|  |
| --- |
| Deakin University |
| SIT323 ASESSMENT 1 – VALIDATION AND TESTING (old files) |
| By Connor Vernon |
| Partner: Richardo Wijaya |
| **ID: 216150769** |
| **8/26/2018** |

|  |
| --- |
| This document contains the tables of unit tests on a Crozzle Application. It shows how a function will behave depending on what parameters are passed and what changes are made to the original Crozzle files. Tests that have passed mean that the function works correctly to validate or return a correct value. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Scenario ID** | **Test Description** | | | | |
| 1 | Check that the strings are correctly interpreted as a Boolean value. If the expected and actual return values are true, the method works correctly to identify Boolean values. Expected and actual false returns mean that it can identify non-Boolean. | | | | |
| **Test Method** | **Method Tested** | | | | |
| UnitTest1.TestMethod1() | Boolean Validator.isBoolean(String field, out Boolean aBoolean) | | | | |
| **Test Case ID** | **Parameters** | **Expected Data** | **Actual Data** | **Test Result** | **Test Comments** |
| 1.1 | field = “false” | expectedReturn1 = true | actualReturn1 = true | passed | “false” is a proper Boolean variable |
| 1.2 | field =”falsee” | expectedReturn2 = false | ActualReturn2 = false | passed | Program returns false due to invalid Boolean value |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Scenario ID** | **Test Description** | | | | |
| 2 | Check that the strings are correctly interpreted as integer values. If the expected and actual return values are true, the method works correctly to identify Integer values. Expected and actual false returns mean that it can identify non-Integers. | | | | |
| **Test Method** | **Method Tested** | | | | |
| UnitTest1.TestMethod2() | Boolean Validator.isInt32(String field, out int anInteger) | | | | |
| **Test Case ID** | **Parameters** | **Expected Data** | **Actual Data** | **Test Result** | **Test Comments** |
| 2.1 | field = “25” | expectedReturn1 = true | actualReturn1=true | Passed | True as the string value can be interpreted as integer |
| 2.2 | field = “25.32” | expectedReturn2 = false | actualReturn2=false | Passed | Found that floating point numbers cannot be read |
| 2.3 | field = “-25” | expectedReturn3 = true | actualReturn3=true | Passed | True as the string value can be interpreted as integer |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Scenario ID** | **Test Description** | | | | |
| 3 | Check that the strings are correctly interpreted as colour hex values. If the expected and actual return values are true, the method works correctly to identify Hex values. Expected and actual false returns mean that it can identify illegal characters | | | | |
| **Test Method** | **Method Tested** | | | | |
| UnitTest1.TestMethod3() | Boolean Validator.isHexColourCode(String hexColour) | | | | |
| **Test Case ID** | **Parameters** | **Expected Data** | **Actual Data** | **Test Result** | **Test Comments** |
| 3.1 | hexColour = “#2ecc71” | expectedReturn1 = true | actualReturn1 = true | Passed | The colour Green. Passed as it met the requirements of REGEX |
| 3.2 | hexColour = “##ecc71” | expectedReturn2 = false | actualReturn2 = false | Passed | False due to illegal symbol after the initial hash tag |
| 3.3 | hexColour = “#2ecc7g” | expectedReturn3 = false | actualReturn3 = false | Passed | False due to a character being out of range |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Scenario ID** | **Test Description** | | | | |
| 4 | Checking that the string value passed matches a key-value pair. . If the expected and actual return values are true, the method works correctly to identify keyvalue pairs and matches the key Pattern specified. Expected and Actual values that equal false mean that the function can correctly identify an invalid key value pair. | | | | |
| **Test Method** | **Method Tested** | | | | |
| UnitTest1.TestMethod4() | Boolean KeyValue.TryParse(String originalKeyValueData, String keyPattern, out KeyValue aKeyValue) | | | | |
| **Test Case ID** | **Parameters** | **Expected Data** | **Actual Data** | **Test Result** | **Test Comments** |
| 4.1 | originalKeyValueData = "MAXIMUM = 100" KeyPattern = @"[a-zA-Z0-9]" | expectedReturn1 = true | actualReturn1 = true | Passed | True because String passed has a equals sign |
| 4.2 | originalKeyValueData = "MAXIMUM="  KeyPattern = @"[a-zA-Z0-9]" | expectedReturn2 = false | actualReturn2 = false | Passed | False due to no value given for one of the pairs |
| 4.3 | originalKeyValueData = "MAXIMUM= "  KeyPattern = @"[a-zA-Z0-9]" | expectedReturn3 = true | actualReturn3 = true | Passed | Whitespace added after “=”. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Scenario ID** | **Test Description** | | | | |
