**Workshop2**

**SFT221NCC**

**Name: Chaerin Yoo**

**Student ID: 102998234**

|  |  |  |  |
| --- | --- | --- | --- |
| Function: findString | Data | Expected result | Description |
| Test 1(first string, Black box) | str=”abc”  list = {“abc”, “def”, “ghi”}  nstrings = 3 | 0 | Tests to see if a string is found in the first position of string array |
| Test 2(second string, Black box) | str=”def”  list = {“abc”, “def”, “ghi”}  nstrings = 3 | 1 | Tests to see if a string is found in the second position of string array |
| Test 3(last string, Black box) | str="apple"  list = {"banana", "orange", "apple"}  nstrings = 3 | 2 | Tests to see if a string is found in the last position of string array |
| Test 4(empty, White box) | str=””  list = {}  nstrings = 0 | -1 | Compares the function's ability to properly handle an empty string and an empty list. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Function: init | Data | Expected result | Description |
| Test 1(Black box) | ar={1,2,3,4,5,6} value=7  size=6 | ar={7,7,7,7,7,7} |  |
| Test 2(Black box) | ar={0,0,0} value=3  size=3 | ar={3,3,3,} |  |
| Test 3(Black box) | ar={} value=9  size=0 | Nothing |  |
| Test 4(White box) | ar={-23, -43, -56, -45, -65, -67, -91} value=73  size=7 | ar={73, 73, 73, 73, 73, 73, 73} |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Function: add2Cart | Data | Expected result | Description |
| Test 1(Black box) | cart = {items: {}, count: 0}  , item = 2 | 0 | Tests if the function adds a valid item (item 2) to an empty cart successfully. |
| Test 2(Black box) | cart = {items: {1, 2, 3}, count: 3}, item = 2 | 1 | Tests if the function handles an attempt to add a duplicate item (item 2) to the cart correctly. |
| Test 3(White box) | cart = {items: {}, count: 0}, item = 100 | 0 or 1 | Tests if the function handles an attempt to add an item with a high ID value (item 100). Whether this is valid depends on the range of valid item IDs. |
| Test 4(Black box) | cart = {items: {3, 5, 7}, count: 3}, item = 6 | 0 | Tests if the function can add a valid item (item 6) to a cart with a mix of items. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Function: clear | Data | Expected result | Description |
| Test 1(first string, Black box) | str=”abc”  list = {“abc”, “def”, “ghi”}  nstrings = 3 | 0 | Tests to see if a string is found in the first position of string array |
| Test 2(second string, Black box) | str=”def”  list = {“abc”, “def”, “ghi”}  nstrings = 3 | 1 | Tests to see if a string is found in the second position of string array |
| Test 3(last string, Black box) | str="apple"  list = {"banana", "orange", "apple"}  nstrings = 3 | 2 | Tests to see if a string is found in the last position of string array |
| Test 4 | str=””  list = {}  nstrings = 0 |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Did you find more test cases via black box or white box techniques? Do you believe adequate testing could be done with just one of the techniques? Was it easier to develop black box or white box tests? Why is one faster than the other?

* In my experience, I have found that black box techniques are often more effective in locating test cases. The purpose of black box testing is to look into functionality without getting into technical specifics, in contrast to white box testing, which allows for a more in-depth analysis of the internal logic. Even though both methods are necessary, black box testing can be enough for early cover. Because they focus simply on the interface and the outcome that is expected, black box tests are typically simpler to build. When compared to white box tests, which need a more in-depth skills of the codebase, black box tests are typically easier to design because of their seeming simplicity.

Consider how you would set up integration tests for the functions above. What would be your general approach to creating integration tests? Do you need to create additional code to set up and run the test? How will you write code to compare the results to ensure they are correct? How much additional time do you think writing the additional code will take?

* Building up integration tests starts with determining the interfaces and linkages between functions. I create test cases that reflect actual situations; these test cases can contain both edge and typical cases. More code could be needed to set up test environments or to false external dependencies. I will analyze the actual findings in respect to the outcomes that were predicted by making use of comparison functions or claims during the evaluation process. The length of the project will determine how long it takes to write this extra code—a few hours to many days. Its complexity will determine how long it takes to compose.

Create one integration test for the combination of two functions above. Show the code you created to set up the test, execute the test and compare the result to the expected result. You only need to demonstrate one set of test data, but it should be obvious how to add more tests easily. Comment your code to point out the set up, execution, and comparison parts of the code.