SFT221 – Workshop 2

Learning Outcomes

- Create black box test data from function descriptions,
- Create white box test data by examining code,
- Create driver code to execute tests.

Instructions

Black Box Testing

Read each of the following function descriptions and create black box test data to test each of the functions. This will be data that tests the functionality of the individual functions by performing unit tests on the functions. You **should not** refer to the code at this stage – that will come later when you create white box test data.

You can report your results in a format similar to this:

Function: findString	Data	Expected Result	Description
Test 1 (first string, Blackbox)	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			
Test 3			
Test 4			

These functions are part of a simple shopping cart application where the user is asked to select items to add to their cart by item name. The program maintains a list of item names for sale and checks the names entered against the list available for sale. If it finds a match, it adds the item into the shopping cart. The shopping cart is a structure which keeps a list of the items in the cart and the number of items in the cart. Items are indicated by their position in the name array. There is a parallel arral of prices so that once the index of the item in the item name array is known, the index can be used to find the associated price. At the end of the program, an itemized list of the cart contents and prices is shown with a total with taxes calculated.

This is a list of the functions you need to create black box test data for:

```
/*
* Find the position of a string in an array of strings.
* @param str - the string to find
* @param list - the list of strings to search
* @param nstrings - the number of strings in the list
* @returns the position of the string in the list or -1 if not found
```

```
*/
int findString(const char str[], const char list[][MAX_STRING_LEN + 1],
const int nstrings);

/*
* Initialize all members of an array to a single value.
* @param ar - the array to initialize
* @param value - the value to set all array members to
* @param size - the size of the array
*/
void init(int ar[], const int value, const int size);

/*
* Add an item to the shopping cart.
* @param cart - the cart to add to
* @param item - the item to add
* @returns zero on success or non-zero if an error occurs
*/
int add2Cart(struct Cart* cart, const int item);

/*
* Clear input buffer until next newline character.
*/
void clear();
```

White Box Testing

Now, you should consult the code supplied with this workshop. Examine the code for each of the functions and add white box tests to the black box tests you created previously. Be sure to indicate if it is a white box or black box test as shown in the example. You can add the white box tests as additional rows to the table you created for each function.

Deliverables

DueDate:

This workshop is due at **11:59 pm 2 days after your lab day**. Late submits will not be accepted.

You should submit:

- One table of tests for each function. Each table should look similar to the example above and list the test, whether it is black or white box, test data, expected result and an explanation of the test and what it seeks to accomplish.

A Reflection, Research and Assessment

Reflections should consider the questions in depth and not be trivial. Some reflections might require research to properly answer the question. As a rough guideline, the answer to each reflection questions should be about 100 words in length.

- 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?
- 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?
- 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.

Marking Rubric

Comprehensiveness of black box test cases	20%
Comprehensiveness of white box test cases	20%
Quality of test data and explanations	10%
Reflection 1	15%
Reflection 2	15%
Reflection 3	20%