| 5 | Making sure that the expected score matches the actual score generated when the crozzle application is validated. If the expected matches the actual value, the method can successfully generate the correct score. | | | | |
| **Test Method** | **Method Tested** | | | | |
| UnitTest1.TestMethod5() | String Crozzle.Score() | | | | |
| **Test Case ID** | **Parameters** | **Expected Data** | **Actual Data** | **Test Result** | **Test Comments** |
| 5.1 | n/a | expectedTrueScore = “303” | Crozzle.Score = “303” | Passed | True because the expected score matches actual score from the validation |
| 5.2 | n/a | expectedInvalidScore = “INVALID CROZZLE” | Crozzle2.Score = “INVALID CROZZLE” | Passed | A row has been removed from the first crozzle text file making the score invalid |
| 5.3 | n/a | expectedNewWordsScore = "336"; | Crozzle3.Score(); | Passed | Two new words added hence there is a higher score of 336 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Scenario ID** | **Test Description** | | | | |
| 6 | Checking that the Crozzle application can successfully find duplicates within the Crozzle. If errors are detected, the application method can successfully log errors if the duplicate word count exceeds the max limit specified in the config files. | | | | |
| **Test Method** | **Method Tested** | | | | |
| UnitTest1.TestMethod6() | void Crozzle.CheckDuplicateWords(int lowerLimit, int upperLimit) | | | | |
| **Test Case ID** | **Parameters** | **Expected Data** | **Actual Data** | **Test Result** | **Test Comments** |
| 6.1 | lowerLimit = 1  upperLimit = 1 | expectedFalseErrors = false | crozzleSequences.ErrorsDetected = false | Passed | No errors can be detected in the original files |
| 6.2 | lowerLimit = 1  upperLimit = 2 | expectedFalseErrorsNew = false | crozzleSequences2.ErrorsDetected = false | Passed | No errors as two duplicates allowed in new file |
| 6.3 | lowerLimit = 1  upperLimit = 1 | expectedTrueErrorsNew = true | crozzleSequences3.ErrorsDetected = true | Passed | Errors returned as duplicates exceed max duplicates |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Scenario ID** | **Test Description** | | | | |
| 7 | Confirming that the Crozzle Application can determine that the Crozzle displayed is valid and checking that the correct error message is returned if there are any errors present. | | | | |
| **Test Method** | **Method Tested** | | | | |
| UnitTest1.TestMethod7() | void Crozzle.Validate() | | | | |
| **Test Case ID** | **Parameters** | **Expected Data** | **Actual Data** | **Test Result** | **Test Comments** |
| 7.1 | n/a | expectedTrueValidityOriginal = true | crozzle.CrozzleValid = true | Passed | True as Crozzle files meet all conditions |
| 7.2 | n/a | expectedFalseValidityNew = false | crozzle2.CrozzleValid = false | Passed | False value as validate method detected errors |
| 7.3 | n/a | expectedErrorMessageNew = "code 11002: the number of horizontal words intersecting AL is 0, but this is not within [1, 100]" | crozzle2.CrozzleGridErrors[0] = “code 11002: the number of horizontal words intersecting AL is 0, but this is not within [1, 100]” | Passed | Correct error message matches that of the error in the program |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Scenario ID** | **Test Description** | | | | |
| 8 | Confirming that the method returns the correct string value. The string value is dependent on different settings in each file. If the expected value matches the actual value, the application returns the correct results based off changes such as different scores. | | | | |
| **Test Method** | **Method Tested** | | | | |
| UnitTest1.TestMethod8() | String Crozzle.ToStringHTML() | | | | |
| **Test Case ID** | **Parameters** | **Expected Data** | **Actual Data** | **Test Result** | **Test Comments** |
| 8.1 | n/a | expectedAfterValidate = Refer to appendix 1.A | crozzle.ToStringHTML() = Refer to appendix 1.A | Passed | Crozzle file is valid when Validate method called before saving and receiving HTML |
| 8.2 | n/a | expectedBeforeValidate =  Refer to appendix 1.B | crozzle.ToStringHTML() = Refer to appendix 1.B | Passed | Crozzle file invalid when Validate method called before saving and receiving HTML |
| 8.3 | n/a | expectedNewHTML = Refer to appendix 1.C | crozzle.ToStringHTML() = Refer to appendix 1.C | Passed | Correct output from new files which has a different score of 336 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Scenario ID** | **Test Description** | | | | |
| 9 | Checking that the crozzle application can successfully count the number of groups of connected words in the application. The test is conducted on different changes to the crozzle files. If the expected matches the actual value, the application can successfully identify the number of groups of connected words. | | | | |
| **Test Method** | **Method Tested** | | | | |
| UnitTest1.TestMethod9() | int CrozzleMap.GroupCount() | | | | |
| **Test Case ID** | **Parameters** | **Expected Data** | **Actual Data** | **Test Result** | **Test Comments** |
