment is happening.
* Section 3 is testing that when the control variable changes, it affects the number of loops.
* Section 4 is testing that changing the source value express does not affect the number of loops.
* Section 5 is testing that when the limit value expression changes it does not affect the number of loops.
* Section 6 is testing that when the increment value expression changes it does not affect the number of loops.
* Section 7 is demonstrating the looping of a string.
* Section 8 is testing that changing the character control variable for a string does not affect the number of loops.
* Section 9 is testing that changing he string that the loop iterates over is not affecting the number of loops.