| 9.1 | n/a | expectedGroupCount = 1 | crozzleMap.GroupCount() = 1 | Passed | All words are connected on crozzle hence 1 group |
| 9.2 | n/a | expectedNewGroupCount1 = 2 | crozzleMap2.GroupCount() = 2 | Passed | Crozzle has two separate groups hence group count 2 |
| 9.3 | n/a | expectedNewGroupCount2 = 4 | crozzleMap3.GroupCount() = 4 | Passed | Crozzle split into 4 different groups hence group count 4 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Scenario ID** | **Test Description** | | | | |
| 10 | Check that values of Configuration properties are the same as the values within the configuration file as well as testing different value types such as integer and Boolean. If these expected values match the actual values, the method can read parts of the config file. Additionally, testing is done if the configuration file is returned as true. This means that the program can correctly open the config file. | | | | |
| **Test Method** | **Method Tested** | | | | |
| UnitTest1.TestMethod10() | Boolean Configuration.TryParse(path, out Configuration aConfiguration); | | | | |
| **Test Case ID** | **Parameters** | **Expected Data** | **Actual Data** | **Test Result** | **Test Comments** |
| 10.1 | path = “Test 1 Configuration.txt" | expectedFileReturn = True | actualFileReturn = true; | Passed | TryParse can successfully find and open the config file |
| 10.2 | path = “Test 1 Configuration.txt" | expectedMaxUniqueWords = 1000 | aConfiguration. MaximumNumberOfUniqueWords= 1000 | Passed | Expected value matches the value in config text file |
| 10.3 | path = “Test 1 Configuration.txt" | expectedTrueUppercase = true | aConfiguration.Uppercase = true | Passed | Tested to see that program can detect uppercase in file. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Scenario ID** | **Test Description** | | | | |
| 11 | Check that values of Wordlist properties are the same as the values within the wordlist file and that the correct number of words can be obtained from the wordlist file. If these expected values match the actual values, it means the method can correctly identify properties in the wordlist file. Additionally, a test is run to check that the current wordlist file is valid. If the return value is true, it means that the method can identify a valid file if the conditions are met. | | | | |
| **Test Method** | **Method Tested** | | | | |
| UnitTest1.TestMethod11() | Boolean Wordlist.TryParse(String path, Configuration aConfiguration, out WordList aWordList) | | | | |
| **Test Case ID** | **Parameters** | **Expected Data** | **Actual Data** | **Test Result** | **Test Comments** |
| 11.1 | path = “Test 1 Wordlist.txt"  aConfiguration = {CrozzleApplication.Configuration} | expectedContainedWord = true; | aWordList.Contains(“BETTY”) = true | Passed | Wordlist file contains the string “BETTY”. |
| 11.2 | path = “Test 1 Wordlist.txt "  aConfiguration = {CrozzleApplication.Configuration} | expectedFieldNumber = 31 | aWordlist.List.Count = 31 | Passed | Counting the total words in the wordlist file |
| 11.3 | path = “Test 1 Wordlist.txt "  aConfiguration = {CrozzleApplication.Configuration} | expectedValid = true; | aWordlist.Valid = true | Passed | Wordlist file does not break conditions hence valid. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Scenario ID** | **Test Description** | | | | |
| 12 | Check that values of Crozzle properties are the same as the values within the Crozzle file. If these expected values match the actual values, it means the method can correctly identify properties in the crozzle file. Additionally, testing is done if the expected wordlist path matches that of the actual wordlist path that is passed into the TryParse of crozzle. If it matches, then the method can correctly locate and open files. | | | | |
| **Test Method** | **Method Tested** | | | | |
| UnitTest1.TestMethod12() | Boolean Crozzle.TryParse(String path, Configuration aConfiguration, WordList wordList, out Crozzle aCrozzle | | | | |
| **Test Case ID** | **Parameters** | **Expected Data** | **Actual Data** | **Test Result** | **Test Comments** |
| 12.1 | path = “Test 1 Crozzle.txt"  aConfiguration = {CrozzleApplication.Configuration}  wordlist = {CrozzleApplication.WordList} | actualReturnRows = 10 | crozzle.Rows = 10 | Passed | Application can read rows from crozzle file |
| 12.2 | path = “Test 1 Crozzle.txt"  aConfiguration = {CrozzleApplication.Configuration}  wordlist = {CrozzleApplication.WordList}= null | actualReturnColumns = 15 | crozzle.Columns = 15 | Passed | Application can read columns from crozzle file |
| 12.3 | path = “Test 1 Crozzle.txt"  aConfiguration = {CrozzleApplication.Configuration}  wordlist = {CrozzleApplication.WordList} | expectedWordListPath = “Test 1 Wordlist.txt” | Crozzle.WordListPath = “Test 1 Wordlist.txt” | Passed | Can successfully read the wordlist text file from the crozzle text file. |

# Appendices

## Appendix 1A

The expected result of the toStringHTML() function is generated by using StreamWriter to save the return result to a text file called “html.txt” which is located in the folder of “unitTest8”. StreamReader was then used to read the expected string from the text file. The HTML below shows the expected and actual result of the test. The highlighted text shows the validity of each file as well as the score generated from the Validate() method.

|  |
| --- |
| <!DOCTYPE html><html><head><style> table, td { border: 1px solid black; border-collapse: collapse; } td { width:24px; height:18px; text-align: center; } </style><style>  .empty { background-color: #777777; }  .nonempty { background-color: #ffffff; }  </style></head><body><table><tr><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="nonempty">B</td><td class="nonempty">E</td><td class="nonempty">T</td><td class="nonempty">T</td><td class="nonempty">Y</td><td class="empty"> </td><td class="nonempty">O</td><td class="nonempty">S</td><td class="nonempty">C</td><td class="nonempty">A</td><td class="nonempty">R</td><td class="empty"> </td></tr><tr><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="nonempty">I</td><td class="empty"> </td><td class="empty"> </td><td class="nonempty">O</td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="nonempty">H</td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td></tr><tr><td class="empty"> </td><td class="nonempty">J</td><td class="nonempty">I</td><td class="nonempty">L</td><td class="nonempty">L</td><td class="empty"> </td><td class="nonempty">M</td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="nonempty">A</td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td></tr><tr><td class="empty"> </td><td class="nonempty">E</td><td class="empty"> </td><td class="nonempty">L</td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="nonempty">H</td><td class="empty"> </td><td class="empty"> </td><td class="nonempty">R</td><td class="empty"> </td><td class="nonempty">M</td><td class="empty"> </td></tr><tr><td class="empty"> </td><td class="nonempty">S</td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="nonempty">A</td><td class="empty"> </td><td class="empty"> </td><td class="nonempty">L</td><td class="empty"> </td><td class="nonempty">A</td><td class="empty"> </td></tr><tr><td class="empty"> </td><td class="nonempty">S</td><td class="empty"> </td><td class="nonempty">M</td><td class="nonempty">A</td><td class="nonempty">R</td><td class="nonempty">Y</td><td class="empty"> </td><td class="nonempty">R</td><td class="nonempty">O</td><td class="nonempty">B</td><td class="nonempty">E</td><td class="nonempty">R</td><td class="nonempty">T</td><td class="empty"> </td></tr><tr><td class="empty"> </td><td class="nonempty">I</td><td class="empty"> </td><td class="nonempty">A</td><td class="empty"> </td><td class="nonempty">O</td><td class="empty"> </td><td class="empty"> </td><td class="nonempty">R</td><td class="empty"> </td><td class="empty"> </td><td class="nonempty">S</td><td class="empty"> </td><td class="nonempty">T</td><td class="empty"> </td></tr><tr><td class="empty"> </td><td class="nonempty">C</td><td class="empty"> </td><td class="nonempty">R</td><td class="empty"> </td><td class="nonempty">G</td><td class="nonempty">A</td><td class="nonempty">R</td><td class="nonempty">Y</td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="nonempty">H</td><td class="empty"> </td></tr><tr><td class="nonempty">J</td><td class="nonempty">A</td><td class="nonempty">C</td><td class="nonempty">K</td><td class="empty"> </td><td class="nonempty">E</td><td class="empty"> </td><td class="nonempty">O</td><td class="empty"> </td><td class="nonempty">A</td><td class="empty"> </td><td class="nonempty">F</td><td class="nonempty">R</td><td class="nonempty">E</td><td class="nonempty">D</td></tr><tr><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="nonempty">R</td><td class="nonempty">O</td><td class="nonempty">N</td><td class="nonempty">A</td><td class="nonempty">L</td><td class="nonempty">D</td><td class="empty"> </td><td class="empty"> </td><td class="nonempty">W</td><td class="empty"> </td></tr></table><p>Crozzle file is valid.</p><p>Configuration file is valid.</p><p>Word list file is valid.</p><p>Score = 303</p></body></html> |

## Appendix 1B

The expected result of the toStringHTML() function is generated by using StreamWriter to save the return result to a text file called “html2.txt” which is located in the folder of “unitTest8”. StreamReader was then used to read the expected string from the text file. The HTML below shows the expected and actual result of the test. The highlighted text shows the validity of each but no score is generated as the validate() method was called after the expected and actual results were set.

|  |
| --- |
| <!DOCTYPE html><html><head><style> table, td { border: 1px solid black; border-collapse: collapse; } td { width:24px; height:18px; text-align: center; } </style><style>  .empty { background-color: #777777; }  .nonempty { background-color: #ffffff; }  </style></head><body><table><tr><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="nonempty">B</td><td class="nonempty">E</td><td class="nonempty">T</td><td class="nonempty">T</td><td class="nonempty">Y</td><td class="empty"> </td><td class="nonempty">O</td><td class="nonempty">S</td><td class="nonempty">C</td><td class="nonempty">A</td><td class="nonempty">R</td><td class="empty"> </td></tr><tr><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="nonempty">I</td><td class="empty"> </td><td class="empty"> </td><td class="nonempty">O</td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="nonempty">H</td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td></tr><tr><td class="empty"> </td><td class="nonempty">J</td><td class="nonempty">I</td><td class="nonempty">L</td><td class="nonempty">L</td><td class="empty"> </td><td class="nonempty">M</td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="nonempty">A</td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td></tr><tr><td class="empty"> </td><td class="nonempty">E</td><td class="empty"> </td><td class="nonempty">L</td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="nonempty">H</td><td class="empty"> </td><td class="empty"> </td><td class="nonempty">R</td><td class="empty"> </td><td class="nonempty">M</td><td class="empty"> </td></tr><tr><td class="empty"> </td><td class="nonempty">S</td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="nonempty">A</td><td class="empty"> </td><td class="empty"> </td><td class="nonempty">L</td><td class="empty"> </td><td class="nonempty">A</td><td class="empty"> </td></tr><tr><td class="empty"> </td><td class="nonempty">S</td><td class="empty"> </td><td class="nonempty">M</td><td class="nonempty">A</td><td class="nonempty">R</td><td class="nonempty">Y</td><td class="empty"> </td><td class="nonempty">R</td><td class="nonempty">O</td><td class="nonempty">B</td><td class="nonempty">E</td><td class="nonempty">R</td><td class="nonempty">T</td><td class="empty"> </td></tr><tr><td class="empty"> </td><td class="nonempty">I</td><td class="empty"> </td><td class="nonempty">A</td><td class="empty"> </td><td class="nonempty">O</td><td class="empty"> </td><td class="empty"> </td><td class="nonempty">R</td><td class="empty"> </td><td class="empty"> </td><td class="nonempty">S</td><td class="empty"> </td><td class="nonempty">T</td><td class="empty"> </td></tr><tr><td class="empty"> </td><td class="nonempty">C</td><td class="empty"> </td><td class="nonempty">R</td><td class="empty"> </td><td class="nonempty">G</td><td class="nonempty">A</td><td class="nonempty">R</td><td class="nonempty">Y</td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="nonempty">H</td><td class="empty"> </td></tr><tr><td class="nonempty">J</td><td class="nonempty">A</td><td class="nonempty">C</td><td class="nonempty">K</td><td class="empty"> </td><td class="nonempty">E</td><td class="empty"> </td><td class="nonempty">O</td><td class="empty"> </td><td class="nonempty">A</td><td class="empty"> </td><td class="nonempty">F</td><td class="nonempty">R</td><td class="nonempty">E</td><td class="nonempty">D</td></tr><tr><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="nonempty">R</td><td class="nonempty">O</td><td class="nonempty">N</td><td class="nonempty">A</td><td class="nonempty">L</td><td class="nonempty">D</td><td class="empty"> </td><td class="empty"> </td><td class="nonempty">W</td><td class="empty"> </td></tr></table><p>Crozzle file is valid.</p><p>Configuration file is valid.</p><p>Word list file is valid.</p><p></p></body></html> |

## Appendix 1C

The expected result of the toStringHTML() function is generated by using StreamWriter to save the return result to a text file called “html.txt” which is located in the folder of “unitTest8”. StreamReader was then used to read the expected string from the text file. The HTML below shows the expected and actual result of the test. The highlighted text shows the validity of each file as well as the different score generated from the Validate() method and new crozzle files.

|  |
| --- |
| <!DOCTYPE html><html><head><style> table, td { border: 1px solid black; border-collapse: collapse; } td { width:24px; height:18px; text-align: center; } </style><style>  .empty { background-color: #777777; }  .nonempty { background-color: #ffffff; }  </style></head><body><table><tr><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="nonempty">B</td><td class="nonempty">E</td><td class="nonempty">T</td><td class="nonempty">T</td><td class="nonempty">Y</td><td class="empty"> </td><td class="nonempty">O</td><td class="nonempty">S</td><td class="nonempty">C</td><td class="nonempty">A</td><td class="nonempty">R</td><td class="empty"> </td></tr><tr><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="nonempty">I</td><td class="empty"> </td><td class="empty"> </td><td class="nonempty">O</td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="nonempty">H</td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td></tr><tr><td class="empty"> </td><td class="nonempty">J</td><td class="nonempty">I</td><td class="nonempty">L</td><td class="nonempty">L</td><td class="empty"> </td><td class="nonempty">M</td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="nonempty">A</td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td></tr><tr><td class="empty"> </td><td class="nonempty">E</td><td class="empty"> </td><td class="nonempty">L</td><td class="empty"> </td><td class="empty"> </td><td class="nonempty">O</td><td class="empty"> </td><td class="nonempty">H</td><td class="empty"> </td><td class="empty"> </td><td class="nonempty">R</td><td class="empty"> </td><td class="nonempty">M</td><td class="empty"> </td></tr><tr><td class="empty"> </td><td class="nonempty">S</td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="nonempty">R</td><td class="empty"> </td><td class="nonempty">A</td><td class="empty"> </td><td class="empty"> </td><td class="nonempty">L</td><td class="empty"> </td><td class="nonempty">A</td><td class="empty"> </td></tr><tr><td class="empty"> </td><td class="nonempty">S</td><td class="empty"> </td><td class="nonempty">M</td><td class="nonempty">A</td><td class="nonempty">R</td><td class="nonempty">Y</td><td class="empty"> </td><td class="nonempty">R</td><td class="nonempty">O</td><td class="nonempty">B</td><td class="nonempty">E</td><td class="nonempty">R</td><td class="nonempty">T</td><td class="empty"> </td></tr><tr><td class="empty"> </td><td class="nonempty">I</td><td class="empty"> </td><td class="nonempty">A</td><td class="empty"> </td><td class="nonempty">O</td><td class="empty"> </td><td class="empty"> </td><td class="nonempty">R</td><td class="empty"> </td><td class="empty"> </td><td class="nonempty">S</td><td class="empty"> </td><td class="nonempty">T</td><td class="empty"> </td></tr><tr><td class="empty"> </td><td class="nonempty">C</td><td class="empty"> </td><td class="nonempty">R</td><td class="empty"> </td><td class="nonempty">G</td><td class="nonempty">A</td><td class="nonempty">R</td><td class="nonempty">Y</td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="nonempty">H</td><td class="empty"> </td></tr><tr><td class="nonempty">J</td><td class="nonempty">A</td><td class="nonempty">C</td><td class="nonempty">K</td><td class="nonempty">I</td><td class="nonempty">E</td><td class="empty"> </td><td class="nonempty">O</td><td class="empty"> </td><td class="nonempty">A</td><td class="empty"> </td><td class="nonempty">F</td><td class="nonempty">R</td><td class="nonempty">E</td><td class="nonempty">D</td></tr><tr><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="empty"> </td><td class="nonempty">R</td><td class="nonempty">O</td><td class="nonempty">N</td><td class="nonempty">A</td><td class="nonempty">L</td><td class="nonempty">D</td><td class="empty"> </td><td class="empty"> </td><td class="nonempty">W</td><td class="empty"> </td></tr></table><p>Crozzle file is valid.</p><p>Configuration file is valid.</p><p>Word list file is valid.</p><p>Score = 336</p></body></html> |